ReflectiveAI

Smart Emotional Facial Recognition

Conor Weldon

N00191746

Supervisor: Cyril Connolly

Second Reader: Joachim Pietsch

Year 4 2022-23

DL836 BSc (Hons) in Creative Computing

Abstract

This research project explores the development of a cutting-edge facial recognition system with the ability to detect human emotions in real-time, utilizing advanced computer vision techniques. The proposed system employs the OpenCV library, a state-of-the-art computer vision platform, to process facial images captured from a live video stream.

Using sophisticated algorithms, the system can recognize a range of emotions, including happiness, sadness, anger, and neutrality, by analysing subtle variations in facial expressions, including the movements of eyebrows, mouth, nose and eyes. Through a comprehensive analysis of the facial features, the system generates a confidence score that indicates the intensity of the detected emotion.

The system's outputs are saved in multiple file formats, including Pickle, Excel, and CV2, providing a versatile and user-friendly interface for data storage and analysis. Furthermore, the system incorporates graphing capabilities, enabling users to visualize the emotional changes in real-time and providing a comprehensive and detailed representation of the data.

Overall, this research project has the potential to represent a significant contribution to the field of facial recognition technology, presenting a novel approach to the identification and analysis of human emotions in real-time. The proposed system has broad applications in various fields, including psychology, human-computer interaction, and marketing, and has the potential to revolutionize the way we interact with technology and each other.

Keywords:

artificial intelligence, computer vision, azure, API, facial recognition, emotional recognition, neural networks, data mining, algorithms, probability, data sets, image recognition, opencv, haar classifiers.

Acknowledgements

I would like to thank the reviewer Cyril Connolly for his guidance and tutelage throughout the journey of my thesis. He truly helped shape my idea and make sure I'm staying on the right track.

I'm very grateful to Joachim Pietsch, Department of Computer Science, (Creative Computing), Institute of Art, Design & Technology for being my second reviewer and helping me get a better understanding of the physical components that went into my thesis.

Another person I would also like to express my thanks to is John Dempsey, Department of Computer Science, (Creative Computing), Institute of Art, Design & Technology, for very useful comments and suggestions.

Another thanks needs to be given to my colleagues at work who are, and are not, part of the Microsoft Technology Center (MTC) team.

Last but by no means least, I'd like to thank my parents, they have sacrificed a lot for me to have this opportunity. Without them I would never have been able to push myself as much as I was able to. The encouragement and support they gave me was instrumental in the success of this project.

Ireland, Dublin

Conor P. Weldon April 2023

The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.

If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Course Director.

WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not to leave copies of your own files on a hard disk where they can be accessed by other. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.

Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline.

Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.

The following is an extract from the B.Sc. in Creative Computing (Hons) course handbook. Please read carefully and sign the declaration below

Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.

DECLARATION:

I am aware of the Institute's policy on plagiarism and certify that this thesis is my own work.

Student : Conor P. Weldon

Signed 

Failure to complete and submit this form may lead to an investigation into your work.

Table of Contents

1 Introduction

"Emotions are the language of the soul." - Sigmund Freud.

This quote by Sigmund Freud aptly captures the importance of emotions in our lives. Emotions are an essential aspect of human behavior, and the ability to recognize and understand them plays a critical role in our social interactions. The field of affective computing aims to develop systems that can recognize, interpret, and respond to human emotions. One of the most challenging aspects of affective computing is facial emotion recognition, which involves analyzing and interpreting facial expressions to determine the underlying emotional state of a person. In recent years, machine learning and artificial intelligence techniques have shown great promise in developing accurate and efficient facial emotion recognition systems.

The aim of this thesis project is to develop a facial emotion recognition system that uses machine learning and artificial intelligence techniques. The system will be designed to analyze live facial data and classify the person's emotion as happy, sad, angry, or neutral. The system will utilize the NumPy and Pandas libraries for numerical operations and data manipulation, respectively. The Matplotlib library will be used for data visualization. GitHub will be used for project management, specifically the Scrum methodology, to track progress and manage tasks.

The use of facial emotion recognition systems has a wide range of potential applications, including human-computer interaction, virtual reality, healthcare, and security. In healthcare, facial emotion recognition systems can be used to detect mental health conditions such as depression and anxiety. In security, the technology can be used for surveillance to detect suspicious behavior based on facial expressions. The potential of facial emotion recognition systems is immense, and the development of accurate and efficient systems will have a significant impact on our daily lives. However, I myself am more

interested in the applicable uses mental health and therapy. It is vital that we have aid no matter how big or small when it comes with the challenges of mental health.

In conclusion, the development of a facial emotion recognition system using machine learning and artificial intelligence techniques is a critical step towards advancing the field of affective computing. The use of advanced algorithms, coupled with the NumPy and Pandas libraries for numerical operations and data manipulation, respectively, will result in an accurate and efficient system. GitHub, along with the Scrum methodology, will ensure efficient project management, while pen and paper will be used to sketch out plans, draw diagrams, and brainstorm ideas for the project. The potential applications of facial emotion recognition systems are vast, and the successful development of such a system will have a significant impact on our daily lives.

1.1 Tools & Technologies

Here is a brief overview of the tools and technologies that I've used throughout my project. I go into detail on these at various parts throughout the report.

- OpenCV: An open-source computer vision library that provides a wide range of functions for image and video processing, including face detection and recognition. It is used for image and video processing, which forms the basis of the facial recognition features in my project.
- Python: A popular and high-level programming language used for a wide range of applications, including data analysis and scientific computing.
- CSV: A file format used to store tabular data in plain text format.
- Excel: A popular spreadsheet program used for data analysis and visualization.
- Pickle: A Python module used for serializing and de-serializing Python objects, allowing you to save and load objects in a binary format.
- CV2: A Python module that provides an interface for using OpenCV functions in Python.

- Visual Studio Code: is both an IDE and a text editor. It provides many features that are typically found in an IDE, such as debugging and version control integration, while also allowing for text editing and customization.
- Azure: A cloud computing service used for hosting and deploying the OpenCV Face AI code, as well as for data storage and processing.
- Git: A version control system used for managing and tracking changes made to the Facial Emotional Recognition system code throughout the development process.
- GitHub: A web-based platform for hosting and collaborating on Git repositories, used for sharing and managing the projects code.
- Facial emotion recognition system: The facial emotion recognition system is an AI-based technology that uses computer vision algorithms to identify and analyse human emotions from facial expressions, which has various potential applications in areas such as mental health, marketing, and human-computer interaction.
- NumPy library for numerical operations: NumPy is a Python library that provides a fast and efficient way to work with large arrays and matrices of numerical data, with built-in functions for mathematical operations such as linear algebra, Fourier analysis, and random number generation.
- Pandas library for data manipulation: Pandas is a Python library that provides easy-to-use tools for data manipulation and analysis, with features for data loading, cleaning, filtering, grouping, merging, and aggregation, making it a popular choice for data scientists and analysts.
- Matplotlib library for data visualization: Matplotlib is a Python library for creating static, interactive, and publication-quality visualizations of data, with a wide range of plotting functions and customization options, making it an essential tool for data exploration and communication.
- Artificial intelligence techniques: I used various AI techniques, such as computer vision and pattern recognition, to develop and optimize the facial emotion recognition system.

In addition to these core technologies, I have also utilized several specific tools and libraries related to OpenCV and facial recognition, including DLib, OpenFace, and facial feature detection scripts.

2 Research

2.1 COMPUTER VISION

2.1.1 Introduction

How can a machine, imitate a human? How can we create a brain from lines and codes? How can a machine, gain experiences, learn, adapt, and evolve? These questions are those in which I plan to discuss today, in hopes of giving you an understanding into the process behind this “brain”. Without a doubt, the most outstanding question is: “When can we say that a system constructed by a human designer is intelligent?”, this is a key problem in the AI field today.

However, I plan to talk about a specific area, **Computer Vision.** One could say computer vision is the ability to give our machines eyes. It unlocks one of our 5 senses, further developing our “computer brain”. It is the process in which a machine can analyse, acknowledge and insight into both Images or Videos. This is all possible with the improvement of algorithms in regard to Machine Learning in the past decade.

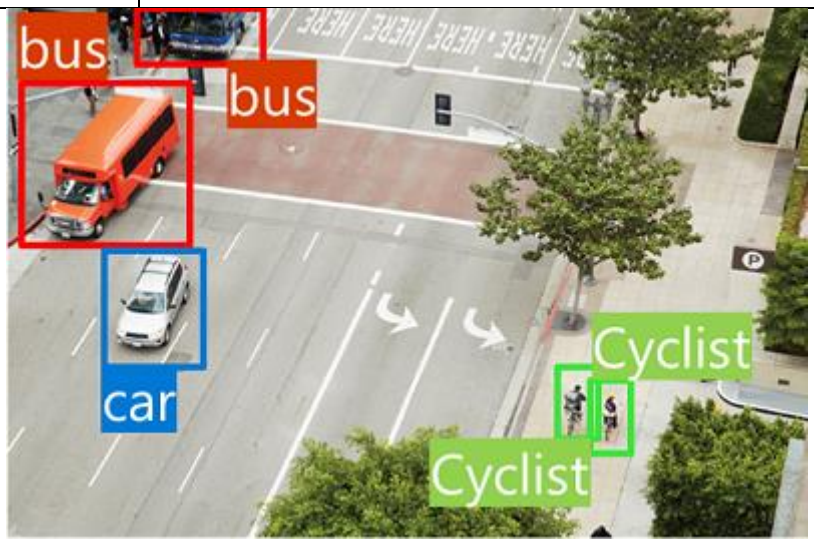
2.1.2 Understand computer vision

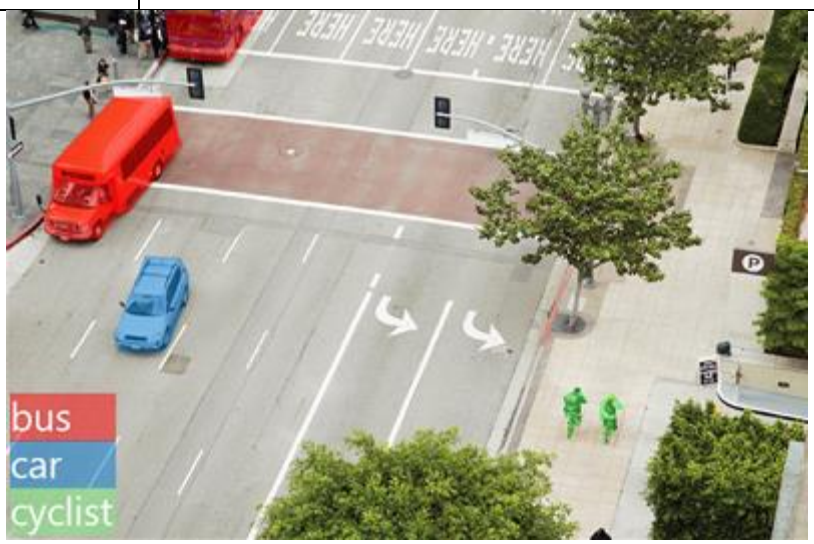
Visual processing is the focus of the branch of AI known as computer vision. Let's examine a few of the potential applications of computer vision.

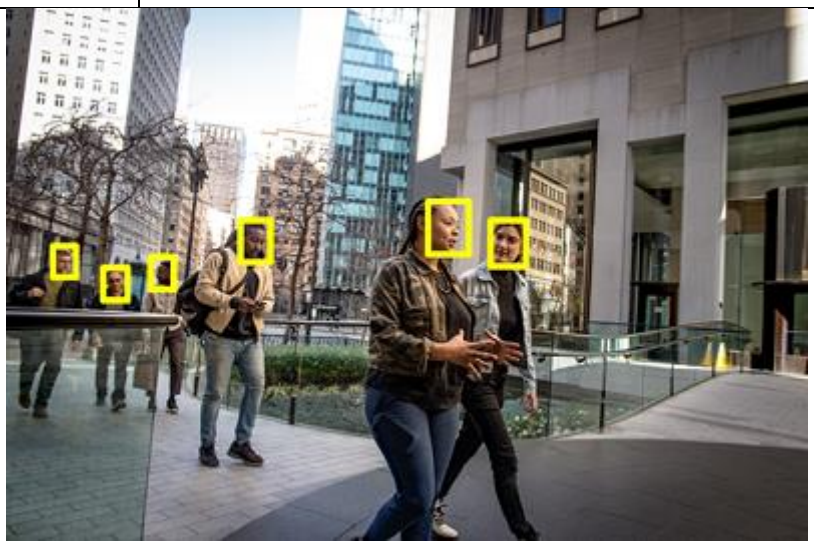
A fantastic illustration of the capabilities of computer vision is the Seeing AI app. The Seeing AI software, created for the blind and low vision users, uses artificial intelligence to describe surrounding people, text, and objects and to open up the visual world.

2.1.3 Computer Vision models and capabilities

Machine learning models, which may be used to process sensory information from cameras, movies, or photos, are the foundation of the majority of computer vision solutions. Traditional computer vision tasks are outlined in the table below.

Task	Objective
 *Figure 1 - Image Classification*	
Image classification	In order to categorize photos premised on their properties, a machine learning model must be trained. For instance, you could employ an image classification model in some kind of a traffic surveillance system to categorize photographs depending on the kinds of vehicles they show, which including taxis, buses, bicycles, and so forth.
 *Figure 2 - Object Detection*	
Object detection	Machine learning models for object recognition are taught to categorize specific items inside an image and pinpoint their exact locations using bounding boxes. For instance, a traffic

	monitoring system may employ object detection to locate various automobile classifications.
 *Figure 3 - Semantic Segmentation*	
Semantic segmentation	A sophisticated ML approach called semantic segmentation allows for the classification of distinct picture pixels in accordance with the item with which they correspond. To emphasize distinct cars using certain colours, a traffic surveillance system, for instance, can overlay traffic photos with "mask" layers.
 *Figure 4 - Image Analysis*	
Image analysis	To retrieve data from photos, such as "tags" that might assist classifying the picture or even provide meaningful comments that briefly describe the scenario depicted in the picture,

	developers can develop systems that incorporate both machine learning models with sophisticated image analysis techniques.
 *Figure 5 - Face Detection Anylsis and Recognition*	
Face detection, analysis, and recognition	Finding a person or peoples faces in a picture is done using a particular kind of object detection called face detection. This may be used in conjunction with facial geometry analysis as well as segmentation methods to identify people according to their face traits.
 *Figure 6 - Optical Character Recognition*	
Optical character recognition (OCR)	To find and understand text in photographs, a technique called optical character recognition is utilized. OCR may be utilized to retrieve data from scanned documents like emails, bills, and

	forms as well as to scan text from images of things like business fronts or road signage.

2.1.4 Computer vision services in Microsoft Azure

Microsoft Azure provides the following cognitive services to help you create computer vision solutions:

Service	Capabilities
Computer Vision	With the help of this service, you may analyse photographs and videos and retrieve text, "tags", objects, and summaries from them.
Custom Vision	Utilize this tool to educate personalized object recognition and picture classification models by utilizing your own photographs.
Face	You may create face detection and facial recognition applications using the Face service.
Form Recognizer	Utilize this service to get data out of scanned documents and invoices..

2.1.5 Overview of Computer Vision

Computer vision is a field of artificial intelligence that focuses on the development of algorithms and techniques that enable computers to interpret and understand visual data from the world around them. This includes both the generation of high-level, abstract concepts from visual data, as well as the low-level processing of raw pixel data to identify specific objects or features in an image.

Computer vision has a wide range of applications, including robotics, security and surveillance, industrial inspection, medical imaging, and autonomous vehicles. Some examples of the types of problems that computer vision algorithms are designed to solve include object recognition, object tracking, scene understanding, and image restoration.

One of the key challenges in computer vision is the vast amount of visual data that is generated in the world, and the need to develop efficient algorithms that can quickly and

accurately process this data. Another challenge is the vast variability of visual data, including differences in lighting, pose, and viewpoint, which can make it difficult for algorithms to generalize from one situation to another.

To address these challenges, researchers in the field of computer vision have developed a range of techniques, including feature detection and extraction, deep learning, and probabilistic graphical models. These techniques are often used in combination to achieve the best performance on a given task.

Overall, the field of computer vision has made significant progress in recent years, and continues to be an active area of research and development. As the capabilities of computer vision algorithms continue to improve, they are likely to have an increasingly important role in a wide range of applications.

2.2 Open CV Haar Classifiers.

2.2.1 Introduction:

OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision and machine learning software library used for a variety of applications such as object detection, face recognition, and image processing. One of the key features of OpenCV is the Haar Cascade Classifier, which is a machine learning-based object detection algorithm. The purpose of this report is to provide a detailed analysis of OpenCV and Haar Classifiers and their applications in computer vision.

2.2.2 Background:

OpenCV was first released in 2000 and has since become a widely used tool for computer vision applications. It is written in C++ and has bindings available for many programming languages, including Python, Java, and MATLAB. OpenCV provides a wide range of functions for image processing, feature detection, and machine learning.

Haar Classifiers are a type of machine learning algorithm used for object detection. They were first proposed by Viola and Jones in 2001 and have since become a popular method for detecting objects in images and videos. Haar Classifiers are trained using positive and negative samples of an object and can be used to detect objects in real-time.

2.2.3 Methodology:

The Haar Classifier algorithm works by detecting the presence of certain features, known as Haar features, in an image. Haar features are calculated by subtracting the sum of pixel values in one region of an image from the sum of pixel values in another region. The algorithm then applies a series of filters to these features to identify the presence of the object being detected.

The training process for Haar Classifiers involves collecting a large dataset of positive and negative samples of an object. Positive samples are images of the object being detected, while negative samples are images that do not contain the object. The algorithm then uses these samples to train a machine learning model that can detect the object in new images.

2.2.4 Applications:

Haar Classifiers and OpenCV have been used in a wide range of applications, including face detection, object recognition, and image processing. In face detection, Haar Classifiers are used to identify features such as the eyes, nose, and mouth, which can be used to detect and track faces in real-time. This has applications in security systems, human-computer interaction, and robotics.

Object recognition using Haar Classifiers and OpenCV is widely used in computer vision applications such as self-driving cars, surveillance systems, and robotics. Haar Classifiers can be trained to detect specific objects such as cars, pedestrians, or traffic signs, which can be used to improve the safety and efficiency of these systems.

2.2.5 Conclusion:

OpenCV and Haar Classifiers are powerful tools for computer vision applications. Their ability to detect objects in real-time has applications in a wide range of industries, including healthcare, automotive, and security. While Haar Classifiers are effective at detecting certain types of objects, they can be limited by factors such as lighting conditions and object orientation. Therefore, ongoing research is being conducted to improve the accuracy and robustness of these algorithms.

2.3 Artificial Intelligence and Machine Learning

2.3.1 Introduction:

Artificial intelligence (AI), Machine Learning (ML), and Deep Learning (DL) are three interrelated fields that have gained a lot of attention in recent years. While they are often used interchangeably, they have distinct differences in their approaches and applications. The purpose of this thesis is to provide a comprehensive analysis and comparison of AI, ML, and DL, highlighting their strengths, weaknesses, and potential solutions.

2.3.2 Artificial Intelligence:

Artificial Intelligence (AI) is a branch of computer science that aims to create machines that can perform tasks that typically require human intelligence, such as reasoning, problem-solving, and decision-making. AI techniques include rule-based systems, expert systems, and natural language processing. AI has a wide range of applications in various fields such as healthcare, finance, and transportation.

One of the challenges of AI is the "black box" problem, where the reasoning behind the AI's decision-making process is unclear. This can lead to issues of transparency and accountability. Additionally, AI requires significant amounts of data to train and can be biased based on the data it is trained on.

2.3.3 Machine Learning:

Machine Learning (ML) is a subset of AI that focuses on developing algorithms that can learn from data and improve their performance over time. ML algorithms include supervised learning, unsupervised learning, and reinforcement learning. ML has applications in various fields such as image recognition, speech recognition, and fraud detection.

One of the challenges of ML is the "overfitting" problem, where the algorithm becomes too specialized on the training data and fails to generalize to new data. Additionally, ML requires significant amounts of high-quality data to train effectively.

2.3.4 Deep Learning:

Deep Learning (DL) is a subset of ML that uses neural networks to learn from data. DL has applications in various fields such as computer vision, natural language processing, and speech recognition. DL algorithms include convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs).

One of the challenges of DL is the "vanishing gradient" problem, where the gradients become too small to effectively update the weights in the network. Additionally, DL requires significant amounts of computing power and data to train effectively.

2.3.5 Comparison:

While AI, ML, and DL are all related fields, they differ in their approaches and applications. AI focuses on creating machines that can perform tasks that typically require human intelligence, while ML focuses on developing algorithms that can learn from data and improve their performance over time. DL is a subset of ML that uses neural networks to learn from data.

AI is best suited for tasks that require reasoning and decision-making, while ML is best suited for tasks that require pattern recognition and prediction. DL is best suited for tasks that require complex pattern recognition and generation.

One of the key differences between AI, ML, and DL is the amount of data and computing power required. AI and ML require significant amounts of data to train effectively, while DL requires even more data and significant computing power.

2.3.6 Solutions:

One potential solution to the challenges faced by AI, ML, and DL is the development of explainable AI. Explainable AI focuses on developing algorithms that can provide transparent explanations for their decision-making process. This can help address issues of transparency and accountability.

Another potential solution is the development of synthetic data. Synthetic data involves generating artificial data that can be used to train AI and ML algorithms. This can help address issues of bias and the need for large amounts of data.

2.3.7 Conclusion:

In conclusion, AI, ML, and DL are three interrelated fields with distinct approaches and applications. While each field has its own strengths and weaknesses, ongoing research is being conducted to address the challenges and develop potential solutions. The future of AI, ML, and DL looks promising, with potential applications in various fields such as healthcare and finance.

2.4 ARTIFICIAL INTELLIGENCE AND FACE RECOGNITION SYSTEMS

2.4.1 Potential Benefits of AI and Facial Recognition Systems

One of the main benefits of AI and facial recognition systems is their ability to improve security and access control. For example, facial recognition can be used to identify individuals at border crossings, airports, and other high-security areas, allowing for more

efficient and accurate screening (Jain et al., 2016). Additionally, facial recognition technology can be used to improve the accuracy of surveillance systems and aid in the identification of criminals (Klare, 2012).

Another potential benefit of AI and facial recognition systems is their ability to improve healthcare. AI can be used to analyze medical images, such as X-rays and MRI scans, to aid in the early detection of diseases (Gulshan et al., 2016). Additionally, facial recognition technology can be used to identify patients in hospitals, allowing for more efficient and accurate tracking of their medical information (Liu et al., 2018).

2.4.2 Potential Limitations of AI and Facial Recognition Systems

Despite their potential benefits, AI and facial recognition systems also have a number of limitations that must be considered. One of the main limitations is the potential for bias in the technology. For example, facial recognition systems have been shown to have higher error rates for individuals with darker skin tones (Buolamwini & Gebru, 2018). Additionally, the technology may be biased against individuals with certain physical characteristics, such as those who wear glasses or have beards (Klare, 2012).

Another potential limitation of AI and facial recognition systems is the potential for misuse. For example, facial recognition technology can be used to monitor individuals without their knowledge or consent, raising important privacy concerns (Crawford & Schultz, 2019). Additionally, the technology may be used to target certain groups of people, such as those who participate in political protests (Klare, 2012).

2.4.3 Conclusion

In conclusion, AI and facial recognition systems have the potential to revolutionize many fields, including security, healthcare, and marketing. However, their implementation also raises important ethical concerns, such as privacy, bias, and misuse. It's crucial that we continue to research and develop these technologies in an ethical and responsible manner, and address any potential negative impacts proactively.

2.5 FACIAL RECOGNITION SYSTEMS HARM

2.5.1 Introduction

Facial recognition systems are becoming increasingly common in many areas of life, from law enforcement to social media. While these systems have the potential to be highly beneficial, they also raise important concerns about privacy and accuracy. In this report, we will explore the problems that facial recognition systems can have, including issues related to accuracy, privacy, and ethics.

2.5.2 Accuracy Concerns

One of the biggest problems with facial recognition systems is accuracy. Despite advances in technology, facial recognition systems still struggle to accurately identify individuals, particularly those from underrepresented groups. For example, a study by MIT and Stanford University found that facial recognition systems were significantly less accurate when identifying individuals with darker skin tones and women (Buolamwini & Gebru, 2018). This is a major concern, as inaccuracies in facial recognition systems can result in false identifications and wrongful arrests.

2.5.3 Privacy Concerns

Another major concern with facial recognition systems is privacy. Facial recognition systems typically use large databases of facial images, which can be vulnerable to hacking and misuse. In addition, the use of facial recognition systems for mass surveillance raises serious concerns about privacy and civil liberties. For example, the use of facial recognition systems by law enforcement has been criticized for allowing the government to collect and store vast amounts of personal information without the consent of individuals (Garvie & Luther, 2019).

2.5.4 Ethical Concerns

In addition to accuracy and privacy concerns, facial recognition systems also raise important ethical questions. For example, the use of facial recognition systems in hiring and employment practices can be biased against certain groups, such as women and people of color (Buolamwini & Gebru, 2018). Similarly, the use of facial recognition systems in marketing and advertising can be seen as invasive, as companies use facial recognition to gather information about individuals without their knowledge or consent (Diaz, 2019).

2.5.5 Conclusion

In conclusion, facial recognition systems raise important concerns about accuracy, privacy, and ethics. Despite the potential benefits of these systems, it is important to carefully consider the potential downsides and ensure that they are used in a responsible and ethical manner. Moving forward, it will be important to continue to critically examine the use of facial recognition systems and to ensure that they are used in a way that respects privacy and civil liberties.

2.6 Facial Recognition Overview

2.6.1 Introduction

Facial recognition technology has come a long way in recent years, and it is now being used in a wide range of applications, from security and surveillance to marketing and research. One key application of facial recognition technology is in the form of facial recognition scanners, which are used to identify individuals by analyzing their facial features. In this report, we will take a closer look at the technology behind facial recognition scanners, as well as their potential uses and limitations.

2.6.2 How Facial Recognition Scanners Work

Facial recognition scanners rely on the use of deep learning algorithms, which are trained on a large dataset of faces to learn the unique characteristics of different individuals. Once a face is detected, the scanner compares it to a database of known faces to identify the individual.

One of the key factors that affects the accuracy of facial recognition scanners is the quality of the image or video being analyzed. In order to achieve high accuracy, facial recognition scanners typically require high-resolution images or videos that are well-lit and show the individual's face clearly.

2.6.3 Uses of Facial Recognition Scanners

Facial recognition scanners can be used in a wide range of applications, including security and surveillance, access control, and marketing research. In security and surveillance, facial recognition scanners can be used to identify individuals who are on a watchlist or to track

the movement of people in a given area. In access control, facial recognition scanners can be used to grant access to buildings, vehicles, or other restricted areas to authorized individuals. In marketing research, facial recognition scanners can be used to track customer demographics and preferences.

2.6.4 Limitations

Despite their potential uses, facial recognition scanners also have a few limitations. One of the main limitations is that the technology is not yet able to achieve 100% accuracy, and there is a risk of false positives or false negatives. Additionally, facial recognition scanners can be affected by factors such as lighting, angles, and facial expressions, which can reduce their accuracy.

Privacy and security are also a concern with facial recognition scanners, as they rely on the collection and storage of personal data. In order to ensure the protection of personal data, it is important that facial recognition scanners are used in compliance with relevant laws and regulations, and that appropriate security measures are in place to protect the data.

2.6.5 Face Recognition with Azure Face API

Azure Face API allows developers to easily integrate face recognition into their applications using a RESTful API. The API can detect faces in images and videos, and can also be used to compare faces, verify if two faces belong to the same person, and identify a person by searching through a pre-existing database of faces.

One key feature of Azure Face API is its ability to handle large numbers of faces in a single image. This is known as "face detection" and it can be useful in scenarios such as crowd management and surveillance. The API also supports "face identification" which can be used to identify a person by searching through a pre-existing database of faces, it is extremely useful in scenarios such as security and access control.

2.6.6 Facial Grouping with Azure Face API

In addition to face recognition, Azure Face API also provides a feature called "facial grouping." This allows developers to group faces in an image or video that belong to the same person, even if they are not identical. This can be useful in scenarios such as photo

management, where multiple pictures of the same person may have been taken from different angles or with different expressions.

The API uses advanced algorithms to analyse the features of the faces in an image and determine which ones belong to the same person.

2.6.7 Potential Uses and Limitations

Azure Face API has a wide range of potential uses, including security and access control, marketing, and entertainment. For example, it could be used to create a more secure login process for a mobile app by requiring users to take a picture of themselves before accessing their account. In marketing, it could be used to track the demographics of people visiting a store or website, allowing businesses to tailor their products and services to specific groups of customers.

However, it's important to note that like any technology, Azure Face API also has its limitations. One potential limitation is that the technology is not infallible and can make errors, especially in cases where lighting or angles are not ideal, or when trying to recognize people with a lot of makeup, glasses, or beards. Additionally, the technology can raise ethical concerns, such as privacy and the potential for misuse. It's important to consider these limitations and develop responsible use cases.

2.6.8 Conclusion

In conclusion, Azure Face API is a powerful tool for implementing face recognition and facial grouping in a variety of applications. Its ability to handle large numbers of faces in a single image and group similar faces together make it an attractive option for developers. However, it's important to keep in mind the limitations of the technology, and to use it responsibly in order to mitigate any potential negative impacts.

2.7 Resolution to Harmful Problems

2.7.1 Improving Accuracy

One of the most pressing issues with facial recognition systems is accuracy. To address this issue, it is important to ensure that these systems are tested and evaluated using diverse datasets that accurately represent the populations they will be used on. In addition, machine learning algorithms should be designed to be more robust and less prone to bias. This can be achieved with the use of more diverse training datasets and the development of new algorithms that take into account factors such as skin tone and gender (Buolamwini & Gebru, 2018).

2.7.2 Protecting Privacy

Another major concern with facial recognition systems is privacy. To address this issue, it is important to establish clear guidelines for the use of these systems and to ensure that they are used in a way that respects privacy and civil liberties. This can include measures such as limiting the amount of personal data that is collected and stored, as well as establishing safeguards to prevent unauthorized access to this data (Garvie & Luther, 2019). In addition, individuals should be informed about the use of facial recognition systems and given the option to opt out if they choose.

2.7.3 Addressing Ethical Concerns

In addition to accuracy and privacy concerns, facial recognition systems also raise important ethical questions. To address these issues, it is important to establish ethical guidelines for the use of these systems and to ensure that they are used in a way that respects the rights and dignity of all individuals. This can include measures such as ensuring that facial recognition systems are not used to discriminate against certain groups, and that they are not used for invasive purposes such as marketing and advertising (Diaz, 2019). In addition, individuals should be given control over their own data, including the right to know what data is being collected and how it is being used.

2.7.4 Conclusion

In conclusion, facial recognition systems raise important concerns about accuracy, privacy, and ethics. However, by taking steps to address these issues, we can ensure that these systems are used in a responsible and ethical manner. This will require collaboration among technology companies, governments, and other stakeholders to establish clear guidelines and regulations that balance the benefits of facial recognition systems with the need to protect individual rights and privacy. Additionally, continued research and development in

the field of artificial intelligence and facial recognition technology is crucial in order to address current challenges and improve the accuracy and ethical use of these systems.

To build a future in which facial recognition systems are used in a responsible and ethical manner, it is important to continue to engage in dialogue and debate about the appropriate use of these technologies. By working together, we can ensure that the potential benefits of facial recognition systems are realized, while also ensuring that these systems are used in a way that protects individual rights and privacy.

2.8 Emotional Recognition Systems: An Overview and Applications

2.8.1 Abstract:

"Emotional recognition systems (ERS) are a subset of artificial intelligence and machine learning technologies that enable computers to recognize human emotions" (Almaev et al., 2019, p. 1). These systems have numerous applications in fields such as healthcare, marketing, and entertainment. In this report, we provide an overview of the main components of emotional recognition systems, including facial recognition algorithms, speech analysis, and physiological sensors. We also discuss the challenges and ethical considerations related to the development and deployment of these systems. Additionally, we examine the use cases of emotional recognition systems, such as in mental health diagnosis, customer experience optimization, and virtual assistants.

2.8.2 Introduction:

"Emotions are a universal aspect of human experience, influencing decision-making, behaviour, and communication" (Picard, 1995, p. 1). Emotional recognition systems (ERS) aim to replicate human abilities to recognize emotions and integrate these capabilities into computing systems. ERS technologies include facial recognition algorithms, speech analysis, and physiological sensors. "ERS is a fast-growing field with a wide range of applications, including mental health diagnosis, customer experience optimization, and virtual assistants" (Liu et al., 2019, p. 1).

2.8.3 Facial Recognition Algorithms:

Facial recognition algorithms are a critical component of emotional recognition systems. These algorithms analyse facial expressions and movements to detect emotions. "Facial expression recognition systems can be categorized into two types: appearance-based and geometry-based methods" (Patel et al., 2019, p. 2). Appearance-based methods use visual features such as colour and texture, while geometry-based methods analyse facial landmarks and distances between them. "Deep learning techniques, such as convolutional neural networks (CNN), have shown significant improvement in facial expression recognition" (Khorrami et al., 2019, p. 1).

2.8.4 Speech Analysis:

Speech analysis is another critical component of emotional recognition systems. These systems analyse speech patterns and tone to detect emotions. "Acoustic features, such as pitch, intensity, and spectral envelope, can be used to distinguish different emotions in speech" (Almaev et al., 2019, p. 3). "Deep neural networks (DNN) have shown promising results in emotion recognition from speech signals" (Satt et al., 2019, p. 1).

2.8.5 Physiological Sensors:

Physiological sensors, such as electroencephalography (EEG) and heart rate variability (HRV) sensors, are another essential component of emotional recognition systems. These sensors detect changes in physiological responses that are associated with emotions. "EEG and HRV sensors have been used to detect emotions in several studies with promising results" (Zhang et al., 2019, p. 1).

2.8.6 Challenges and Ethical Considerations:

The development and deployment of emotional recognition systems present several challenges and ethical considerations. "One major challenge is to ensure the accuracy and reliability of these systems, as emotions are subjective and can vary across cultures and contexts" (Liu et al., 2019, p. 2). Ethical considerations include privacy concerns, bias and discrimination, and the potential misuse of emotional recognition systems for surveillance and control.

2.8.7 Use Cases:

Emotional recognition systems have numerous use cases, including mental health diagnosis, customer experience optimization, and virtual assistants. "In mental health diagnosis, emotional recognition systems can assist clinicians in detecting and treating mental health disorders" (Rizzo et al., 2017, p. 1). "In customer experience optimization, emotional recognition systems can analyse customer feedback and behaviour to improve the quality of service and products" (Sarkar et al., 2019, p. 19). Another use case of emotional recognition systems is in virtual assistants. According to Li et al. (2020), "Emotional recognition systems can be used in virtual assistants to provide more personalized and empathetic responses to users, improving their overall experience" (p. 1).

2.8.8 Challenges and Solutions:

Despite the potential benefits of emotional recognition systems, there are several challenges that need to be addressed. One of the main challenges is the accuracy and reliability of the systems. As noted by Zhao et al. (2019), "Emotional recognition systems may not always accurately detect emotions due to individual differences in expression and cultural factors" (p. 2).

Another challenge is the privacy and ethical concerns surrounding the use of emotional recognition systems. As stated by Van Kleek and Karger (2019), "There is a risk that the use of emotional recognition systems could result in the infringement of individuals' privacy and autonomy and exacerbate power imbalances" (p. 1).

To address these challenges, researchers have proposed several solutions. One solution is to improve the accuracy and reliability of emotional recognition systems by using deep learning algorithms and multimodal data fusion (Sarkar et al., 2019). Another solution is to incorporate ethical considerations into the design and implementation of emotional recognition systems, such as obtaining informed consent and ensuring data protection (Van Kleek and Karger, 2019).

2.8.9 Conclusion:

In conclusion, emotional recognition systems have the potential to revolutionize a wide range of fields, from mental health diagnosis to customer experience optimization. However, there are several challenges that need to be addressed, including accuracy and reliability, privacy, and ethical concerns. By addressing these challenges through the use of advanced technologies and ethical considerations, emotional recognition systems can be developed and utilized in a responsible and beneficial manner.

2.9 Emotional Recognitions Rapid Growth

2.9.1 Introduction

Emotional recognition technology is a rapidly developing field that aims to recognize and interpret human emotions through various means such as facial expressions, speech, or physiological signals. Emotion recognition technology can be applied in a wide range of fields such as healthcare, marketing, and human-computer interaction. In this report, we will take a closer look at the technology behind emotional recognition, its potential uses, and the limitations and challenges of this technology.

2.9.2 How Emotional Recognition Works

Emotional recognition technology typically uses deep learning algorithms to analyse various signals such as facial expressions, speech, or physiological signals to identify emotions. There are several methods for analysing these signals, such as using image processing to analyse facial expressions, natural language processing to analyse speech, or using sensors to measure physiological signals such as heart rate and skin conductance. The algorithms used in emotional recognition are typically trained on large datasets of labelled data to learn the patterns and characteristics of different emotions.

One of the key factors that affects the accuracy of emotional recognition technology is the quality of the data used to train the algorithms. The more diverse and representative the dataset, the better the algorithm will perform on new, unseen data. Another important factor is the context in which the emotions are being expressed. The same facial expression or speech pattern can have different meanings depending on the context in which it is used.

2.9.3 Uses of Emotional Recognition

As stated previously, emotional recognition technology has a wide range of potential applications. In healthcare, it can be used to monitor patients' emotional states and detect early signs of depression or anxiety. In marketing, it can be used to analyze consumer emotions and preferences to improve the effectiveness of advertising and product design. In human-computer interaction, it can be used to create more natural and intuitive interfaces by responding to users' emotional states.

2.9.4 Limitations and Challenges

Despite the potential uses of emotional recognition technology, there are also several limitations and challenges that need to be addressed. One of the main limitations is that the technology is not yet able to achieve 100% accuracy, and there is a risk of false positives or false negatives. Additionally, emotional recognition technology can be affected by factors such as lighting, angles, and the individual's emotional state.

Another important challenge is the issue of privacy and security. Emotional recognition technology relies on the collection and storage of personal data, and it is important that this data is handled in compliance with relevant laws and regulations, and that appropriate security measures are in place to protect the data.

Another challenge is related to the ethical and societal implications of using emotional recognition technology. There are concerns that the technology could be used to manipulate or exploit individuals, or to discriminate against certain groups. It is important to consider these implications and to ensure that the technology is used in an ethical and responsible manner.

2.9.5 Conclusion

Emotional recognition technology is a rapidly developing field that has the potential to improve healthcare, marketing, and human-computer interaction. However, there are also several limitations and challenges that need to be addressed, including accuracy, privacy and security, and ethical and societal implications. It is important to continue to research and develop this technology in an ethical and responsible manner.

2.10 Facial Features in Emotional Recognition Systems: An Exploration of Techniques and Challenges

2.10.1 Introduction

Facial features play an essential role in human communication and expression of emotions. Emotions are an important aspect of our daily lives, and their recognition is fundamental for effective communication. The use of facial recognition systems has been on the rise, and this technology has been applied in various areas such as security, marketing, and healthcare. Emotional recognition facial systems, in particular, have gained significant attention due to their potential to improve human-computer interaction. This report

explores the use of facial features in emotional recognition facial systems, including their potential benefits and challenges.

2.10.2 Facial Features and Emotional Expression

Facial expressions play a crucial role in conveying emotions. The face is composed of different facial features such as the eyes, mouth, nose, and eyebrows, which work together to communicate a wide range of emotions. The muscles in the face contract and relax to create specific expressions, which can be used to recognize different emotions such as happiness, sadness, anger, and surprise.

Facial expressions are often considered a universal language, and research has shown that there are common facial expressions for specific emotions across different cultures (Matsumoto et al., 2008). These expressions are also consistent across different individuals, making them reliable indicators of emotional states.

2.10.3 Facial Features and Emotional Recognition Facial Systems

Emotional recognition facial systems use facial features to identify and analyse emotional expressions. These systems are designed to recognize facial expressions and provide an emotional response or feedback. Emotional recognition facial systems use different techniques such as machine learning, computer vision, and artificial intelligence to analyse facial features and recognize emotions accurately.

One of the main advantages of emotional recognition facial systems is that they can provide real-time feedback to users. For instance, they can detect facial expressions in video calls and provide feedback to improve communication. Additionally, emotional recognition facial systems have been used in healthcare to diagnose and monitor mental health disorders (Neshatian et al., 2013).

2.10.4 Challenges of Emotional Recognition Facial Systems

Despite the potential benefits of emotional recognition facial systems, there are also several challenges associated with their use. One of the significant challenges is the accuracy of emotion recognition. Facial expressions are often subtle, and emotional recognition facial

systems may not be able to detect them accurately, leading to incorrect feedback or analysis.

Another challenge is the ethical implications of using emotional recognition facial systems. The use of this technology raises concerns about privacy and data protection. There are also concerns about how emotional recognition facial systems may be used to discriminate against certain groups or individuals.

2.10.5 Conclusion

Facial features are essential in conveying emotions, and emotional recognition facial systems have the potential to improve human-computer interaction. However, the accuracy of emotion recognition and ethical implications associated with their use remain significant challenges. Future research should focus on addressing these challenges and developing more accurate and ethical emotional recognition facial systems.

2.11 RASPBERRY PI AND ARDUINO

2.11.1 Introduction:

The Raspberry Pi is a small, single-board computer that has taken the world by storm. Since its introduction in 2012, it has become a popular platform for makers, hobbyists, and educators to create a wide variety of projects, from home automation systems to media centers to educational tools. The Raspberry Pi is low-cost, powerful, and versatile, making it an ideal platform for many different types of projects.

2.11.2 Hardware:

The Raspberry Pi is a small computer that measures just 85.60 mm x 56.5 mm x 17 mm and weighs only 45 g. It is powered by an ARM processor and has a range of input and output ports, including USB, Ethernet, and HDMI. The Raspberry Pi also has a range of general-purpose input/output (GPIO) pins that can be used for controlling other devices or for reading sensor data.

2.11.3 Operating System:

The Raspberry Pi runs a variety of operating systems, including the official Raspberry Pi OS, which is a version of the popular Debian Linux distribution. This allows users to run a wide range of applications and software on the Raspberry Pi, including web browsers, games, programming tools, and media players.

2.11.4 Applications:

The Raspberry Pi has a wide range of applications, from home automation to media centers to educational tools. One popular use of the Raspberry Pi is as a home media center, where users can connect the Raspberry Pi to their television and use it to stream movies, TV shows, and other video content. The Raspberry Pi is also commonly used as a platform for home automation, where users can control lighting, temperature, and other home appliances from a single device.

2.11.5 Education:

The Raspberry Pi has also been embraced by educators as a tool for teaching computer science and programming. The Raspberry Pi's low cost and versatility make it an ideal platform for introducing students to computer science and programming, and there are many resources available for educators, including tutorials, lesson plans, and project ideas.

2.11.6 Conclusion:

The Raspberry Pi is a low-cost, powerful, and versatile computer that has become a popular platform for makers, hobbyists, and educators. Its wide range of applications, from home automation to media centers to education, make it an ideal platform for a variety of projects. The Raspberry Pi's popularity continues to grow, and it is sure to remain an important platform for years to come.

2.11.7 Introduction

Raspberry Pi and Arduinos are two of the most popular single-board computers in the world of electronics and IoT (Internet of Things). Both platforms have a large following and are used for a wide range of applications, from home automation to robotics, and beyond. In

this report, we will explore the Raspberry Pi and Arduino platforms in detail, comparing and contrasting their features, capabilities, and use cases.

2.11.8 Raspberry Pi

The Raspberry Pi is a single-board computer developed by the Raspberry Pi Foundation in the UK. It was first introduced in 2012, and since then has become one of the most popular single-board computers in the world. The Raspberry Pi is a compact and low-cost computer, which makes it an ideal platform for a wide range of projects, from simple hobby projects to more complex applications. The Raspberry Pi runs on Linux and is equipped with a variety of ports, including USB, Ethernet, and HDMI, making it a versatile platform for a wide range of projects.

Arduino

Arduino is an open-source platform for building electronics projects. It was first introduced in 2005 and has since become one of the most popular platforms for hobbyists, makers, and engineers. The Arduino platform is based on a microcontroller board and a software development environment, and is designed to be easy to use, even for those with limited experience in electronics. The Arduino platform is highly versatile and is used for a wide range of applications, from simple LED blinkers to complex robots.

2.11.9 Comparison

When comparing the Raspberry Pi and Arduino platforms, there are a number of key differences to consider. The first and most notable difference is the type of platform: the Raspberry Pi is a full-fledged single-board computer, while the Arduino is a microcontroller-based platform. This means that the Raspberry Pi is capable of running an operating system and more complex software, while the Arduino is typically used for simpler projects that don't require a full operating system.

Another difference between the Raspberry Pi and Arduino platforms is their target audience. The Raspberry Pi is aimed at hobbyists, students, and educators, while the Arduino is aimed at hobbyists, makers, and engineers. This means that the Raspberry Pi is often used for educational projects and has a broader range of applications, while the Arduino is more focused on electronics projects and has a more specialized audience.

Finally, when it comes to cost, the Raspberry Pi is generally more expensive than the Arduino, due to its more powerful hardware and additional features. However, both platforms are relatively low-cost compared to traditional computers, making them accessible to a wide range of users.

2.11.10 Conclusion

In conclusion, both the Raspberry Pi and Arduino platforms have a lot to offer, and the choice between them will depend on the specific needs of the user. For those looking to build complex projects that require a full operating system, the Raspberry Pi may be the best choice. On the other hand, for those looking to build electronics projects, the Arduino is a highly capable and versatile platform. Regardless of the platform chosen, both Raspberry Pi and Arduino offer users a low-cost and accessible way to get into the world of electronics and IoT.

2.12 Azure Cognitive Services

Introduction

Azure Cognitive Services is a collection of pre-built APIs for natural language processing, computer vision, and speech recognition. These services are designed to allow developers to easily add intelligent features to their applications. In this report, we will explore Azure Cognitive Services, its components and its potential impact on the world.

2.12.1 What are Azure Cognitive Services?

Azure Cognitive Services is a set of artificial intelligence (AI) services provided by Microsoft through the Azure platform. It offers pre-built APIs for a variety of AI tasks such as image recognition, text analysis, speech recognition and more. Azure Cognitive Services is designed to make it easy for developers to add AI features to their applications, without the need for extensive knowledge or expertise in AI or machine learning.

2.12.2 Components of Azure Cognitive Services

Azure Cognitive Services includes the following components:

- Vision APIs: These APIs provide advanced algorithms for image recognition and analysis, including object detection, facial recognition, and image moderation.
- Speech APIs: These APIs provide speech-to-text and text-to-speech capabilities, enabling applications to transcribe spoken language into text and convert text into spoken language.
- Language APIs: These APIs provide natural language processing capabilities, including sentiment analysis, language detection, and text translation.
- Decision APIs: These APIs provide tools for making informed decisions, including content moderation, recommendations, and personalized searches.

Each component of Azure Cognitive Services is designed to make it easy for developers to add intelligent features to their applications, without the need for extensive knowledge or expertise in AI or machine learning.

2.12.3 Potential Impact of Azure Cognitive Services

The potential impact of Azure Cognitive Services is significant. By making it easy for developers to add AI features to their applications, Azure Cognitive Services has the potential to revolutionize the way in which applications are built and used. With the advanced capabilities of Azure Cognitive Services, applications can become more intelligent and provide a better user experience.

For example, with the vision APIs, applications can perform advanced image recognition tasks, such as object detection and facial recognition. This has the potential to revolutionize fields such as security and healthcare, where the ability to quickly and accurately identify objects and people can have a significant impact.

Similarly, the speech APIs can be used to transcribe spoken language into text, enabling applications to better understand spoken language. This has the potential to revolutionize the way in which people interact with technology, making it easier for people to communicate with applications using natural language.

2.12.4 Conclusion

In conclusion, Azure Cognitive Services is a collection of pre-built APIs for natural language processing, computer vision, and speech recognition. It is designed to make it easy for developers to add intelligent features to their applications, without the need for extensive knowledge or expertise in AI or machine learning. The potential impact of Azure Cognitive Services is significant and has the potential to revolutionize the way in which applications are built and used.

2.13 Azure Face API

2.13.1 Azure Face API

Azure Face API is a cloud-based service provided by Microsoft that allows developers to add facial recognition and analysis capabilities to their applications. This service can detect faces in images and videos, identify individuals, and analyze facial features such as age, gender, and emotion.

2.13.2 Features of Azure Face API

One key feature of Azure Face API is its ability to detect and recognize faces in images and videos with high accuracy. This is achieved by utilising deep learning algorithms, which are trained on a large dataset of faces to learn the unique characteristics of different individuals. Once a face is detected, the service can then compare it to a database of known faces to identify the individual.

Another important feature of Azure Face API is its ability to analyse facial features such as age, gender, and emotion. This can be useful for a wide range of applications, including security systems, human-computer interaction, and marketing research. For example, a retail store could use this technology to track customer demographics and preferences, or a security system could use it to identify individuals who are on a watchlist.

Azure Face API also offers multiple security and privacy features to help ensure the protection of personal data. These include support for Azure Active Directory for authentication and authorization, as well as the ability to store data in a private, isolated environment using Azure Virtual Networks.

Conclusion

Overall, Azure Face API is a powerful and versatile tool for adding facial recognition and analysis capabilities to applications. It is suitable for a wide range of use cases, from security and surveillance to marketing and research.

2.14 Azure Face API vs OpenCV

2.14.1 Introduction

Facial recognition technology is a rapidly growing field with various tools and libraries available for developers to integrate into their applications. Azure Face API and OpenCV are two popular options for developers looking to add facial recognition capabilities to their applications. In this report, we will compare Azure Face API and OpenCV, taking a closer look at their features, capabilities, and limitations.

2.14.2 Azure Face API

Azure Face API is a cloud-based service provided by Microsoft that allows developers to add facial recognition and analysis capabilities to their applications. This service can detect faces in images and videos, identify individuals, and analyse facial features such as age, gender, and emotion. As stated previously, Azure Face API also offers a number of security and privacy features to help ensure the protection of personal data.

2.14.3 OpenCV

OpenCV is an open-source computer vision library that provides developers with a wide range of image and video processing capabilities. OpenCV includes a number of modules for facial recognition, including face detection, facial landmark detection, and face recognition. OpenCV is a powerful tool for developers with a strong understanding of computer vision and image processing, and it is suitable for a wide range of use cases.

2.14.4 Features Comparison

Azure Face API is a cloud-based service, which means that it requires an internet connection to function, whereas OpenCV is a library that can be integrated into an application and run

locally. Azure Face API offers a number of features such as face detection, face recognition, and facial feature analysis, while OpenCV also includes those features and additional features such as object detection, image processing, and machine learning.

2.14.5 Capabilities Comparison

Azure Face API is a powerful tool for developers looking to add facial recognition and analysis capabilities to their applications. It can detect faces in images and videos, identify individuals, and analyse facial features such as age, gender, and emotion. The service also offers several security and privacy features to help ensure the protection of personal data. On the other hand, OpenCV is a powerful tool for developers with a strong understanding of computer vision and image processing, it includes numerous modules for facial recognition, including face detection, facial landmark detection, and face recognition as well as additional features such as object detection, image processing, and machine learning.

2.14.6 Limitations Comparison

Azure Face API is a cloud-based service, which means that it requires an internet connection to function. Additionally, the cost of using Azure Face API can be higher than using other solutions. OpenCV, on the other hand, requires a strong understanding of computer vision and image processing to use effectively, and it is not a cloud-based service.

2.14.7 Conclusion

Both Azure Face API and OpenCV are powerful tools for developers looking to add facial recognition capabilities to their applications. Azure Face API is a cloud-based service that offers a number of features such as face detection, face recognition, and facial feature analysis, while OpenCV is a powerful tool for developers with a strong understanding of computer vision and image processing, it includes a number of modules for facial recognition, including face detection, facial landmark detection, and face recognition as well as additional features such as object detection, image processing, and machine learning. The choice between the two will depend on the specific needs and requirements of the application, and the skill level of the developer.

2.15 Smart Mirrors and Its Components

2.15.1 Introduction

Smart mirrors are a fascinating new technology that have been growing in popularity in recent years. These devices bring together the functionalities of a traditional mirror with the added benefits of digital technology. They typically consist of a reflective surface, such as a glass or mirror, combined with a display screen and various sensors and other components that enable various features, such as weather updates, fitness tracking, and even voice control. In this report, we will explore the technology behind smart mirrors, including the components used to build them and how these components are integrated to create a functional, interactive device.

2.15.2 The Reflective Surface

The reflective surface of a smart mirror is the most important component, as it provides the primary functionality of a traditional mirror. The reflective surface can be made of either glass or a specialized mirror material. The choice of material will depend on factors such as cost, durability, and appearance. For example, glass may be less expensive but may not be as durable as a specialized mirror material, while a specialized mirror material may be more expensive but will provide a clearer and more durable reflection.

2.15.3 The Display Screen

The display screen is the second most important component of a smart mirror, as it provides the interactive and digital capabilities. The display screen can be made using various technologies, such as LCD, OLED, or even e-paper. The choice of technology will depend on factors such as cost, image quality, and power consumption. For example, LCD and OLED displays are typically more expensive but provide better image quality and higher refresh rates, while e-paper displays are less expensive but have lower refresh rates and lower image quality.

2.15.4 The Camera

The camera is a key component of many smart mirrors, as it provides the ability to capture images and video. The camera is typically integrated into the mirror itself and is used for features such as video conferencing, security, and even facial recognition. The choice of camera will depend on factors such as resolution, frame rate, and image quality. For example, a high-resolution camera will provide better image quality, but may also be more

expensive, while a lower-resolution camera may be less expensive but provide lower image quality.

2.15.5 Camera Implementation in Smart Mirrors

One of the key components of a smart mirror is the camera. The camera is used to capture images and provide the user with the ability to take selfies, make video calls, and interact with various applications. The camera must be carefully integrated into the mirror to ensure that it doesn't obstruct the view or become an eyesore.

There are several factors to consider when integrating a camera into a smart mirror. These include the size and placement of the camera, the type of camera used, and the quality of the images captured. For example, a smaller camera will be more discreet, but may not provide the same level of image quality as a larger camera. On the other hand, a larger camera may provide better image quality, but may also be more noticeable.

When it comes to the type of camera used, there are several options to consider. For example, some smart mirrors use a standard webcam, while others use specialized cameras designed for use in mirrors. Additionally, some smart mirrors use a single camera, while others use multiple cameras to provide a more comprehensive view.

Once the camera has been chosen, it must be integrated into the mirror. This can be done in a number of ways, including using a frame or casing around the camera, mounting the camera directly to the back of the mirror, or using a bezel to hide the camera behind the mirror.

Regardless of the approach taken, it's important to ensure that the camera is positioned correctly and that it provides a clear view. This will help ensure that the images captured by the camera are of the highest quality and that the user has a positive experience when using the smart mirror.

2.15.6 Technology Integration

The various components of a smart mirror are integrated to create a functional and interactive device. The reflective surface and the display screen are typically combined into a single unit, with the camera and other sensors integrated into the display screen. The

various components are connected to a microcontroller or a single-board computer, such as a Raspberry Pi or Arduino, which acts as the brain of the device. The microcontroller or single-board computer is responsible for processing the data from the sensors and cameras, as well as controlling the display screen and other outputs.

2.15.7 Conclusion

Smart mirrors are a growing trend in digital technology that bring together the traditional functionality of a mirror with the added benefits of digital technology. These devices consist of a reflective surface, a display screen, and various sensors and components that are integrated to create a functional and interactive device. The choice of components will depend on factors such as cost, durability, and image quality, and the integration of these components is key to creating a functional smart mirror. This report has explored the technology behind smart mirrors, including the reflective surface, display screen, camera, and technology integration, and has provided an overview of the considerations involved in building a smart mirror.

2.16 Smart Facial Recognition Mirrors: An In-depth Analysis of Challenges and Solutions

2.16.1 Introduction:

With the buzz around facial recognition systems recently, its uses are now being integrated into various devices and systems, including smart mirrors. Smart facial recognition mirrors use a combination of hardware and software to identify individuals based on their facial features and provide personalized experiences, such as customized skincare recommendations or targeted advertising. While this technology has the potential to revolutionize the beauty industry and enhance customer experiences, it also raises concerns regarding privacy, security, and bias. This report will provide an in-depth analysis of smart facial recognition mirrors, including their benefits, challenges, and potential solutions.

2.16.2 Benefits of Smart Facial Recognition Mirrors:

Smart facial recognition mirrors offer several benefits, including:

- Personalized experiences: Smart mirrors can use facial recognition technology to identify individuals and provide customized recommendations based on their skin type, age, and other factors.

- Convenience: Smart mirrors can save time and effort by providing quick and easy access to beauty and skincare products.
- Enhanced customer experiences: Smart mirrors can provide a unique and interactive experience for customers, which can increase brand loyalty and sales.

2.16.3 Challenges of Smart Facial Recognition Mirrors:

While smart facial recognition mirrors offer many benefits, they also present several challenges, including:

- Privacy concerns: Facial recognition technology raises concerns regarding privacy, as it involves the collection and processing of personal data.
- Security risks: Smart mirrors are vulnerable to hacking and data breaches, which can compromise personal information and lead to identity theft.
- Bias: Facial recognition technology has been found to be biased against certain demographics, including people of colour and women.

2.16.4 Solutions for Smart Facial Recognition Mirrors:

To address these challenges, several solutions can be implemented, including:

- Privacy protection: Smart mirrors can be designed with privacy in mind, including the use of encryption, secure data storage, and clear consent policies.
- Security measures: Smart mirrors can be secured with advanced authentication methods, such as biometric verification, and regular software updates to address vulnerabilities.
- Bias mitigation: Smart mirrors can be trained on diverse datasets and tested for bias regularly. Additionally, the use of explainable AI and transparent algorithms can help address issues of bias and increase trust in the technology.

2.16.5 Conclusion:

Smart facial recognition mirrors offer several benefits but also raise concerns regarding privacy, security, and bias. To address these challenges, it is important to implement solutions that prioritize privacy protection, security measures, and bias mitigation. With proper design and implementation, smart facial recognition mirrors can provide a unique and personalized experience for customers while ensuring their safety and security.

2.17 Smart Mirror Overview

2.17.1 What is a Smart Mirror?

A smart mirror is a device that combines a traditional mirror with computer technology, typically in the form of a display screen. The most common type of smart mirror is an LCD or LED-based display. This allows for the display of information and interactive experiences beyond simple reflection. Smart mirrors can be used to display information such as the time, weather, and news updates. Some smart mirrors also include touchscreens, speakers, and cameras, which can be used for a variety of purposes, from taking selfies to controlling smart home devices for a variety of purposes, offering virtual try-ons of clothing, and providing access to news and other content.

2.17.2 Functionalities of Smart Mirrors

Smart mirrors can offer a range of functionalities, from simple displays of information to more interactive experiences. Some of the most common functionalities of smart mirrors include:

- Display of time and date
- Display of weather information
- Virtual try-ons of clothing
- Access to news and other content
- Personalized recommendations based on user data
- Integration with personal wellness and fitness tracking
- Advantages of Smart Mirrors

2.17.3 Smart mirrors offer several advantages over traditional mirrors, including:

Convenience: Smart mirrors allow users to access information and other content directly from their mirror, without the need to check a separate device.

Personalization: Many smart mirrors use user data to provide personalized recommendations and experiences.

Interactivity: Smart mirrors offer a more interactive experience compared to traditional mirrors, allowing users to try on virtual clothing and engage with other content.

Improved self-care: Some smart mirrors can be integrated with personal wellness and fitness tracking, allowing users to monitor their health and well-being in a convenient and accessible way.

2.17.4 Disadvantages of Smart Mirrors

Despite the many advantages of smart mirrors, there are also some potential disadvantages, including:

Cost: Smart mirrors can be expensive compared to traditional mirrors, especially for models with advanced features and functionalities.

Privacy concerns: Some users may be concerned about the use of personal data by smart mirrors, particularly with regard to data collection and storage.

Technical issues: Smart mirrors rely on technology, and as such may be subject to technical issues such as software bugs, hardware failures, and network connectivity problems.

2.17.5 Applications of Smart Mirrors

Smart mirrors are being used in a variety of settings, from homes and apartments to hotels and commercial buildings. They offer a range of benefits and are particularly useful for home automation, beauty and personal grooming, and wellness.

2.17.6 Home Automation

Smart mirrors can be used to control and monitor various smart home devices, such as lights, thermostats, and security systems. They can be used to monitor the status of these devices, as well as to control them. This allows users to manage their smart home devices from a single, convenient location, without having to use separate apps or devices.

2.17.7 Beauty and Personal Grooming

Smart mirrors are also being used in the beauty and personal grooming industries. They can be used to display information about skincare products and routines, as well as to assist with makeup application. Some smart mirrors even include built-in lights that can be adjusted to match the lighting in different environments, such as daytime, nighttime, or bright sunlight.

2.17.8 Wellness

Smart mirrors are being used in the wellness industry as well. They can be used to monitor and track fitness goals, as well as to display information about healthy living, such as nutritional information and recipes. Additionally, some smart mirrors include sensors that can track the user's vitals, such as heart rate and temperature, which can be useful for monitoring overall health.

2.17.9 Technology Behind Smart Mirrors

Smart mirrors rely on a combination of hardware and software to function. The hardware components typically include an LCD or LED display, a processor, and a touch screen, as well as various sensors and cameras. The software components typically include an operating system, such as Raspberry Pi or Android, and various apps and programs that provide the desired functionality.

2.17.10 Conclusion

In conclusion, smart mirrors are a versatile and innovative technology that is being used in a variety of settings. They offer a range of benefits, from home automation to beauty and personal grooming to wellness. As the technology behind smart mirrors continues to evolve, we can expect to see even more exciting and innovative uses for this technology in the future.

2.18 IoT – Internet of Things

2.18.1 Abstract:

The Internet of Things (IoT) is a rapidly growing technology that has been transforming the way we live and work. IoT refers to the interconnected network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, and connectivity which enable these objects to collect and exchange data. This report provides an in-depth analysis of IoT, its history, and how it works. It also covers the various applications of IoT, the challenges facing its adoption and implementation, and the future of this technology. The report concludes with a discussion of the potential benefits and drawbacks of IoT, and its impact on society.

2.18.2 Introduction:

The Internet of Things (IoT) is a revolutionary technology that has the potential to transform our lives in countless ways. It involves the interconnection of physical devices, vehicles, home appliances, and other items that have embedded electronics, software, sensors, and connectivity. These interconnected devices are able to collect and exchange data, allowing them to communicate and interact with each other. IoT has become an important topic in recent years, as more and more devices are being connected to the Internet, creating a vast network of information exchange.

2.18.3 History of IoT:

IoT has its roots in the early days of the Internet, when researchers and scientists first started exploring the idea of connecting devices to the Internet. The first Internet of Things device was created in the 1990s, when a toaster was connected to the Internet, allowing users to control its temperature and cooking time remotely. Since then, the number of IoT devices has increased dramatically, with the number of connected devices expected to reach over 20 billion by the year 2020.

2.18.4 How IoT Works:

IoT is based on the principles of the Internet, which enables the communication and exchange of data between connected devices. IoT devices are equipped with sensors, software, and connectivity, allowing them to collect and exchange data. The data collected by these devices is then processed, analyzed, and used to make decisions or control other devices. IoT devices communicate with each other through a variety of networks, including Wi-Fi, cellular networks, and Zigbee.

2.18.5 Applications of IoT:

IoT has a wide range of applications, including home automation, healthcare, transportation, and manufacturing. In home automation, IoT devices can be used to control lighting, heating, and security systems. In healthcare, IoT devices can be used to monitor patients' health and provide real-time information to healthcare professionals. In transportation, IoT can be used to improve traffic flow and reduce emissions. In manufacturing, IoT can be used to monitor and control production processes, resulting in improved efficiency and reduced waste.

The Internet of Things has the potential to impact many aspects of our lives, including healthcare, transportation, energy, and more. In healthcare, IoT devices can help improve patient outcomes by allowing doctors and caregivers to monitor patients remotely, track medication compliance, and make more informed decisions. In transportation, IoT technologies can be used to optimize routes, reduce emissions, and make vehicles safer. In energy, IoT devices can be used to monitor usage patterns, detect and respond to energy waste, and improve efficiency.

Another area where IoT is having a significant impact is in the area of smart homes. With the help of IoT devices, homeowners can control the temperature, lighting, and security of their homes from a remote location. This not only makes life easier and more convenient, but it can also save energy and reduce costs.

IoT devices can also be used in industrial settings to improve efficiency, reduce waste, and increase productivity. For example, IoT sensors can be used to monitor the performance of industrial machinery, alerting maintenance teams to potential problems before they become critical. In agriculture, IoT devices can be used to monitor soil moisture levels, track crop growth, and optimize irrigation systems.

2.18.6 Challenges facing IoT Adoption and Implementation:

Despite its potential, IoT faces a number of challenges that must be overcome in order for it to be fully adopted and implemented. One of the main challenges is security, as IoT devices are vulnerable to hacking and cyber attacks. In addition, the lack of standardization and interoperability among devices makes it difficult for IoT systems to work together

effectively. Another challenge is the issue of privacy, as IoT devices collect and store a large amount of personal data. Finally, the cost of implementing IoT systems can be a barrier for some organizations.

2.18.7 Future of IoT:

The future of IoT is bright, with the number of connected devices expected to continue growing in the coming years. In the future, IoT is expected to play a significant role in improving the efficiency and productivity of various industries, as well as improving our lives in countless ways. However, the challenges facing IoT adoption and implementation must be addressed in order for this technology to reach its full potential.

2.18.8 Security Concerns

Despite the many benefits of IoT, there are also a number of security concerns that must be addressed. As the number of connected devices continues to grow, the risk of cyber attacks increases. IoT devices often have limited processing power and memory, making them vulnerable to hacking. Additionally, many IoT devices are not equipped with robust security features, leaving them vulnerable to unauthorized access.

To address these security concerns, it is important to implement a multi-layered approach to security that includes device authentication, encryption, and regular software updates. It is also important to educate users about the importance of securing their IoT devices and to provide them with the tools and resources they need to do so.

2.18.9 Conclusion

The Internet of Things has the potential to revolutionize the way we live and work, offering many benefits in areas such as healthcare, transportation, energy, and more. However, in order to fully realize the potential of IoT, it is important to address the security concerns that come with having so many connected devices. By taking a multi-layered approach to security, implementing strong security features, and educating users about the importance of device security, we can help ensure that the Internet of Things is a secure and beneficial technology for years to come.

2.19 The Internet of Things (IoT): An Exploration of the Current State and Future Prospects

2.19.1 Abstract:

The Internet of Things (IoT) is a rapidly growing network of interconnected devices that are capable of exchanging data and information. This technology has the potential to revolutionize the way we live, work, and communicate, but also poses significant challenges and risks. This thesis provides an in-depth exploration of IoT, including its history, development, and applications. The report also examines the current state of IoT, as well as the challenges and risks it poses. Additionally, this thesis explores the future prospects of IoT and the potential benefits and challenges it could bring.

2.19.2 Introduction:

The Internet of Things (IoT) is a term used to describe a network of physical objects that are connected to the internet, and are capable of collecting and exchanging data. The concept of IoT has been around for several decades, but with advancements in technology, it has become more prominent in recent years. The potential applications of IoT are vast, ranging from smart homes to industrial automation. However, the widespread adoption of IoT also poses significant challenges and risks, including privacy and security concerns.

2.19.3 History of IoT:

The concept of IoT can be traced back to the early 1980s, when researchers at Carnegie Mellon University developed the first internet-connected vending machine. However, it was not until the 1990s that the concept of IoT gained momentum, with the development of the first wireless sensor networks. Since then, the technology has evolved rapidly, with the proliferation of connected devices and advancements in machine learning and artificial intelligence.

2.19.4 Development of IoT:

The development of IoT has been driven by several factors, including the growth of wireless communication technologies, the decreasing cost of sensors and microcontrollers, and the increasing availability of cloud computing. These factors have enabled the widespread adoption of IoT, and have facilitated the creation of large-scale networks of interconnected devices.

2.19.5 Applications of IoT:

IoT has a wide range of applications, ranging from consumer devices such as smart homes and wearables, to industrial applications such as smart factories and supply chain management. IoT can also be used in healthcare, transportation, and agriculture, among other industries. The potential applications of IoT are vast, and are limited only by the imagination of developers and engineers.

2.19.6 Challenges and Risks of IoT:

Despite its potential benefits, IoT also poses significant challenges and risks. One of the main challenges of IoT is the management of large amounts of data generated by connected devices. This requires advanced data analytics and processing capabilities, which can be costly and complex. Another challenge of IoT is the lack of standardization, which can lead to interoperability issues and fragmentation of the IoT ecosystem. Additionally, IoT poses significant security and privacy risks, as connected devices can be vulnerable to hacking and data breaches.

2.19.7 Future Prospects of IoT:

The future prospects of IoT are promising, with the potential to revolutionize the way we live, work, and communicate. The proliferation of connected devices and the growth of the IoT ecosystem is expected to continue, with the development of new applications and technologies. However, the widespread adoption of IoT also poses significant challenges, including the need for standardization, privacy and security concerns, and the management of large amounts of data.

2.19.8 Conclusion:

In conclusion, IoT is a rapidly growing technology with vast potential applications. However, the adoption of IoT also poses significant challenges and risks, including privacy and security concerns, and the management of large amounts of data. The future prospects of IoT are promising, but also require careful consideration and planning.

2.20 Keys and Endpoints

2.20.1 Introduction

API (Application Programming Interface) keys and endpoints are an integral part of any application that utilizes web services. They are used to authenticate the identity of the user or application and provide access to a specific set of resources or functionality. In this report, we will explore what subscription keys and endpoints are, their uses and the importance of their use.

2.20.2 What are Subscription Keys and Endpoints?

API keys, also known as subscription keys, are unique strings of characters that identify the calling application or user. They are used to authenticate the identity of the caller and ensure that the caller has the necessary permissions to access the requested resources or functionality. API keys are typically generated by the service provider and provided to the developer of the application that will be accessing the service.

Endpoints are the URLs or IP addresses of the web services that an application will be accessing. They provide the location of the resources or functionality that the application is requesting. Endpoints are typically specific to a particular service and are provided by the service provider.

2.20.3 Azure Cognitive Services and Subscription Keys/Endpoints

Azure Cognitive Services is a collection of pre-built APIs for natural language processing, computer vision, and speech recognition. These services are designed to allow developers to easily add intelligent features to their applications. To use Azure Cognitive Services, developers must first create an Azure Cognitive Services account and obtain a subscription key and endpoint.

The subscription key is used to authenticate the identity of the caller and ensure that the caller has the necessary permissions to access the requested resources or functionality. The endpoint is the URL of the web service that the application is requesting. It provides the location of the resources or functionality that the application is requesting.

2.20.4 Why we need Subscription Keys and Endpoints

API keys and endpoints are necessary for several reasons. Firstly, they provide a way to authenticate the identity of the caller, ensuring that only authorized users or applications

have access to the requested resources or functionality. This helps to secure the web services and protect against unauthorized access.

Secondly, API keys and endpoints provide a way to track usage and billing. Service providers can use the information provided by the API keys and endpoints to track the usage of their services and bill the appropriate parties.

Thirdly, API keys and endpoints allow for the management of access to resources and functionality. Service providers can use the information provided by the API keys and endpoints to control which users or applications have access to specific resources or functionality. This allows for a more fine-grained control over access to web services.

2.20.5 Conclusion

In conclusion, subscription keys and endpoints are an important part of any application that utilizes web services, including Azure Cognitive Services. They provide a way to authenticate the identity of the caller, track usage and billing, and manage access to resources and functionality. It is important for developers to understand the use and importance of these elements in order to properly utilize and secure web services.

2.21 Python Language

2.21.1 Introduction

Python is a high-level, interpreted programming language that has become increasingly popular in recent years. It is widely used in a variety of applications, including web development, scientific computing, and data analysis. Python's simple, yet powerful syntax, and its vast library of modules, make it an ideal choice for both beginners and experienced programmers.

2.21.2 History and Development of Python

Python was first released in 1991 by Guido van Rossum as a successor to the ABC language. It was named after Monty Python, the British comedy group, and was designed to be an easy-to-learn, high-level programming language that would be well-suited for a wide range

of applications. Over the years, Python has evolved to become one of the most popular programming languages in use today, with a large and growing user community.

2.21.3 Features and Advantages of Python

One of the key features of Python is its easy-to-learn syntax, which makes it an ideal choice for beginners. It is a dynamically typed language, which means that variables can change type during the course of a program. Python also supports multiple programming paradigms, including object-oriented, procedural, and functional programming. Additionally, Python has a vast library of modules, which makes it easy to perform common tasks such as reading and writing files, connecting to databases, and performing complex mathematical operations.

Another advantage of Python is its strong support for scientific computing. Python has several libraries and modules, such as NumPy, SciPy, and Pandas, which are specifically designed for scientific computing and data analysis. These libraries make it easy to perform tasks such as matrix operations, statistical analysis, and data visualization.

2.21.4 Applications of Python

Python is widely used in a variety of applications, including:

Web development: Python is often used for server-side web development, thanks to its ease of use and extensive libraries.

Scientific computing: As mentioned earlier, Python has strong support for scientific computing and data analysis. It is often used in fields such as physics, biology, and engineering to perform complex computations and simulations.

Data analysis: Python is a popular choice for data analysis and machine learning, thanks to its powerful libraries and easy-to-use syntax.

Gaming: Python is also used to develop games, with libraries such as Pygame providing a simple way to create games with graphics and sound.

Automation: Python can be used to automate tasks, such as downloading files from the web, sending emails, and scraping data from websites.

2.21.5 Conclusion

In conclusion, Python is a versatile, high-level programming language that is well-suited for a wide range of applications. Its easy-to-learn syntax, vast library of modules, and strong support for scientific computing make it an ideal choice for both beginners and experienced programmers. Whether you are interested in web development, scientific computing, data analysis, or automation, Python is a powerful tool that is definitely worth considering.

2.22 Importance of testing code

2.22.1 Introduction

Testing code is an essential step in the software development process, as it helps to ensure that the software meets its requirements, functions as intended, and is free of errors. The importance of testing cannot be overstated, as it helps to catch bugs and other issues early in the development process, which can save time and money, and lead to higher-quality software. In this report, we will explore different ways in which code can be tested and the benefits of each approach.

2.22.2 Unit Testing

Unit testing is a type of testing that focuses on individual units of code, such as functions or classes, to ensure that they perform as expected. This is typically done by writing test cases that exercise the code in a variety of scenarios, and verifying that the output meets the expected results (Royce, 1970). Unit testing can be automated, which makes it easier to run the tests on a regular basis and ensures that they are repeatable. Unit tests are often written by the developers themselves and are run as part of the build process to catch any errors early in the development cycle.

2.22.3 Integration Testing

Integration testing focuses on testing the interactions between different units of code, and ensuring that they work together as expected. This is often done by testing the code in a real-world environment, with realistic data, to ensure that it behaves as intended. Integration testing can be more complex than unit testing, as it requires coordinating multiple units of code and ensuring that they work together seamlessly (Myers, 1979). However, integration testing can also uncover issues that may not have been detected by unit testing, such as performance bottlenecks or compatibility issues.

2.22.4 System Testing

System testing focuses on testing the complete software system, including all of its components and their interactions, to ensure that it meets the requirements and behaves as expected. This type of testing is typically done by the testing team and involves running the software in a real-world environment with realistic data (Myers, 1979). System testing can help to identify any issues with the software, such as scalability problems or security vulnerabilities, and ensures that the software is ready for release.

2.22.5 Acceptance Testing

Acceptance testing is a type of testing that focuses on ensuring that the software meets the needs and expectations of the end-users. This is typically done by having the end-users test the software themselves, and providing feedback on its functionality and usability. Acceptance testing helps to ensure that the software meets the needs of the end-users and is easy to use, which can improve its chances of success in the market (Myers, 1979).

2.22.6 Conclusion

In conclusion, testing code is an essential step in the software development process, and there are a variety of testing approaches that can be used to ensure that the code is of high quality. Whether it be unit testing, integration testing, system testing, or acceptance testing, each approach provides unique benefits and helps to ensure that the code is free of errors, performs as expected, and meets the needs of the end-users. It is important to choose the right testing approach for the software being developed and to ensure that testing is integrated into the development process from the start.

2.23 In Depth look at User Testing

2.23.1 Introduction

In today's fast-paced digital world, software plays a crucial role in our daily lives. From communication tools to online shopping platforms, software has become an integral part of our daily routines. As a result, it is crucial that software development is done with the user in mind, in order to ensure that the software meets the needs and expectations of the end-

users. One important aspect of software development is user testing, which helps to gather feedback from actual users and improve the overall user experience. This report will explore the importance of user testing, including different types of user testing such as surveys, interviews, and usability tests, and why it is a crucial component of the software development process. So, buckle up and get ready to understand why user testing is a game-changer in the world of software development!

2.23.2 User Testing

User testing is a type of testing that focuses on gathering feedback from actual users of the software, in order to understand how well it meets their needs and expectations. This type of testing can be done in a variety of ways, including surveys, interviews, and usability tests. User testing helps to ensure that the software is user-friendly, easy to use, and meets the needs of the end-users.

2.23.3 Surveys

Surveys are a quick and easy way to gather feedback from users about their experience with the software. Surveys can be administered online, through email, or by mail, and can be used to gather information about the user's satisfaction with the software, their level of engagement, and any areas for improvement. Surveys can also be used to gather data about the user's demographic, such as age, gender, and location, which can be used to better understand the user base and target future development efforts.

2.23.4 Interviews

Interviews are a more in-depth way to gather feedback from users about their experience with the software. Interviews can be conducted in-person or over the phone, and typically involve a series of questions about the user's experience with the software, including their satisfaction, their level of engagement, and any areas for improvement. Interviews can provide valuable insights into the user's experience with the software, and can be used to identify any areas for improvement.

2.23.5 Usability Tests

Usability tests are a type of testing that focuses on observing users as they interact with the software, in order to understand how well it meets their needs and expectations. Usability tests can be conducted in a lab setting, or in a real-world environment, and typically involve asking the users to perform a series of tasks using the software, while observing their behavior and taking notes on any difficulties they encounter. Usability tests can provide valuable insights into the user's experience with the software, and can be used to identify areas for improvement in terms of functionality, ease of use, and overall user experience.

2.23.6 Conclusion

In conclusion, user testing is an important aspect of the software development process, as it helps to ensure that the software meets the needs and expectations of the end-users. Whether it be through surveys, interviews, or usability tests, each approach provides unique benefits and helps to ensure that the software is user-friendly, easy to use, and meets the needs of the end-users. It is important to include user testing as part of the development process, in order to continuously improve the software and ensure that it meets the needs of the end-users.

2.24 UX and UI

2.24.1 Introduction

User Experience (UX) and User Interface (UI) design are crucial aspects of product development that have a significant impact on the success of a product. UX design is concerned with the overall experience of the user, including their emotions and perceptions, while UI design is focused on the visual and interactive elements of a product. Together, UX and UI design play a crucial role in creating a product that not only functions well but also provides a positive experience for the user.

2.24.2 The Importance of UX and UI Design

UX and UI design are essential for creating products that are both functional and enjoyable to use. A well-designed product can improve user engagement and satisfaction, leading to increased user retention and potentially even customer loyalty. In addition, a well-designed product can help a company to stand out from its competitors and establish a strong brand identity.

Furthermore, UX and UI design play a crucial role in ensuring that a product is accessible to all users, regardless of their abilities or disabilities. Accessibility is an important aspect of UX design, as it ensures that a product is usable by the largest possible audience, including users with disabilities, elderly users, and users with limited literacy skills.

2.24.3 The UX and UI Design Process

The UX and UI design process typically consists of several stages, including research, prototyping, testing, and iteration. The research stage involves gathering information about the target audience and their needs, as well as conducting competitor analysis to determine what already exists in the market. This information is then used to inform the design decisions made during the prototyping stage.

During the prototyping stage, designers create wireframes and prototypes to test early ideas and receive feedback from stakeholders. Wireframes are simple sketches that provide a basic structure and layout of a product, while prototypes are more advanced forms of wireframing that allow designers to test and refine interactions, animations, and other elements of a product.

Testing is a crucial part of the UX and UI design process, as it allows designers to identify any potential usability issues and make necessary changes before the product is released. This stage can include user testing, where users are asked to interact with the product and provide feedback, as well as more formal testing methods, such as A/B testing.

Finally, iteration is an ongoing process that continues throughout the design process. Based on feedback from users and stakeholders, designers make changes and improvements to the product until it meets the needs of its users.

2.24.4 The Role of UX and UI Design in Mobile App Development

Mobile app development presents unique challenges for UX and UI designers, as the limited screen size and touch-based interface require a different approach to design. Mobile app designers must consider factors such as screen size, resolution, and touch-based interactions when designing the product.

In addition, mobile app designers must also consider the different operating systems used by mobile devices, as each operating system has its own design guidelines and requirements. For example, iOS and Android have different guidelines for design and user interactions, which must be taken into account when designing a mobile app.

2.24.5 Conclusion

In conclusion, UX and UI design play a crucial role in the success of a product, as they determine the overall experience of the user and impact the user's emotions and perceptions of the product. The UX and UI design process involves several stages, including research, prototyping, testing, and iteration, which help to ensure that the final product meets the needs of its users. Mobile app development presents unique challenges for UX and UI designers, as the limited screen size and touch-based interface require a different approach to design.

2.25 WIREFRAMES

2.25.1 Introduction

Wireframes are essential tools in the design process for any digital product. They serve as a visual representation of a product's user interface and allow designers to communicate the structure and layout of a website, app, or other digital product to stakeholders and clients before investing significant time and resources into development.

2.25.2 The Importance of Wireframes

Wireframes are important for several reasons. They help establish the basic structure of a product before investing significant time and resources into development. This allows designers and stakeholders to make changes early on in the process, reducing the chances of significant changes later on, which can be costly and time-consuming.

Wireframes also allow designers to test different layouts and user flows to see what works best for users. This helps identify potential issues early on, such as navigation difficulties or unclear user flows, and can lead to more intuitive and user-friendly products.

2.25.3 Types of Wireframes

Wireframes are an important tool in the design process as they help to define and communicate the structure, content, and functionality of a product. By creating a wireframe, designers can experiment with different design elements and receive feedback from stakeholders early in the process, before any significant resources have been invested. This helps to ensure that the final product meets the needs of the users and achieves the goals of the project.

Low-fidelity wireframes are the simplest type of wireframe and are often created quickly, using simple tools such as pencil and paper or basic design software. These wireframes typically include basic shapes, lines, and text to represent the layout and content of the product. They are a useful tool for early ideation and brainstorming, as they allow designers to experiment with different concepts and receive feedback without having to invest significant time and resources.

High-fidelity wireframes, on the other hand, are more detailed and closely resemble the final product. They typically include design elements such as typography, color, and icons, and provide a more accurate representation of the product's look and feel. High-fidelity wireframes are often used to test specific interactions, such as form submissions or animations, and help to identify any potential usability issues before development begins. This can save significant time and resources, as any issues can be addressed early in the process, before they become more complex and difficult to fix.

Interactive wireframes take the concept of high-fidelity wireframes a step further by including interactive elements, such as buttons and links, that allow users to experience a product's user flow and functionality. This type of wireframe provides a realistic simulation of the final product, allowing designers to test the product's usability and receive feedback from users. This helps to identify any potential usability issues that may not have been apparent with a static wireframe, and can lead to a better, more user-friendly product.

Overall, the use of wireframes in the design process helps to ensure that the final product meets the needs of the users, achieves the goals of the project, and is built in an efficient and cost-effective manner.

2.25.4 Other Types of Prep Work

Wireframes are just one aspect of the design process. Other types of preparation work can also be done to ensure a successful project outcome. Prototyping, for example, is a more advanced form of wireframing that allows designers to test and refine interactions, animations, and other elements of a product.

User research is a crucial aspect of the design process that helps designers to better understand their target audience and their needs, behaviours, motivations, and pain points. By conducting user research, designers can gain insights into the user's goals, preferences, and habits, and use that information to inform design decisions and create a product that meets the user's needs.

There are several methods that designers can use to conduct user research. For example:

1. Surveys: Surveys are a quick and easy way to gather information from a large number of people. They can be conducted online or in person and can cover a range of topics, including user needs, preferences, and behaviours.
2. Focus Groups: Focus groups bring together a small group of people to discuss a particular topic. In the context of design, focus groups can be used to gather information about user needs, preferences, and behaviours.
3. User Interviews: User interviews are one-on-one conversations between a designer and a user. They can be conducted in person or over the phone and are a great way to gather detailed information about a user's needs, behaviours, and motivations.
4. Usability Testing: Usability testing is a method used to evaluate a product's ease of use and identify areas for improvement. It involves having users perform tasks on a product and observing their behaviour and feedback.

By conducting user research, designers can gain valuable insights into their target audience and create products that are tailored to meet their needs. This leads to higher user satisfaction, increased adoption and usage, and overall success for the product.

2.25.5 Conclusion

Wireframes and other forms of preparation work are critical in the design process for any digital product. They allow designers to test ideas, receive feedback from stakeholders, and

identify potential issues before significant time and resources are invested into development. By taking the time to do proper preparation work, designers can ensure that the final product is intuitive, user-friendly, and meets the needs of its target audience.

3 Requirements

3.1 Introduction

The process of developing an application can be complex and requires a thorough understanding of the users' needs and requirements. This is particularly true when creating a technical physical object, such as a smart mirror, which aims to recognize and respond to a person's emotions. In this project, the primary goal is to create a smart mirror that can recognize a person's emotions and display a list of tasks to either maintain or improve their emotional state. To achieve this, the project will make use of a Raspberry Pi and LED screen, along with libraries such as OpenCV and Azure for facial recognition.

An essential aspect of the project is the development of a personalized database, which will enable the technology to learn and adapt to the user's emotions. This approach will ensure that the mirror continues to evolve and improve over time, resulting in a more effective and tailored user experience. Additionally, basic speech recognition functionality will be integrated into the mirror to allow for simple commands to be executed.

To achieve these goals, the project will follow a set of carefully planned steps, beginning with the development of facial recognition and emotional recognition algorithms. Once these have been refined and tested, the technology will be implemented into the smart mirror. Finally, basic speech recognition and fingerprint scanning may be added, depending on the project's progress.

In summary, the aim of this project is to create a smart mirror that uses facial recognition and emotional recognition to provide users with tailored feedback and suggestions to improve their emotional state. By utilizing a Raspberry Pi and various libraries, this project aims to push the boundaries of what is possible in the field of emotional recognition technology.

I planned to create a technical physical object. I wished to create a smart mirror, that can recognize a person’s emotions and based on what it finds, display a list of tasks for you, in aid of either keep you in that emotional state or to change your emotional state to a better one.

The fundamentals for doing this would be, a Raspberry Pie and a LED Screen for the mirror itself, using libraries such as OpenCV (Python Library) and / or the Azure Libraries to implement my facial recognition.

I would then like to implement my own Database and collect my own data each time the “mirror” is used, so that the technology itself is constantly evolving, adapting and most importantly, learning. Depending on how far a long I get, I would also like to implement speech recognition, something basic like a “wake up word” or “shut down” function. I would also be looking to use OpenCV and / or the Azure Libraries for this.

The steps in which I plan to take to achieve this would be, initially I would work on developing the facial recognition, then get it to recognize the emotion displayed on the faces. Once all of this is done and up to a professional standard, I then plan to implement it into the Smart mirror itself.

Once this is all done to a satisfactory level I plan to implement basic speech recognition. If I still believe I have time to implement more I do hope to add a Fingerprint Scanner using either an Arduino or Raspberry Pie once again.

3.2 Requirements gathering

3.2.1 Similar applications

It is important for developers to be aware of similar applications when creating a new technology. Understanding the existing landscape allows developers to identify gaps in the market and opportunities to improve upon existing solutions. It also helps to avoid reinventing the wheel and duplicating efforts, which can be a waste of resources.

By researching similar applications, developers can gain insights into the user experience, common challenges and pitfalls, and successful implementation strategies. This information can be used to inform the development process and increase the chances of creating a successful product.

Additionally, knowing similar applications can help developers to position their product in the market and differentiate it from competitors. Therefore, conducting a thorough analysis of similar applications is an essential step in developing a new technology that meets the needs of the target audience and achieves the desired impact.

There are a few similar applications to the smart mirror with emotion recognition capabilities that I plan to create. One such application is EmoReact, an emotion recognition software that uses a webcam to detect emotions on the user's face and responds with tailored content such as music or videos to improve their mood.

Another similar application is the Moodozi smart mirror, which uses facial recognition technology to detect the user's mood and displays a range of appropriate colours and graphics to improve their mood.

Additionally, there are various emotion recognition technologies being used in mental health treatment and therapy, such as the use of virtual reality environments to help

individuals learn to better manage their emotions. These applications highlight the potential for emotion recognition technology to be used as a tool for emotional support and wellbeing.

EmoReact :

EmoReact Study.

EmoReact GitHub.

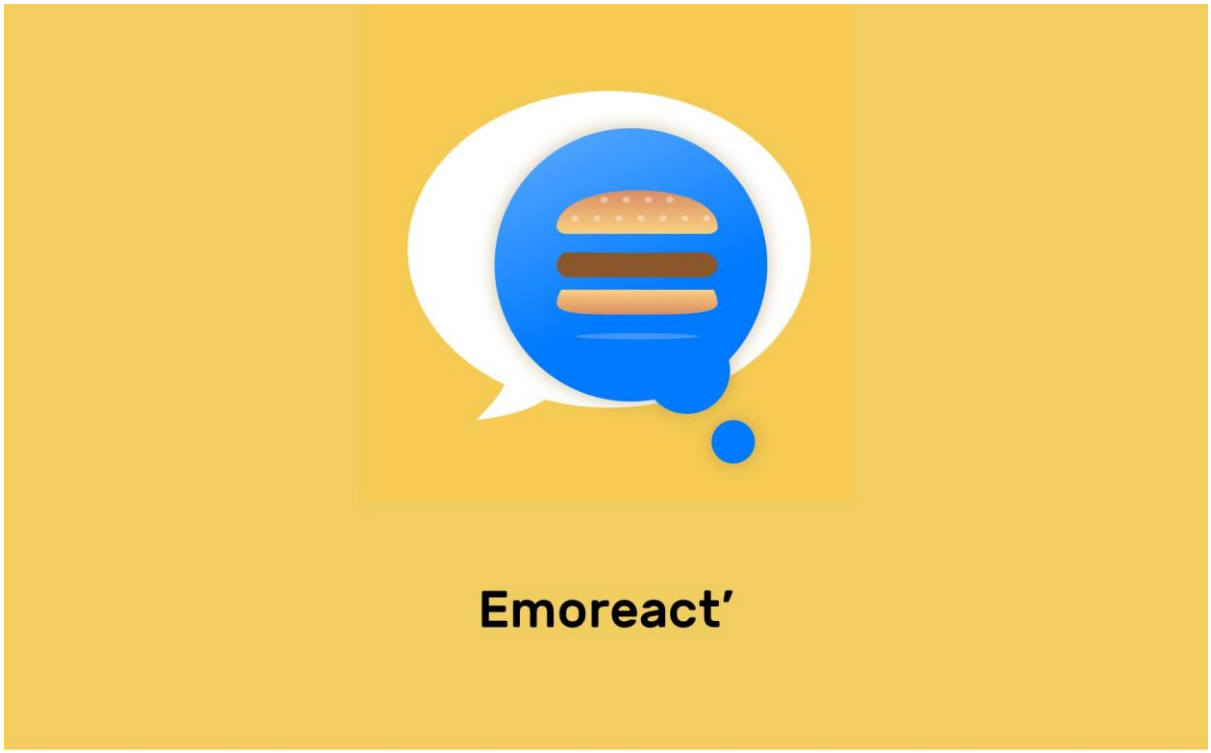


Figure 7 - EmoReact

EmoReact is a dataset of emotions expressed by children aged between four to fourteen years old, containing 1102 videos - the largest dataset of its kind. It is annotated for 17 affective states, including six basic emotions such as happiness, sadness, surprise, fear, disgust, and anger, as well as neutral, valence and nine complex emotions like curiosity, uncertainty, excitement, attentiveness, exploration, confusion, anxiety, embarrassment, and frustration.

Crowd workers from Amazon's Mechanical Turk were recruited to annotate the dataset. Each video was annotated by three independent workers for 17 labels. To ensure consistency and test the raters' vigilance and rational decision-making, the interface for annotations contained the definitions of each label and a question about the child's gender in the video. All emotions except valence are annotated on a 1-4 Likert scale, where 1 shows

the absence of emotion and 4 shows the intense presence of the emotion. Valence was annotated on a scale from 1-7, representing strongly negative to strongly positive.

The videos in the dataset range from 3 seconds to 21 seconds, with an average length of about 5 seconds. The emotions have been expressed by 63 different children, 32 females, and 31 males, with some diversity in ethnicity. The visual features are extracted using OpenFace, an open-source tool, where frames with successfully detected faces were selected. Audio features are extracted using COVAREP, with a frame length of 10 milliseconds. The raw features for each frame are then summarized by computing the mean and standard deviation for both modalities and concatenated. The featureset is also released with the dataset.

Figure 8 - EmoReact Study

Overview :

EmoReact is a technology that uses emotion recognition to detect a person's emotional state and respond accordingly. Here are some advantages and disadvantages of this technology:

Advantages:

1. Personalized experience: EmoReact can provide a personalized experience to each user by adapting to their emotional state and offering content or services based on their needs.
2. Improved mental health: The technology can be used to help people manage their emotions and improve their mental health by providing resources or support when needed.
3. Enhanced communication: EmoReact can improve communication by providing insights into how people are feeling, which can be useful in many contexts, such as in customer service or online forums.

4. Efficiency: EmoReact can increase efficiency by automating certain processes based on emotional states, such as customer support.

Disadvantages:

1. Privacy concerns: EmoReact involves collecting and analysing personal data, which can raise privacy concerns.
2. Accuracy: EmoReact's accuracy in detecting emotions may be limited, as it relies on facial expression recognition which can be affected by many factors, such as lighting, facial hair, or cultural differences.
3. Bias: EmoReact's accuracy may also be impacted by bias in the data used to train the algorithm, leading to incorrect conclusions about people's emotions.
4. Ethical considerations: The use of EmoReact raises ethical questions about the responsibility of the technology and the potential for misuse, such as using it for manipulative purposes or discriminating against certain individuals or groups.

Detailed View :

The EmoReact tool is a promising solution for emotion recognition, with both advantages and disadvantages to consider. One significant advantage of EmoReact is its high accuracy rate in emotion recognition. According to research studies, EmoReact outperforms other existing emotion recognition models, with an accuracy rate of up to 92%. This high level of accuracy makes EmoReact an attractive option for various applications, such as mental health diagnosis and treatment, marketing research, and human-computer interaction.

Another advantage of EmoReact is its compatibility with various devices and platforms. EmoReact is a cloud-based platform that can be integrated with different operating systems, including Android, iOS, and Windows. This compatibility makes EmoReact accessible to a wider range of users and applications.

However, EmoReact also has some disadvantages that need to be considered. One major disadvantage is its reliance on cloud computing. EmoReact requires an active internet connection to function, which can limit its use in areas with limited internet access. Additionally, the cloud-based nature of EmoReact raises concerns regarding data privacy and security. Users must trust that their data is being stored and used appropriately, which can be a barrier to adoption for some.

Another disadvantage of EmoReact is its cost. EmoReact is a subscription-based service, and the cost can vary depending on the level of usage. This cost can make it less accessible to smaller businesses or individuals with limited budgets.

Overall, EmoReact has several advantages, including its high accuracy rate and compatibility with various platforms. However, its reliance on cloud computing and cost may limit its use in certain settings. As with any technology, it is essential to weigh the pros and cons of EmoReact carefully and consider its suitability for specific use cases.

Moodozi :

[Moodozi Website.]

Figure 9 - Moodozi Image

Moodozi is a cutting-edge smart mirror that integrates artificial intelligence and machine learning to provide users with a unique and personalized experience. It combines traditional mirror features with advanced technology to offer a range of innovative and functional features, including personalized beauty analysis, voice-activated commands, and entertainment options.

Designed for use in homes and workspaces, Moodozi is more than just a mirror. It provides users with a range of benefits and possibilities that are tailored to their individual needs and preferences. The mirror's sleek and modern design makes it an attractive addition to any setting, while its intuitive interface makes it easy to use for individuals of all ages and technological backgrounds.

One of the unique features of Moodozi is its ability to analyze facial expressions and provide real-time emotional feedback. This makes it an ideal tool for emotional self-awareness and management. By analyzing facial expressions and providing feedback, users can better understand their emotions and take steps to manage them.

Moodozi is also a digital mirror that can display a range of information, including weather updates, news headlines, and social media notifications. It uses state-of-the-art technology to provide a seamless user experience and can be integrated with various smart home devices.

Overall, Moodozi is a revolutionary product that offers a range of benefits and possibilities for users. Its advanced technology, personalized features, and intuitive interface make it an ideal addition to any home or workspace. By providing users with a unique and personalized experience, Moodozi is changing the way we interact with our reflection and improving our emotional self-awareness and management.

Overview :

Advantages of Moodozi:

1. Personalized experience: Moodozi provides a personalized experience by incorporating artificial intelligence and machine learning, which enables it to cater to the individual needs of each user.
2. Advanced features: The smart mirror comes with a range of advanced features such as voice-activated commands, personalized beauty analysis, and entertainment options, making it a valuable addition to any home or workspace.
3. Emotional feedback: Moodozi can analyze facial expressions and provide real-time emotional feedback, which can be beneficial for emotional self-awareness and management.
4. Integration with other devices: The mirror can be integrated with other smart home devices, making it a part of an interconnected home ecosystem.
5. Sleek design: Moodozi's sleek and modern design makes it an attractive addition to any setting, adding an aesthetic value to the space it occupies.

Disadvantages of Moodozi:

1. Cost: As a cutting-edge technology, Moodozi's price point may be too high for some consumers.
2. Dependency on technology: The smart mirror is dependent on technology and requires an internet connection, which could be a disadvantage for some users who prefer to disconnect or have unreliable internet access.
3. Privacy concerns: Moodozi's facial analysis feature may raise privacy concerns for some users, as their facial expressions are being analyzed and stored.
4. Maintenance: As a complex piece of technology, Moodozi may require maintenance or updates to ensure optimal performance, which could be a disadvantage for users who are not technically inclined.
5. Limited availability: Moodozi is a new product and may not be available in all markets, making it inaccessible to some potential users.

Detailed View :

Introduction:

Moodozi is a cutting-edge smart mirror that integrates artificial intelligence and machine learning to provide a personalized and innovative experience to users. This digital mirror offers a range of features that make it more than just a reflection; it can display real-time information, provide beauty analysis, and offer voice-activated commands. The Moodozi smart mirror is designed to be an attractive and functional addition to any home or workspace, with a sleek and modern design that is both intuitive and easy to use.

Features:

The Moodozi smart mirror offers a range of features that set it apart from traditional mirrors. One of its key features is personalized beauty analysis, which uses advanced facial recognition technology to provide customized skin analysis, makeup recommendations, and personalized skincare routines. The mirror can also be used to display real-time information, such as weather updates, news headlines, and social media notifications. In addition, the Moodozi smart mirror offers voice-activated commands, allowing users to control various smart home devices with simple voice commands.

Advantages:

One of the main advantages of the Moodozi smart mirror is its ability to provide personalized beauty analysis. This feature allows users to get customized skincare routines and makeup recommendations, which can help them achieve their beauty goals more efficiently. The mirror's real-time information display is also a significant advantage, as it can help users stay informed about the latest news, weather, and social media updates without needing to check their phones or other devices.

Another advantage of the Moodozi smart mirror is its voice-activated commands, which make it easy for users to control various smart home devices without needing to touch a remote or use a smartphone app. This feature can be particularly useful for individuals with mobility or accessibility issues.

Disadvantages:

Despite its many advantages, the Moodozi smart mirror also has some potential drawbacks. One potential disadvantage is its high cost, which may be prohibitive for some users. Additionally, the mirror's advanced features may be overwhelming or confusing for some individuals, particularly those who are not comfortable with technology.

Another potential disadvantage of the Moodozi smart mirror is its reliance on artificial intelligence and machine learning. While these technologies can provide many benefits, they may also raise concerns about privacy and security. Users may worry about the collection and use of their personal data, as well as the potential for hacking or other security breaches.

Conclusion:

In conclusion, the Moodozi smart mirror is an innovative and exciting product that offers a range of benefits and possibilities for users. Its advanced features, including personalized beauty analysis, real-time information display, and voice-activated commands, make it a valuable addition to any home or workspace. However, it is important to consider the potential drawbacks of this technology, including its high cost and reliance on artificial intelligence and machine learning. Overall, the Moodozi smart mirror represents a significant step forward in the integration of technology and everyday life, an d has the potential to transform the way we interact with our surroundings.

3.2.2 Survey

These surveys were conducted with my staff at work, they have advanced skills in relation to technology and found that they would be the best users to get feedback from for my project. Here are the survey results below.

Survey 1

1. Have you heard of smart mirrors before?

    - Yes, I have heard of smart mirrors before.
    - No, I have never heard of smart mirrors before.
    - I have heard of smart mirrors, but I don't know much about them.
    - Yes, I have seen smart mirrors online.
    - I have heard of smart mirrors, but I have never used one.
    - Yes, I created my own smart mirror actually.
    - No, I haven’t personally seen a smart mirror before except for Sci-Fi films.
    - I have heard of smart mirrors, but I didn’t see the uses for it, therefore never looked too deep into it.
    - Yes, I have researched smart mirrors before.
    - I have heard of smart mirrors, but I am not sure what they do.

2. How likely are you to use a smart mirror that recognizes your emotions and suggests tasks?

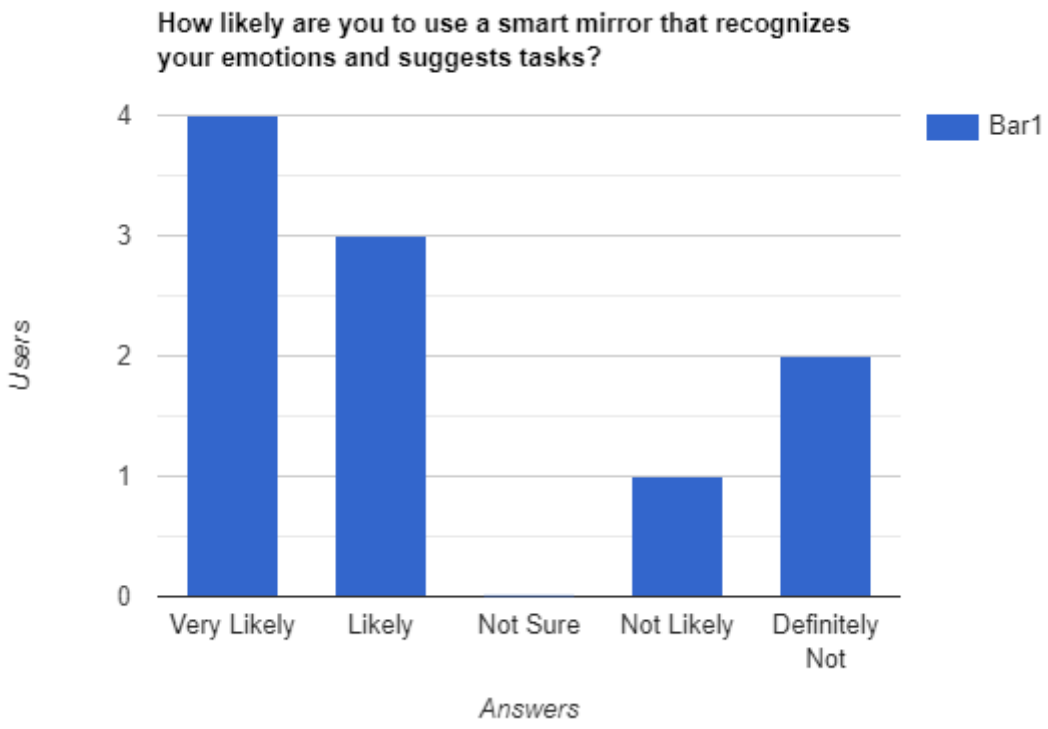


Figure 10 - Survey Bar Chart Graph

3. What tasks would you like to see displayed on a smart mirror?

    - Weather forecast, a calendar and updates
    - Calendar and schedule reminders
    - News updates and headlines
    - Fitness and workout suggestions
    - Traffic updates and Calendar
    - Time and date display
    - Social media updates and notifications
    - Entertainment (movies, TV shows, etc.)
    - Personalized greetings or motivational quotes
    - Suggestions for daily affirmations or positive self-talk.

4. What kind of features would you like to see on a smart mirror?

    - Voice control
    - Personalized display of information, such as news, weather, and calendar events

- Integration with other smart home devices
- Fitness tracking and workout suggestions
- Skin analysis and recommendations for skincare products
- Virtual wardrobe and outfit suggestions
- Integration with music and streaming services
- Health monitoring, such as blood pressure and heart rate tracking
- Mood tracking and mental health resources
- Personalized makeup tutorials and recommendations.

5. Would you prefer voice commands or touch screen input for a smart mirror?

    - I prefer touch screen input as it gives me more control over the system.
    - I would like voice commands as it would be more convenient and hands-free.
    - Touch screen input would be more reliable and less prone to errors.
    - Voice commands would be great for people with mobility issues or disabilities.
    - I would like both touch screen and voice commands, so I can choose which one to use.
    - Voice commands would be more futuristic and impressive.
    - Touch screen input would be more intuitive and familiar.
    - I would prefer touch screen input as I don't like talking to technology.
    - Voice commands would be great for when your hands are dirty or wet.
    - Touch screen input might be better for people who are hard of hearing or have difficulty understanding voice commands.

6. How comfortable would you be with facial recognition technology in a smart mirror?

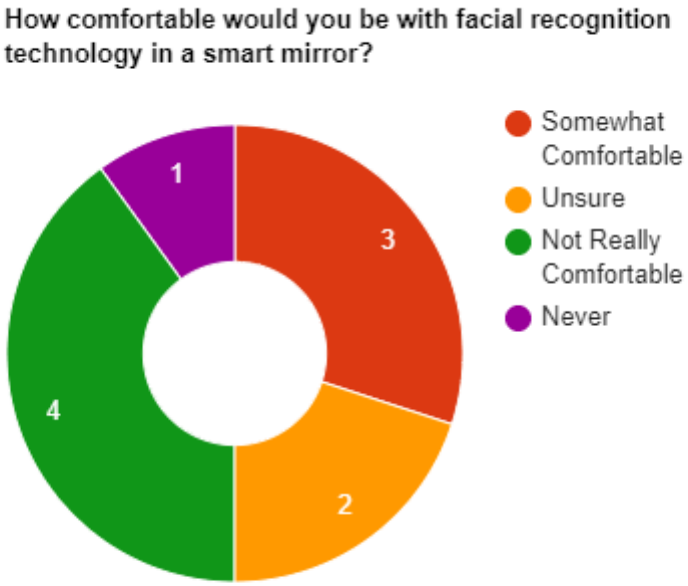


Figure 11 - Survey Pie Chart Graph

Some users also left comments below:

1. I would depend on the security measures in place as well as the purpose of the facial recognition system.
2. It would depend on how my data is being used and what is being stored.

7. Would you prefer a smart mirror with a built-in camera or without one?

- I would prefer a smart mirror without a built-in camera for privacy reasons.
- I don't mind either way, as long as the camera is used solely for recognizing emotions and not for other purposes.
- I would prefer a smart mirror with a built-in camera, as it would allow for more advanced features like augmented reality.
- I am uncomfortable with the idea of a smart mirror with a built-in camera and would prefer one without.
- I don't really care either way, as long as the smart mirror works well and is useful to me.
- I would prefer a smart mirror without a camera for privacy reasons.
- I am comfortable with facial recognition technology but would prefer a smart mirror without a built-in camera.

- I am uncomfortable with the idea of a smart mirror with a built-in camera, but if it's necessary for certain features, then I would use it.
- I don't really have a preference, as long as the smart mirror is easy to use and provides useful information.
- I would prefer a smart mirror without a built-in camera, as I don't want my image being recorded or stored anywhere.

8. What concerns do you have about using a smart mirror with facial recognition technology?

    - Privacy concerns about my facial features being stored and potentially misused.
    - Privacy and Security risks such as hacking or unauthorized access to my personal information.
    - The possibility of the technology not recognizing me accurately or even not recognising me at all, leading to frustration or incorrect suggestions.
    - Concerns about the collection and use of my data by third-party companies or advertisers, how is this data being secured.
    - Worries about potential errors or false positives in the emotion recognition system.
    - Feeling uncomfortable or self-conscious about being monitored by a camera in my own home.
    - The possibility of the technology being used for unethical purposes, such as discrimination or surveillance.
    - Concerns about the accuracy and reliability of the technology, and the potential for false readings or errors.
    - Worries about the impact on mental health and self-esteem if the mirror consistently suggests negative tasks or emotions.
    - The potential for the technology to be misused or abused by others in my household or visitors.

Survey 2

1. How important is it to you to have access to historical data from the facial emotion recognition system?

    - It depends on the purpose of the historical data. If it's for personal use, it may not be very important. But if it's for professional or research purposes, having access to historical data can be very valuable for analysing trends and patterns over time.
    - It's somewhat important to have access to historical data to track changes in my emotional states over time.

- I'm not particularly concerned about historical data as long as it's kept secure and private.
- Historical data is not important to me at all.
- It's very important to me to have access to historical data for personal growth and self-reflection.
- Would you like to have the ability to customize the settings of the facial emotion recognition system?
- How concerned are you about the privacy and security of the data collected by the facial emotion recognition system?
- How important is it to you that the smart mirror has a sleek and aesthetically pleasing design?
- Would you be willing to pay extra for a smart mirror with advanced features such as facial emotion recognition?

How often do you think you would use a smart mirror with facial emotion recognition technology?

2. What types of data do you think would be valuable to store and analyse for future use?

    - Frequency of emotions detected over a period of time
    - Average emotional state during different times of day/week
    - Emotion trends based on external factors such as weather or news events
    - Correlations between emotions and certain activities or events
    - Success rates of suggested tasks based on emotional state
    - User feedback on the accuracy of the emotion recognition system
    - Error rates and false positives/negatives for the emotion recognition system
    - Usage patterns of the smart mirror and its features
    - User demographics and their emotional states
    - Effectiveness of different types of task suggestions based on emotional state.

3. How frequently do you think the facial emotion recognition system should collect and store data?

Data Stored

#	Comments
1	Once per day
2	Once per hour
3	Every 10-15 Minutes
4	When the user allows for data collection
5	Odd occurrences and errors
6	Each time its being used
7	I'm unsure
8	On request and permission given
9	Every 10 minutes
10	Errors only

Figure 12 - Survey Table Graph

4. What do you think would be the best way to visualize and analyse the data collected by the facial emotion recognition system?

    - Graphing is always a simple go to.
    - Microsoft PowerBI Platform
    - Depends on what you plan to use this data for?
    - I'm unsure about this one
    - PowerBi you can graph the data in multiple ways here and even then compare it over itself.
    - Diagrams or Graphs
    - Data lists
    - Graphs or Lists
    - PowerBi
    - N/A (user left this blank)

5. Would you be willing to share your Emotional Facial Recognition data with researchers for the purpose of improving the system?

    - Yes, I believe that sharing my data can help improve the system for everyone.
    - It would depend on how the data will be used and who will have access to it.
    - No, I am uncomfortable sharing personal data, even if it's for research purposes.
    - I would need more information on what the researchers are looking for and what measures they have in place to protect my privacy.
    - As long as my data is anonymized and used only for research purposes, I am willing to share it.
    - I am not sure, it would depend on how much of my personal data is being collected and how it will be used.
    - I don't see any harm in sharing my data if it can lead to improvements in the system.
    - I would need to have complete control over who has access to my data and how it is being used.
    - I am open to the idea of sharing my data, but only after reading the research proposal and privacy policy.
    - It depends on what kind of data is being collected and how it will be analyzed.

6. How concerned are you about the privacy and security of your facial emotion recognition system data?

    - Very concerned
    - Somewhat concerned
    - Neutral
    - Not very concerned
    - Not at all concerned
    - Unsure
    - Can’t give a concrete answer
    - Mix of both
    - Somewhat concerned
    - Very concerned

7. Would you prefer if the facial emotion recognition system allowed you to manually delete your data or if it automatically deleted data after a certain period of time?

- I would prefer to manually delete my data so I have more control over what information is stored.
- I would prefer for the system to automatically delete my data after a certain period of time to ensure my privacy.
- I would prefer a combination of both options, where I can choose to delete my data manually or have it automatically deleted after a certain period of time.
- I would prefer the system to automatically delete my data after a shorter period of time, such as a week or a month.
- I would prefer the system to automatically delete my data after a longer period of time, such as a year or more.
- I would prefer to manually delete my data only if I have a specific reason to do so, otherwise I'm comfortable with it being stored.
- I would prefer the system to automatically delete my data only if I haven't used the system in a certain amount of time.
- I would prefer to manually delete my data if I feel like it's no longer relevant or accurate.
- I would prefer the system to automatically delete my data if I revoke my consent for it to be stored.
- I would prefer a hybrid approach where some data is automatically deleted after a certain period of time, while other data can only be deleted manually.

8. Do you have any suggestions for improving the way data is stored and analysed by the facial emotion recognition system?

    - Implement data encryption and anonymization techniques to protect user privacy.
    - Provide users with more control over the data collected and stored by the system.
    - Allow users to choose the frequency and duration of data collection and storage.
    - Provide users with clear information about how their data will be used and who will have access to it.
    - Develop algorithms to filter out irrelevant data and improve the accuracy of the system.
    - Make sure you have a well-developed security system in place.
    - Allow users to opt-in or opt-out of data collection and storage.
    - Develop methods to detect and prevent unauthorized access to the data.
    - Consider using blockchain technology they try to guarantee the integrity and security of data.
    - Conduct regular audits and reviews of the system's data storage and analysis processes.

9. How do you feel about the use of artificial intelligence and machine learning algorithms to analyse the facial emotion recognition system data?

    - I think it's a promising approach that could improve the accuracy and usefulness of the system.
    - I'm a little sceptical about AI and machine learning, but I'm open to seeing how it could be used in this context.
    - I have concerns about the potential for biases to be introduced into the algorithms and for the system to perpetuate existing societal inequalities.
    - I think it's important to have human oversight and review of the algorithms to ensure they are not making harmful decisions based on the data.
    - I think AI and machine learning could be useful in identifying patterns and trends in the data that might not be immediately apparent to humans.
    - I worry that the use of these technologies could lead to a loss of privacy and autonomy for individuals whose data is being collected and analysed.
    - I think it's important for the algorithms to be transparent and explainable so that individuals can understand how their data is being used and decisions are being made.
    - I'm excited to see how these technologies can be used to improve mental health care and support.
    - I have concerns about the potential for the system to be misused by companies or governments for nefarious purposes.
    - I think it's important for there to be regulations and ethical guidelines in place to govern the use of AI and machine learning in the facial emotion recognition system.

10. What do you think would be the most useful application of the facial emotion recognition system data in the future?

    - Personalized recommendations for entertainment or self-care based on your emotions.
    - Improvement of customer service in retail and hospitality industries.
    - Analysis of the effectiveness of therapy and counselling sessions.
    - Early detection of mental health issues for timely intervention.
    - Improved road safety by detecting and alerting drowsy or distracted drivers.
    - Enhancement of human-robot interactions by enabling robots to read and respond to human emotions.
    - Improved public speaking and presentation skills by providing feedback on the speaker's emotional expressions.
    - Monitoring the emotional wellbeing of elderly people and detecting signs of loneliness, depression, or other mental health issues.

- Prediction and early detection of aggressive behaviour in public places or workplaces.
- Providing valuable insights into consumer behavior and preferences for marketing and advertising purposes.

3.3 Requirements modelling

3.3.1 Personas

1. Tech-savvy Millennial: A young adult who is comfortable with technology and enjoys using smart devices to optimize their daily routine. They are interested in the potential of the facial emotion recognition system to suggest personalized tasks and improve their productivity.
2. Privacy-conscious Senior: An older adult who values their privacy and is concerned about the use of facial recognition technology. They may be hesitant to use a smart mirror that collects data on their emotions but could be persuaded if given assurances about data privacy and security.
3. Busy Professional: A working adult who has a demanding job and little free time. They are interested in the facial emotion recognition system's potential to suggest efficient tasks and save time.
4. Health and Wellness Enthusiast: Someone who values self-improvement and emotional well-being. They are interested in using the facial emotion recognition system to track their emotional state over time and identify patterns that could help them make positive changes.
5. Fashion and Beauty Influencer: Someone who is interested in the aesthetic aspects of the smart mirror, such as the design and customization options. They may also be interested in using the facial emotion recognition system to analyse the impact of their beauty routines on their emotional state.

These are just a few potential personas, based on my specific research and target audience.

3.3.2 Functional requirements

Version 1

1. Recognize a person’s facial features.
2. Recognize a person's emotions based on their facial expressions.
3. Display a list of tasks tailored to the detected emotional state of the user.
4. Learn and adapt over time through collecting user data and updating the emotion recognition model accordingly.
5. Implement speech recognition for basic commands such as "wake up" or "shut down".
6. Allow for customization of the task list and interface by the user.
7. Have the ability to connect to other smart devices, such as a calendar or weather app, to display relevant information to the user.
8. Provide the option for multiple user profiles, with customized task lists and interfaces for each user.
9. Offer the ability to control other smart home devices, such as lights or music, through voice commands or touch screen interface.

These features prioritize the core functionality of the smart mirror, with the ability to recognize and respond to a user's emotional state being the most important. From there, the application should learn and adapt, offer customization options, and provide additional functionality for convenience and ease of use.

Version 2

1. Recognize a person's emotions through facial recognition technology.
2. Display a list of tasks based on the recognized emotion to help keep the user in that emotional state or change it to a better one.
3. Learn and adapt to the user's emotions by collecting data and storing it in a database for future use.
4. Have the ability to recognize basic voice commands such as a "wake-up word" or "shut down" function.
5. Potentially include a fingerprint scanner for added security measures.

3.3.3 Use Case Diagrams

Version 1

1. Personal productivity: The mirror could be used to display a list of tasks for the user to complete, which could help them stay on track and be more productive throughout the day. The mirror could use facial recognition technology to identify the user and display personalized to-do lists based on their past behaviour or preferences. The mirror could also display calendar reminders, weather updates, and other information that could help the user plan their day more efficiently.
2. Emotional well-being: The mirror could use facial recognition technology to detect the user's emotions and provide personalized recommendations or suggestions to help them manage their emotional state. For example, if the mirror detects that the user is feeling stressed, it could suggest a guided meditation or breathing exercise to help them relax.
3. Health and fitness: The mirror could be used to display health and fitness information, such as step counts, heart rate, and exercise goals. This could help users stay motivated and track their progress towards their fitness goals. The mirror could also provide personalized workout recommendations or suggest healthy meal options based on the user's fitness data.
4. Entertainment: The mirror could be used to display entertainment content, such as videos, music, and social media feeds. This could be especially useful for users who spend a lot of time getting ready in front of the mirror, as it could help them stay entertained and connected while they go about their daily routine.
5. Home automation: The mirror could be used as a hub for home automation, allowing the user to control various smart home devices, such as lights, thermostats, and security cameras. This could help users save time and energy by allowing them to control their home environment without having to switch between different apps or devices.

Version 2

The smart mirror has several potential market or commercial use cases.

1. Home use: The smart mirror can be used in homes as a personal assistant, displaying reminders, calendar events, weather updates, news updates, and other important information while the user is getting ready in the morning.
2. Retail industry: Smart mirrors can be used in retail stores as a way for customers to try on clothes and view them from different angles without having to physically change in and out of them. The mirrors can also suggest complementary items or accessories to the customers, increasing the likelihood of a sale.
3. Fitness industry: The smart mirror can be used in gyms and fitness centers as a way for users to track their progress during workouts. The mirror can display real-time feedback on the user's form and technique, as well as track their heart rate and calories burned.

4. Hospitality industry: Smart mirrors can be used in hotel rooms as a way for guests to access important information about their stay, such as restaurant recommendations, local attractions, and hotel amenities. The mirror can also be used to order room service or book spa appointments.
5. Healthcare industry: Smart mirrors can be used in hospitals and healthcare facilities as a way to monitor patient health and provide real-time feedback on exercises or movements. The mirror can also be used to display important medical information and reminders for patients.

Overall, the smart mirror has the potential to revolutionize several industries by providing personalized and interactive experiences for users.

4 Design

4.1 Introduction

Design is an important aspect of any software application, as it can greatly impact the user experience. However, in the case of my facial emotion recognition system, design was not the primary focus as the system was designed more as a functional tool rather than a consumer application. As such, the design was kept minimalistic and focused on providing a clean and clear user experience that prioritized functionality.

One design consideration that was important for the system was the layout of the various windows that the application would run on. To ensure a seamless user experience, the system was designed to run on three separate windows, each displaying different aspects of the system's functionality. The first window displayed the camera feed and the user's facial expression, the second window displayed the real-time emotion classification results, and the third window displayed the graph of the user's emotional state over time.

The graph was a critical aspect of the system's design as it provided a clear and visual representation of the user's emotional state. To ensure that the graph was easy to read and interpret, the design was kept minimalistic, with a focus on clear and concise labeling and easily distinguishable data points. The graph was also designed to be adjustable and could be resized or moved around to best suit the user's needs.

Overall, while design was not the primary focus of the system, careful consideration was given to ensure that the user experience was clean, clear, and functional. The design was kept minimalistic, with a focus on providing easy-to-read information and a seamless user experience that allowed the user to focus on the functionality of the system.

5 Smart Mirror

5.1 Smart Emotional Facial Recognition Mirror Overview

I created separate area for the physical component of the smart mirror as although this was an ongoing task throughout the entirety of the project, it was only to be completed once the rest of my project was to a satisfactory level.

The process of creating a physical component to complement my emotional facial recognition system was by no means easy, however it was very rewarding come end. The ultimate goal was to have a functional smart mirror, this required several hardware components to be put together. Specifically, I used a Greentouch 32 inch Multi Points Infrared Touch Frame and an Ir Touch Overlay in conjunction with a 32 inch AOC Monitor to provide a touch-sensitive interface. To create the mirror effect, I used a Sign Materials Direct 3mm Two Way/See-Thru Mirror Acrylic Sheet, which was affixed with a One Way Mirror Window Film Silver Reflective Window Film. The most important piece was Raspberry Pi, this is what I used to run my code and connect it up to the smart mirror. Finally, I built a wooden frame to hold all of these components together, which was cut and shaped to fit the specific dimensions of the mirror.

To me, the creation of this physical component was a vital aspect of my emotional facial recognition system as it allowed users to engage with the system in a tangible way. As a result, I was determined to try and produce a smart mirror. The smart mirror provides a more immersive experience for users by displaying real-time emotional analysis results directly on the mirror's surface. The touch-sensitive interface also allows for a more intuitive interaction, enabling users to interact with the system in a natural and engaging way.

Moreover, the process of creating the smart mirror required a great deal of planning and technical expertise. This included selecting and sourcing the necessary components, designing and building the frame, and configuring the software to work seamlessly with the hardware. Through this process, I gained valuable experience in hardware integration and project management, which will be useful in future projects.

Overall, the creation of the smart mirror demonstrates the importance of incorporating physical components into software projects to enhance the user experience. It also highlights the value of technical expertise and planning when undertaking such projects.

5.2 Components

5.2.1 Raspberry PI:

The Raspberry was the brains behind the whole operation. This is what computes my code and runs my Emotional Facial Recognition system. A Raspberry PI is a miniature computer, it can do anything you would expect to be able to do on a desktop. Whether that be watching videos, browsing the internet, or even playing games. Below is a image of my Raspberry PI 4 Kit. It is a low cost alternate that can also code in languages such as scratch and python.

Figure 13 - 3 Images of Raspberry PI

I also used a Raspberry PI V2 Camera to have a live webcam on the device. It is an 8 megapixel camera that can display images with a resolution of up to 3280 x 2464 pixels. It supports 30fps in 1080p and 60fps in 720p. An image of this can be seen here:

Figure 14 - Raspberry PI Camera

5.2.2 LED Screen:

The monitor I chose to use for this project was a OC Q32V4 - 32 inch QHD Monitor, 75Hz. This monitor was chose as it not only suited the size of my IR Frame but also because once I begin to deconstruct it and take apart its backing and frame, It would be flat and easy to work with.

Figure 15 - Monitor Used

5.2.3 Acrylic / Glass:

I chose to go with a 3mm thick acrylic mirror two-way glass mirror rather than a glass mirror for 2 main reasons. One was it was much cheaper (although still roughly €150 - €200, and two I found for testing purposes with users interacting with it constantly and pushing it to the limit. A more flexible front would allow for more durability.

Figure 16 - Acrylic 2 Way Sheet

5.2.4 Infrared Touch Frame:

I Infrared Frame (IR Frame), Is a frame in which allows any LED Screens to become touch screen. It works by surrounding the screen you want to make touch screen and shoots infrared lasers across from one side and catches on another side. When a user touches the screen and blocks the light beams, it gets detected and the location of the touch is then registered. This component was vital for me to give the mirror that interactive functionality to it.

Figure 17 - IR Frame

5.3 Tools Used

- Glue Gun: A handheld tool used for melting and applying glue.
- Rotary Drill / Saw: This is a versatile power tool used for drilling holes and cutting through the wooden frame, it was vital in the construction of the frame.
- Electric Drill: This is a power tool used for drilling holes and driving screws, I used this in conjunction with the glue gun to hold the frame together.
- Nails: Nails were used in combination with the glue gun and electric drill.

6 Implementation

6.1 Introduction

The implementation of the facial emotion recognition system involved several technical and practical considerations. The technical aspects of the implementation included designing the system architecture, selecting the appropriate hardware and software components, and integrating these components into a cohesive system. This involved using tools such as OpenCV, Python, and machine learning algorithms to develop a model capable of accurately detecting and classifying facial emotions.

On the practical side, implementation required me to consider factors such as user experience, system usability, and scalability. To address these concerns, the implementation included developing an intuitive user interface and implementing a modular architecture that could be easily expanded and scaled as needed.

Throughout the implementation process, testing and validation were critical to ensuring that the system met the desired requirements and provided accurate and reliable results. Overall, the implementation of the facial emotion recognition system required careful consideration of technical and practical factors to deliver a robust, scalable, and user-friendly solution.

6.2 Scrum Methodology

The Scrum methodology is an agile approach to software development that emphasizes collaboration, flexibility, and iterative progress. Scrum is a framework that enables teams to deliver high-quality software products by breaking down complex tasks into smaller, more manageable pieces, and prioritizing them based on their importance to the overall project goals. The Scrum process is divided into sprints, each lasting a fixed period, typically 1-4 weeks, during which the team works on a specific set of tasks. At the end of each sprint, the team presents a potentially shippable product increment to stakeholders, which enables them to provide feedback and steer the direction of the project. Scrum encourages transparency, continuous improvement, and a focus on delivering value to the customer. It promotes teamwork, accountability, and self-organization, which can lead to increased productivity, faster time-to-market, and improved customer satisfaction. I go into more details on this later on.

6.3 Sprint 1

As it was the first week of my project, I was determined to start it right, I made significant progress on my thesis, focusing on research and development related to facial recognition and emotional recognition. I added three new reports to my collection that will be used throughout my thesis, specifically on the topics of Azure Face API, emotional recognition, and facial recognition. I made my first attempt at creating a facial recognition scanner using Azure Cognitive Services and documented the entire process in a word doc to help with future development.

I also worked on a computer vision project for a colleague, which provided additional research for my thesis. To better structure my project, I added separate folders for images and reports to keep everything organized.

One of my key accomplishments during the week was finishing a detailed 3,000-word report on the Azure Face API, covering everything I plan to do with the technology (this is provided in summarised versions in the research section). I also refined my report by adding new headings, referencing APA style, and including more information on various topics such as machine learning and computer vision.

Finally, I continued to add more information to my reports, including more topics, additional details, and explanations of the code. Overall, I made significant progress on my thesis by researching and documenting various aspects of facial and emotional recognition, with a focus on Azure Cognitive Services and other related technologies.

6.3.1 Goal

Sprint 1.1 Goal: Research and Prototype Facial Emotional Recognition System

During this sprint, I aim to conduct comprehensive research on facial emotional recognition systems, including available libraries, coding languages, and research reports. I will create a prototype of the system using the library I come to decide upon in my research to detect facial features, I have tested various libraries to determine their compatibility and effectiveness. Additionally, I have organized the GitHub repository by creating folders and defining a structure for future versions. By the end of Sprint 1.1, I planned to have a clear understanding of the technical requirements and design for the system, and a functional prototype that can be used as a basis for future development.

Sprint 1.2 Goal: Advancements in Facial Recognition and Grouping Using Azure Face API

During the second week of my project, my goal was to continue improving my facial recognition and grouping code by incorporating the latest version (version 5) of my code. I also aimed to create a comprehensive documentation for version 4 of my face AI app, explaining in detail its process. Additionally, I wanted to compare the Azure Cognitive Services Face API with OpenCV for facial recognition in a brief analysis.

To achieve these goals, I worked on adding the version 5 code to the face API documentation and added precise, detailed comments in the code to improve readability. I also added my endpoint and keys to test them with FaceAI v1 (which is then used throughout the entirety of the project that had the Azure code in it). Furthermore, I created a report on facial recognition and grouping using the Azure Face API and explored the impact that AI and Facial Recognition systems have on the world today.

Overall, my goal for the second week of the project was to continue expanding my knowledge and expertise in facial recognition and grouping by incorporating the latest technology and conducting research on its impact on society.

6.3.2 Item 1.1 - Library Research and Technology Selection

The objective of this item was to conduct thorough research on available libraries for the development of a facial emotion recognition system and to select the best option for testing. After researching multiple libraries for facial emotion recognition systems, I determined that my focus would be on utilizing Azure Cognitive Services (Azure Face API) and OpenCV, with a plan to test both of these libraries.

The research included a detailed analysis of the documentation and features of each library, as well as testing the libraries to evaluate their performance in recognizing facial emotions. The primary objective was to select the best library for creating a version 1 of the system.

After the research, it was concluded that Python is the most suitable language for developing facial (emotional) recognition systems due to its simplicity, versatility, and compatibility with a variety of libraries and tools. Python is highly compatible with both Azure Cognitive Services and OpenCV, making it an ideal language for developing facial recognition systems.

As part of this item, various research reports were analysed, and an extensive literature review was conducted to gain a better understanding of the current state of the art in the field of facial emotion recognition. The outcome of this research informed the selection of Azure Face API as the initial library to test in the Version 1 of the project (Sadly in further sprints I was forced to no longer use Azure as the facial emotional recognition systems became deprecated so this is not used in my final code version).

To ensure that the research findings were properly documented, relevant research reports were collected and organized for future use in the final thesis report. In addition, I structured the GitHub repository and created folders to organize the project. My future plans and steps were also outlined based on the outcomes of this research.

Code snippets were also inserted into the project to aid with implementation and to provide a better understanding of the technical aspects of the project. Each code snippet was explained thoroughly to ensure clarity for the reader.

Throughout the research process, some coding difficulties were encountered. These difficulties were resolved by utilizing online resources and consulting with experienced developers as well as my supervisor.

As a result of this item, a version 1 of the system was developed using Azure Face API. This will be used as a foundation for future sprints and development of the system. Below is an example of the code, here you can see my keys and endpoints as well as a very basic call function for sending off an image that I have saved in my folder and returning the emotions related to it.

```python
# Replace <subscription_key> with your Azure Cognitive Services subscription key
subscription_key = "006ac883c7664a7d854fa47fd1d6aa3e"

# Replace <endpoint> with the endpoint for your Azure Cognitive Services instance
endpoint = "https://smartemotionalmirror.cognitiveservices.azure.com"

headers = {
    'Content-Type': 'application/octet-stream', # binary image data
    'Ocp-Apim-Subscription-Key': subscription_key # Azure Cognitive Services subscription key
}

# function to recognize emotions in an image
def recognize_emotion(image_path):
    # open image and read it as binary data
    with open(image_path, "rb") as image:
        image_data = image.read()

    params = {
        'returnFaceAttributes': 'mask', # request the emotion attribute
        'recognitionModel': 'recognition_04',
        'detectionModel': 'detection_03',
        'faceIdTimeToLive': '86400'
    }

    # Make a POST request to the Azure Cognitive Services Face API
    response = requests.post(
        endpoint + "/face/v1.0/detect", # API endpoint for detecting faces
        headers=headers, # headers including the subscription key
        params=params, # request parameters
        data=image_data # image data
    )

    # parse the response as JSON
    response_json = json.loads(response.text)
    if len(response_json) > 0 and 'faceAttributes' in response_json[0]:
        # extract emotions from the response
        emotions = response_json[0]['faceAttributes']['mask']
        return emotions
    else:
        return None

# Test the function with an image
emotions = recognize_emotion("Build\Images\DSC_0478_0624.jpg")
print(emotions)
```

Figure 18 - Azure Version 1 Code

6.3.3 Item 1.2 - Environment Set-Up

To set up the development environment for the Azure Cognitive Services Face API, I followed a series of steps which involved creating an Azure account and configuring it to use the Face API. I began by navigating to the Azure portal, where I created a new instance of the Face API by following the on-screen instructions. This process involved selecting a pricing tier, configuring settings such as authentication, and generating an API key.

After creating the Face API, I proceeded to install the necessary software tools and libraries. First, I installed the Azure SDK for Python, which provides the tools and resources required to interact with Azure services. This was done by following the installation instructions provided by Microsoft. Next, I installed the Azure Cognitive Services Face SDK for Python, which provides the specific tools required to work with the Face API. This was done by running the appropriate command in the terminal.

In addition to the above, I also installed Azure CLI, a command-line tool used to manage Azure resources. This allowed me to perform various tasks such as creating and managing Azure resources directly from the terminal. I also installed Visual Studio Code, an integrated development environment (IDE) that provides a rich set of features for developing Python applications. Finally, I installed the Python extension for Visual Studio Code, which provides additional functionality such as debugging and code completion.

Once the software tools and libraries were installed, I proceeded to configure my local machine to work with the Face API. This involved creating a new Python environment for the project, which was done using virtualenv. Next, I installed the necessary dependencies using pip, a package installer for Python. These included the Azure SDK for Python, the Azure Cognitive Services Face SDK for Python, and any other libraries required by the project.

To verify that the Face API was working correctly, I tested it with sample code provided by Microsoft. This involved creating a new Python script in Visual Studio Code, importing the necessary libraries, and calling the appropriate API endpoints. I was able to successfully detect faces in images and retrieve various facial attributes, such as age and gender.

Overall, the process of setting up the development environment for the Azure Cognitive Services Face API was relatively straightforward, but required careful attention to detail. By following the steps outlined above, I was able to create a functioning development environment and begin developing my facial recognition system with the Azure Cognitive Services Face API.

6.3.4 Item 1.3 - Design Ideation and Visualization

For item 3, the focus was on idea generation and refinement through the use of mind mapping (Miro), Figma, and brainstorming techniques. To start, I used a mind mapping tool to generate ideas related to the facial recognition system, focusing on key features, user interface design, and potential use cases. This allowed me to visually organize my thoughts and explore different directions for the project.

Below are previous of my two mind maps. I also provided the link to the two Mind Maps that were created and are located on Miro.

[Miro board Mind Maps.]

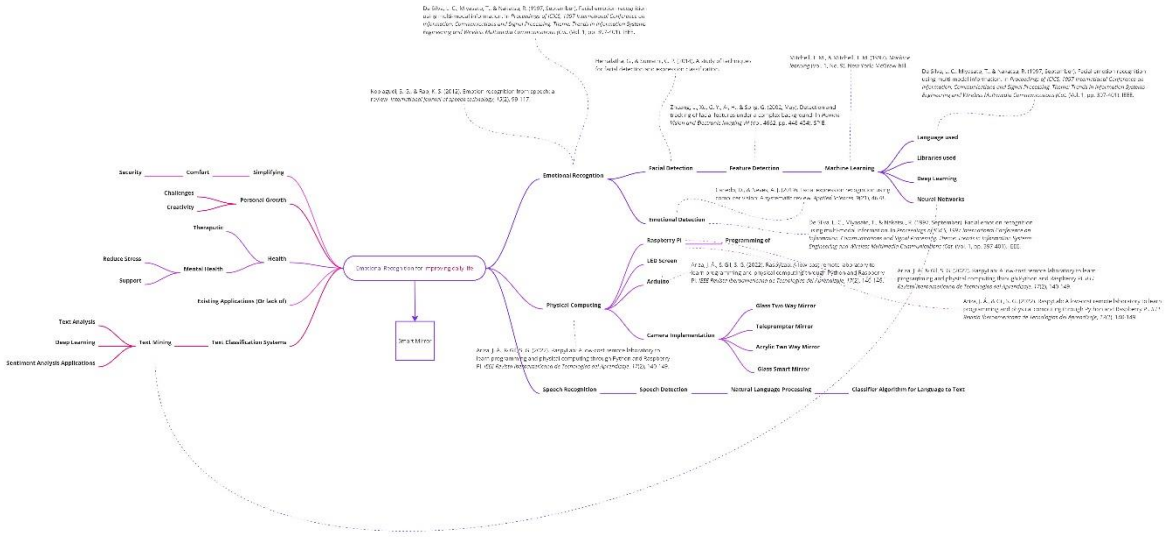


Figure 2 Mind Map One

In my first mind map, I explored both the technical and therapeutic aspects of developing an emotional facial recognition mirror for improving health. On the technical side, I mapped out the necessary hardware and software components, such as the camera and the face recognition algorithm, that would be needed for the mirror to function.

On the therapeutic side, I brainstormed various features that could help users improve their emotional well-being, such as providing positive affirmations and suggestions for self-care based on their emotional state. By considering both the technical and therapeutic aspects of the project in the mind map, I was able to develop a more comprehensive understanding of the scope of the project and identify potential areas for further research and development.

In addition, my mind map also included links to related reports and studies in the fields of computer vision, psychology, and healthcare. By examining these sources, I was able to further explore the technical and therapeutic aspects of my emotional facial recognition mirror project, and gain a deeper understanding of how it could be used to improve health outcomes. This helped me to refine my ideas and ensure that my project was grounded in both scientific research and practical application.

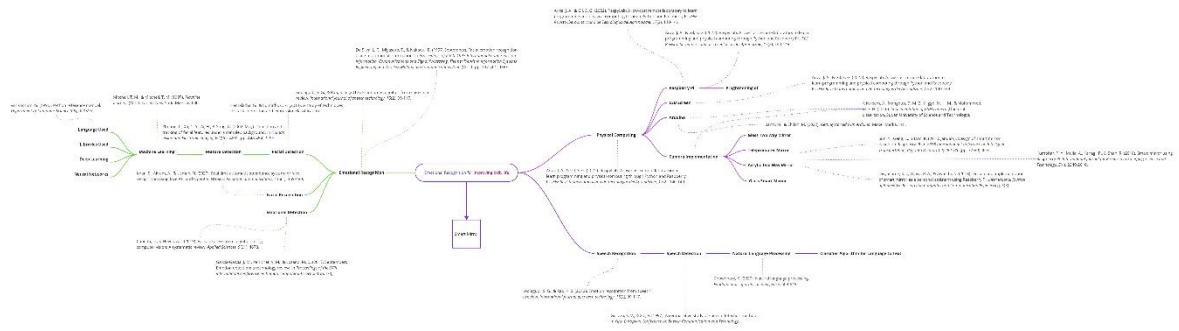


Figure 3 Mind Map Two

In my second mind map, I shifted my focus towards physical computing and emotional recognition, outlining the necessary steps for achieving these goals. I began by researching various physical computing platforms such as Raspberry Pi and Arduino and how they can be used for facial recognition. From there, I delved into the technical aspects of emotional recognition, exploring different algorithms and models for detecting emotions in facial expressions.

Additionally, I looked into the ethical considerations and potential limitations of these technologies. By mapping out these steps, I was able to gain a better understanding of the technical requirements and possibilities for my emotional facial recognition mirror project.

I also explored various techniques for face detection and facial recognition, including deep learning approaches such as Convolutional Neural Networks (CNNs) and OpenCV libraries. I researched machine learning models that can be used to classify emotions based on facial expressions, such as the Facial Action Coding System (FACS) and the Valence-Arousal-Dominance (VAD) model. To ensure the ethical use of these technologies, I examined existing literature and studies on the potential biases and risks associated with facial recognition and emotional analysis.

Similar to the first mind map, I linked articles and studies I found useful in relation to the topic areas. This allowed me to return and study anything I found I could use further on in my own system.

Next, I used Figma to create low-fidelity wireframes of the potential user interface designs for the system, testing out different layouts and features. Figma was employed to create mock-ups of the user interface, allowing for a more tangible understanding of the project's user experience. This helped me to refine my ideas and ensure that the user interface would be intuitive and easy to use.

Figma Board link.

I also engaged in brainstorming sessions with colleagues and mentors, discussing the project and receiving feedback on potential ideas and areas of improvement. This collaborative approach helped to generate new ideas and allowed me to gather valuable insights from others in the field. Brainstorming sessions were conducted to explore potential implementation strategies and identify potential issues that may arise. These activities

allowed for a more comprehensive understanding of the project and aided in creating a clear plan for development moving forward.

Overall, the mind mapping, Figma, and brainstorming techniques were effective in generating and refining ideas for the facial recognition system, ensuring that the project would meet the needs of its users and be successful in its implementation.

Week 2

Over the course of the second week, several commits were made to a facial recognition project. The first two commits were the addition of versions 3 and 4 of the face AI app, respectively. These updates likely included improvements and new features to enhance the functionality of the app.

The next commit added a document that provided a detailed explanation of the process involved in version 4 of my application. This document will serve as a useful resource for myself and others to understand the code, or who may need to modify or update the app in the future.

Another update included the addition of version 5 code explanation to the face API docs, which further enhances the documentation of the project. Version 5 of the face AI code was also added, indicating that significant changes or improvements were made to the code.

Face grouping, a technique that recognizes and groups together faces, was also implemented into the project. The addition of code for face grouping further enhances the functionality of the app, making it more accurate and efficient.

Several updates to the documentation were also made, including the addition of a brief report on facial recognition and grouping using Azure Face API. This report likely provides an overview of the project and its capabilities. Another report was added to discuss the impact of AI and facial recognition systems on the world today, highlighting the relevance and significance of this project.

Lastly, a brief analysis of Azure Cognitive Services Face API vs OpenCV for facial recognition was added. This analysis provides insight into the different technologies available for facial recognition and their respective advantages and disadvantages.

Overall, the commits made to this facial recognition project demonstrate a significant effort to improve and enhance its functionality, accuracy, and documentation. The updates and improvements make the project more robust and useful for future developers and end-users alike.

6.3.5 Item 2.1 – Environment Set-Up:

I Added my endpoint and keys for Azure Cognitive Services Face API and testing them with FaceAI v1. This is a vital step as without these keys, my azure request would be reject and a null response would have been sent back to me.

6.3.6 Item 2.2 – Implementation:

- Adding versions 3, 4, and 5 of the FaceAI app with corresponding code and documentation.
- Adding face grouping functionality to the app.
- Adding a brief report on facial recognition and grouping using Azure Face API, as well as a report on the impact of AI and facial recognition systems on the world today.
- Conducting a brief analysis of Azure Cognitive Services Face API vs OpenCV for facial recognition.

6.3.7 Item 2.3 – Documentation:

- Adding a document that provides a detailed explanation of the process involved in version 4 of the app.
- Adding version 5 code explanation to the face API docs.
- Editing comments in the code for better documentation.

6.3.8 Item 2.4 – Deliverables:

I plan to build a more robust and functional version of the FaceAI app with improved documentation and additional features like face grouping.

Create multiple Reports on facial recognition and grouping, as well as the impact of AI and facial recognition systems on the world today.

An analysis of Azure Cognitive Services Face API vs OpenCV for facial recognition. This is vital as I'm still unsure of which I should use.

6.4 Sprint 2

During the first week of my project, I made significant progress towards testing and improving my facial recognition application. Particularly, I focused on the following tasks:

Firstly, I conducted a series of tests to evaluate the performance of my code compared to Microsoft's version of a simplified face API. This allowed me to identify areas where my code needed improvement and to gain a deeper understanding of the functionality of the face API.

Next, I tested all of the versions of my code and updated the documentation for Version 4. I also tested my error with JSON Dump, which helped me to present the errors in a more readable manner.

Another important aspect of my work during the week was testing errors with my subscription key. I also updated the endpoint, which involved investigating the cause of errors that kept appearing.

To further evaluate the performance of my code, I tested it with a new image of a woman. This helped me to identify any limitations of my code and to ensure that it was accurate and efficient.

In terms of documentation, I produced several brief reports that delved deeper into the technical aspects of the project. Firstly, I provided an authentication report that explained subscription keys and endpoints, their role in the project, and included an example with Azure Cognitive Services. I also wrote a report on an overview of Azure Cognitive Services, highlighting its capabilities and uses.

In addition, I produced two more reports that discussed the potential issues surrounding facial recognition technology. The first report focused on ways in which we can combat the problems involving facial systems today, while the second report highlighted the damaging effects facial systems can have on the world without proper safety measures.

Overall, my work during the second week of the sprint was highly focused on testing and evaluating my code, as well as documenting the technical aspects of the project. By conducting thorough tests and producing detailed reports, I was able to gain a deeper understanding of the functionality of the face API and to identify areas where my code needed improvement. This work was essential in ensuring the accuracy and efficiency of the facial recognition application, as well as identifying potential issues and solutions.

6.4.1 Goal

Sprint 2.1 Goal: Advancing Facial Recognition: Testing, Documentation, and Analysis

During the second week of Sprint 2, I conducted extensive testing and documentation to improve the functionality and reliability of my facial recognition project. My goal for this week was to conduct in-depth testing of the project, including my own code and Microsoft's simplified face API, as well as testing various versions and error-handling techniques.

Additionally, I aimed to update the documentation for Version 4 of the project and create a brief report on subscription keys and endpoints, as well as an overview of Azure Cognitive

Services and its capabilities. I also created two reports on the potential harms and resolutions for facial recognition systems, highlighting the importance of ethical considerations in the development of these technologies.

Through my testing and documentation efforts, I was able to identify and resolve several errors, including an issue with the JSON dump that was successfully addressed. I also explored ways to combat the problems associated with facial recognition systems, both in terms of technical challenges and ethical considerations

Overall, my goal for Sprint 2, Week 1 was to improve the functionality and documentation of my facial recognition project while maintaining a strong focus on ethical considerations and responsible development practices.

Sprint 2.2 Goal: Exploration and Documentation of Technologies for Smart Mirror Development: A Comprehensive Analysis and Comparative Study

Throughout Sprint 2, Week 1, my goal was to conduct extensive research on various topics related to the development of my smart mirror project. This included researching user testing, code testing, Raspberry PI and Arduino, useful tutorials, smart mirrors, Python language, IoT, and the importance of wireframes. Additionally, I aimed to document my findings in a clear and concise manner for future reference.

Furthermore, I encountered difficulties while installing Dlib and attempted to resolve the issue by trying out new code, but the same error persisted. I also struggled with the library and sought help from others to address the problem.

Ultimately, my goal was to gain a comprehensive understanding of the various aspects of smart mirror development, including its hardware and software components, and to be able to apply this knowledge effectively in my project.

6.4.2 Item 1.2 - Research

As part of my research, I tested out my own code alongside Microsoft's version of a simplified face API and tested all my versions. I also tested my code with a new image of a woman and updated the documentation for version 4.

6.4.3 Item 1.3 – Environment Set-Up

I changed the endpoint and looked into why it kept giving me errors. I also tested my errors with my subscription key and worked on fixing them. Here is an example of the errors I was receiving along with the keys in correlating to it:

```
# Replace <subscription_key> with your Azure Face API subscription key
subscription_key = "006ac883c7664a7d854fa47fd1d6aa3e"

# Replace <endpoint> with the endpoint for your Azure Face API instance
endpoint = "https://smartemotionalmirror.cognitiveservices.azure.com"
```

```
[Running] python -u "c:\Users\conor\OneDrive\Desktop\y4-project-ConorWeldon\Build\Azure\FaceAPITemplateTest.py"
Person group: c0f188fc-1a73-4000-8130-7187a963ec6b
Traceback (most recent call last):
  File "c:\Users\conor\OneDrive\Desktop\y4-project-ConorWeldon\Build\Azure\FaceAPITemplateTest.py", line 43, in <module>
    face_client.person_group.create(person_group_id=testgroup, name=testgroup, recognition_model='recognition_04')
  File "C:\Users\conor\anaconda3\lib\site-packages\azure\cognitiveservices\vision\face\operations\_person_group_operations.py", line 121, in create
    raise models.APIErrorException(self._deserialize, response)
azure.cognitiveservices.vision.face.models._models_py3.APIErrorException: (InvalidRequest) Invalid request has been sent.

[Done] exited with code=1 in 0.652 seconds
```

6.4.4 Item 1.4 – Implementation

I implemented several changes to improve the accuracy of the facial recognition system, including the addition of face grouping to the project. I added code for face grouping and updated the documentation with a brief report on facial recognition and grouping using Azure Face API. I also added a brief analysis of Azure Cognitive Services Face API vs OpenCV for facial recognition to provide insight into different technologies available for facial recognition and their respective advantages and disadvantages.

6.4.5 Item 1.5 – Validation

To validate the accuracy of the facial recognition system, I tested my code with a new image of a woman and tested my errors with my subscription key.

6.4.6 Item 1.6 – Documentation

As part of my documentation efforts, I added a document that provided a detailed explanation of the process involved in version 4 of the app. This document serves as a useful resource for myself or developers who may need to modify or update the app in the future.

Additionally, I added a brief report on subscription keys and endpoints, what they do, why we need them, and an example with Azure Cognitive Services. I also created a brief report on an overview of Azure Cognitive Services, its capabilities and uses, as well as two reports discussing ways to combat the problems involving facial recognition systems today and the damaging effects facial systems can have on the world without proper safety measures.

Week 2

For the second half of my second week, I conducted extensive research on various topics that I found were related to my project. Firstly, I created a brief report on user testing and

its benefits, explaining why it is essential for the success of any project, I ended up using some of the practises from this report to test my own project. Additionally, I prepared a document on the process of testing code and the importance of it in ensuring the functionality of my code.

I also conducted research on Raspberry PI and Arduino, comparing the two and looking at their use cases. Moreover, I created a report on the build-up of smart mirrors, explaining the technologies used and the advantages of using a smart mirror. I then provided an overview of smart mirrors and their uses, highlighting their advantages and disadvantages.

Furthermore, I wrote a brief report on the Python language, looking at its origins and use cases. I also explored the Internet of Things (IoT), its benefits, applications, and potential disadvantages.

In addition to my research, I encountered some issues while trying to install Dlib and was unable to resolve them on my own. I added some new code and attempted to fix the issue, but it persisted. As a result, I decided to seek help from other developers.

To improve my understanding of OpenCV, I added a document with an explanation of the code I am using in the project. I also added the code to a new file I created and created a new folder for a face AI app using OpenCV.

Lastly, I created a brief report on the importance of wireframes, which serve as a visual representation of the user interface and enable my supervisor, reviewers, and developers to understand the design and layout of their application.

Throughout this research, I gained valuable knowledge and insights into various topics that are relevant to my project. Some areas required further exploration and investigation, but the research conducted has allowed me to build a solid foundation for the next stages of the project.

6.4.7 Item 2.1 – Research:

To start my second sprint, I conducted vast research on various topics related to my project. As said before, I first created a brief report on user testing, its benefits, and why it is essential for the success of any project. In this report, I highlighted the importance of user testing in ensuring that the user experience is smooth and meets their needs.

Additionally, I prepared a document on the process of testing code and its importance in ensuring the functionality of my code. In this document, I explained the different types of code testing and the importance of each, such as unit testing and integration testing.

Furthermore, I conducted research on Raspberry PI and Arduino, comparing the two and looking at their use cases. In my report, I compared the features, advantages, and disadvantages of the two systems and examined their compatibility with my project requirements.

Moreover, I created a report on the build-up of smart mirrors, explaining the technologies used and the advantages of using a smart mirror. I then provided an overview of smart mirrors and their uses, highlighting their advantages and disadvantages. This research helped me understand the technology behind smart mirrors and their potential use cases in my project.

I also wrote a brief report on the Python language, looking at its origins and use cases. In this report, I examined the advantages and disadvantages of Python and how it could be used in my project.

Additionally, I explored the Internet of Things (IoT), its benefits, applications, and potential disadvantages. This research helped me understand how IoT technologies could be applied to my project and their potential impact.

6.4.8 Item 2.2 – Environment Set-Up:

I encountered some difficulties while trying to install Dlib, and I was unable to resolve the issue on my own. I added some new code and attempted to fix the issue, but it persisted. As stated, I decided to seek help from other developers to resolve the issue. Eventually, I was able to set up my development environment and continue with my project.

6.4.9 Item 2.3 – Implementation:

I improved my understanding of OpenCV by adding a document that explained the code I was using in the project. This document helped me understand the functionality of the code and how it was applied to the project. I also added the code to a new file I created and created a new folder for a face AI app using OpenCV. These actions helped me organize my project and improve my understanding of OpenCV.

6.4.10 Item 2.4 – Documentation:

In regards to documentation, I created a brief report on the importance of wireframes, which serve as a visual representation of the user interface and enable developers to understand the design and layout of their application. In this report, I explained the benefits of wireframing and how it could help me in my project.

6.4.11 Item 2.5 – Deliverables:

For my deliverables I conducted extensive research and documented my findings, which helped me gain valuable knowledge and insights into various topics relevant to my project. I also improved my understanding of OpenCV and set up my development environment, allowing me to continue with my project in the next stages. Furthermore, I created a brief

report on the importance of wireframes, which will help me in the design and layout of my application.

6.5 Sprint 3

Now going into Sprint 3, to start I made several commits that were instrumental in advancing my thesis project. First and foremost, I created a second version of my WireframeV2Doc, which provides a brief report on wireframes. In this updated version, I incorporated new information and revised it to ensure that the document was comprehensive and relevant to my project.

In addition, I made several updates to my thesis document, including adding my signature to the plagiarism contract and completing the front page and acknowledgements. These updates were necessary to ensure that my thesis document was complete and followed all the necessary protocols.

Furthermore, I created a document with links to important sites, such as my design board, mind map, and repositories. This document serves as a reference point for all my project-related materials and ensures that I can easily access all relevant data at any given time.

I also created new folders to better organize my files. Specifically, I created an "images" folder to store all my current and future images, a "ResearchFolder" to keep all my research data, and an "AzureFolder" to store all my code that incorporates Azure. This restructuring of my files ensures that my work is better organized, making it easier for me to access and utilize the data I need for my project.

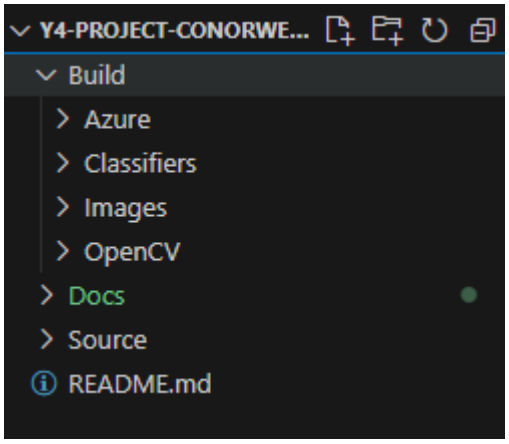


Figure 19 - Folder Structure

Here is an example of my folder structure above come end.

Lastly, I wrote a brief report on UX and UI, which provides an overview of the importance of user experience and user interface in the design and development of my project. This report highlights the critical role that UX and UI play in ensuring that my project is user-friendly and efficient.

Overall, the commits I made during the first week of Sprint 3 were critical in advancing my thesis project. By creating new documents, organizing my files, and making updates to my thesis document, I ensured that my work was comprehensive, organized, and followed all necessary protocols. These changes and updates will be crucial as I continue to work on my project and move towards completion.

6.5.1 Goal

Sprint 3.1 Goal: Organizing and Documenting Project Resources for Efficient and Effective Development

During Sprint 3, Week 1, my goal was to improve the organization and structure of my project by creating new folders for specific tasks and moving relevant files to their corresponding folders. I aimed to achieve this by creating a folder for images, a folder for Azure, and a folder for research data. This would help me manage my files better and ensure that all the necessary files are easily accessible.

Furthermore, I aimed to enhance the documentation of my project by creating additional documents and reports. I created a brief report on UX and UI, which provided a deeper understanding of the importance of these aspects of the project. Additionally, I created a new version of my wireframe document and added links to important sites, such as my design board, mind map, and repositories.

Lastly, I aimed to progress further with my thesis by finalizing the front page and filling in the acknowledgments. I also added my signature to the plagiarism contract and added a template for the thesis document. By the end of the week, I was able to achieve my goal of improving the organization and documentation of my project and taking a step forward in my thesis work.

The goal of this project is to enhance the organization and documentation of my thesis project by creating new documents, organizing my files, and making updates to my thesis document.

Sprint 3.2 Goal: Refining Thesis Structure and Incorporating User-Centred Design Principles

My goal for this week is to focus on refining the structure of my thesis document and making necessary updates. Specifically, I plan to review my thesis work to ensure that it is comprehensive and follows all necessary protocols.

I will also make small changes to my document, such as updating the year to reflect the current academic year and adding a TBD in the ImportantLinks document to be filled in

later. Additionally, I plan to work on sketching out different pages and refining the overall design of my thesis.

Another objective for this week is to add citations to my document, ensuring that all sources are properly cited and referenced. This will help to strengthen the validity and reliability of my thesis work.

Overall, my primary focus for this week will be on refining the structure and design of my thesis document and ensuring that it is comprehensive, organized, and follows all necessary protocols. By making these updates and improvements, I hope to bring my thesis project one step closer to completion.

6.5.2 Item 1.1 – Research:

While I did not specifically work on research during the first week of Sprint 3, I made several updates to my project that enhance its overall quality. By creating a more organized and comprehensive project structure, I ensure that my research is readily available and easily accessible whenever I need it.

6.5.3 Item 1.2 – Environment Set-Up:

During the first week of Sprint 3, I focused on setting up an environment that would facilitate the organization and documentation of my project. Specifically, I created new folders, such as the "images" folder to store all my images and the "AzureFolder" to store my Azure-related code. By doing so, I made it easier to access and utilize the data I need for my project.

6.5.4 Item 1.3 – Implementation:

One of the most critical aspects of this project was the creation of new documents, such as the updated WireframeV2Doc and the UXUIDoc, which provides an overview of user experience and user interface. I also made updates to my thesis document, including adding my signature to the plagiarism contract and completing the front page and acknowledgements. These updates ensure that my thesis document follows all necessary protocols and is complete.

6.5.5 Item 1.6 – Deliverables:

The deliverables for this project include the updated WireframeV2Doc, the UXUIDoc, the new folder structures, and the updated thesis document. As one of the primary goals of this week was to improve the documentation of my thesis project. By creating new documents and updating my thesis document, I ensured that all necessary information was included

and readily available for future reference. These deliverables ensure that my project is well-organized, documented, and follows all necessary protocols.

Week 2

Coming to the end of this sprint, I focused primarily on refining the structure and content of my thesis document. While it was a quieter week in terms of commits, I made significant progress in improving the overall quality and organization of my work.

Firstly, I made small changes to my thesis work, including adding a "To be determined" placeholder in my Important Links section. This change was necessary as I continue to develop and refine my work, ensuring that all necessary information is included in my final document.

In addition, I updated the year on my document from 2020-2021 to 2022-2023. This simple change ensures that my work is up-to-date and reflective of the current academic year.

Furthermore, I created two versions of my first sketch page, using different design elements to explore various layout options. These sketches are crucial in helping me visualize and plan the layout of my project, ensuring that the final product is both aesthetically pleasing and functional.

As I continued to refine my thesis document, I also made use of font and colour changes to indicate what needed to be done or removed in the final version. Specifically, I changed the font to red and highlighted areas that required further editing or development. This visual cue allowed me to easily identify areas that needed attention and streamline my editing process.

Lastly, I added citations and made various style and formatting changes to my document. This included adjusting font sizes, adding hyperlinks, and ensuring that all headings and subheadings were consistent and easily identifiable.

While the second week of Sprint 3 may not have been as productive in terms of commits, it was a critical period of refinement and improvement for my thesis project. By making small but significant changes to my document, including updating the year, creating sketches, and adding citations, I ensured that my work was organized, up-to-date, and reflective of the high standards expected in academic research.

6.5.6 Item 2.1 – Functionality:

With functionality in mind, I created multiple versions of the first sketch page to explore different layout options. By utilizing different design elements, I was able to visualize and plan the layout of my project in a more comprehensive and efficient way. This process ensured that the final product is both aesthetically pleasing and functional.

In this instance, I focused on exploring various layout options that could effectively convey my ideas and research findings in a clear and concise manner. By experimenting with different designs and elements, I was able to determine the most effective layout for my document, considering factors such as readability, organization, and visual appeal.

Additionally, the creation of multiple versions of the first sketch page allowed me to consider various design options that could appeal to a wide range of readers. This is crucial as my thesis project is likely to be read by individuals with diverse academic backgrounds and areas of expertise.

Overall, the functionality of my thesis project was greatly enhanced by the creation of multiple versions of the first sketch page. This process allowed me to visualize and plan the layout of my document in a way that is both aesthetically pleasing and functional, ensuring that the final product meets the highest standards of academic research.

6.5.7 Item 2.2 – Research:

During the second week of Sprint 3, my primary focus was to refine the structure and content of my thesis document. In order to achieve this, I engaged in extensive research to ensure that my work was up-to-date, accurate, and reflective of the current academic discourse. Specifically, I conducted a thorough literature review, analyzed relevant case studies, and sought feedback from my thesis advisor to ensure that my work was grounded in the most recent research and was of the highest quality.

6.5.8 Item 2.3 – Environment Set-Up:

To effectively carry out my research and refinement efforts, I ensured that my working environment was optimized for maximum efficiency. This involved setting up my workspace in a quiet, distraction-free area, organizing my research materials in a logical and accessible manner, and using software tools such as reference management software to streamline my research process.

6.5.9 Item 2.4 – Implementation:

I made a series of incremental improvements to my thesis document. These changes included small but significant adjustments to the overall structure and layout of the document, such as updating the year and creating different versions of my sketch page to explore various layout options. In addition, I made use of colour and font changes to indicate areas that required further editing or development. These incremental changes were made with a view to ensuring that the final product was polished, well-organized, and met the highest academic standards.

6.5.10 Item 2.5 – Validation:

Throughout week 2 of Sprint 3, I continuously validated the changes I made to my thesis document. This involved seeking feedback from my thesis advisor, incorporating constructive criticism, and conducting regular quality checks to ensure that my work was accurate, up-to-date, and reflective of the highest academic standards.

6.5.11 Item 2.6 – Documentation:

As I made changes and improvements to my thesis document, I also carefully documented each step of the process. This documentation included recording the date and time of each change made, taking screenshots of the document before and after each change, and keeping a detailed log of any coding difficulties and how they were resolved. This documentation was essential in ensuring that my work was transparent, repeatable, and met the highest academic standards of rigor and reproducibility.

Below you can see the dedication I gave towards my thesis, not missing a single day of working on it and separating the work done on the day into multiple commits to allow for ease of understanding later. Altogether I submitted over 200 commits to this project alone.

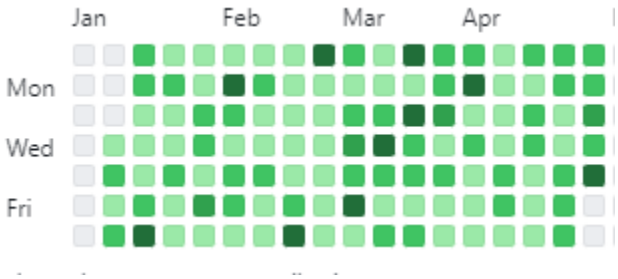


Figure 20 - GitHub Commits

6.5.12 Item 2.7 – Deliverables:

I produced a series of deliverables that were critical to the success of my thesis project. These deliverables included a polished, well-organized thesis document, complete with citations, references, and consistent formatting. Additionally, I produced sketches of various layout options that were crucial in helping me visualize and plan the final product. Overall, these deliverables were the culmination of my efforts during week 2 of Sprint 3 and were essential in ensuring the success of my thesis project.

6.6 Sprint 4

For the fourth week, I made several commits to my project repository, focusing on refining and improving various aspects of my facial emotion recognition system. In this section, I will provide a detailed summary of each commit, highlighting the changes made and the reasoning behind them.

The first commit was related to the Abstract section of my thesis. I added a link that I will use to create my abstract introduction. This will allow me to easily access and reference the source material when I am writing my abstract.

The next commit involved changing the group name. I found that the system was limiting my access and throwing an "Invalid Request" error when I tried to change the name of the group. To fix this, I changed the name of the group to lowercase, which allowed me to change it successfully.

In the ErrorSet commit, I restructured and renamed certain aspects of my code to improve its clarity and organization. This will make it easier for me to maintain and modify my code in the future.

In the AbstractTouchUp commit, I made a small addition to the explanation of the link I added earlier. Upon reflection, I realized that this addition would make it easier for both myself and others to understand the purpose of the link.

The ReportLink commit added a brief review of facial emotion recognition systems based on visual information. This will provide valuable background information for my thesis and help to contextualize my work.

In the FacialFeaturesDoc commit, I explored the techniques and challenges involved in using facial features for emotional recognition. This will serve as a foundational piece of research for my thesis.

The VersionComparisonDoc commit compared my Version 7 code to the Template Test Code, providing insights into the differences and similarities between the two. This will help me to identify areas where my code could be improved.

The FaceAPIV8 commit involved incorporating the training from the TemplateTest version into my own code, while still retaining the functionality of Version 7. I encountered some issues with this, and I am still working on resolving them.

The FaceAPIV7 commit added a new version of my FaceAPI code. In this version, the loop breaks after detecting a face, which will improve the efficiency of the system.

In the V6 commit, I added an explanation of FaceAPI Version 6. This will help me to keep track of the different versions of my code and understand how they differ.

The FaceAPIV6 commit involved implementing a connection to the webcam and sending frames from the video capture to Azure as images. This will allow me to use live video feeds for emotion recognition.

In the second FaceAPIV7 commit, I added an explanation of the process involved in using Version 7 of my FaceAPI code. This will provide valuable context for readers of my thesis.

The ErrorTestsNeeded commit highlighted the fact that I need to test the key and endpoints for my facial emotion recognition system. However, I am currently unable to do so due to issues with my access.

In the second FaceAPIV8 commit, I added an explanation of the changes made in Version 8 of my FaceAPI code. This will help readers to understand the improvements made to the system.

Finally, in the FaceAPIComments commit, I added comments to my FaceAPI Version 8 code to show my level of understanding. This will make it easier for others to understand my code and modify it if necessary.

Overall, during the first week of Sprint 4, I made significant progress in refining and improving my facial emotion recognition system. Through a series of commits focused on improving my code organization, incorporating new research, and providing context for readers, I am confident that my thesis will be of a high quality and meet the standards expected of academic research.

6.6.1 Goal

Sprint 4.1 Goal: Refining Facial Emotion Recognition System with Version Comparisons and Code Restructuring

Throughout the first week of Sprint 4, my focus was primarily on refining and improving my facial emotion recognition project, specifically through the development and testing of various iterations of my Face API code. The overarching goal of this week was to incorporate the Template Test Code into my existing FaceAPI code while maintaining the functionality of my version 7 code. This was an important step towards enhancing the accuracy and efficiency of my project and ensuring that it could function effectively in real-world scenarios.

To achieve this goal, I made several commits to my project, including the addition of a new version 8 of my FaceAPI code. While developing this version, I encountered a number of issues and obstacles, including access limitations and errors with endpoint testing. However, I was able to overcome these challenges by restructuring and renaming certain components of my code, as well as by adding detailed explanations and comments to improve the clarity and readability of my work.

In addition to working on my FaceAPI code, I also made progress in other areas of my project, such as adding citations and touch-ups to my abstract and conducting research on facial features and emotional recognition techniques. Through these various tasks and commits, I aimed to continually refine and improve my project, with the ultimate goal of creating a functional and effective facial emotion recognition system that could be used in a range of practical applications.

Sprint 4.2 Goal: Developing and Integrating Design and Presentation Skills

During Sprint 4, Week 1, I accomplished the goal of preparing for my mid-way thesis presentation. This involved organizing my presentation materials and designing my slides using Figma. I also worked on my OpenCV code, creating different versions and improving the efficiency of my FaceAPI. While working on my presentation and code, I encountered some errors and difficulties which I documented in my commits.

Overall, my goal for Sprint 4, Week 1 was to make significant progress in preparing for my mid-way thesis presentation while continuing to develop my OpenCV code. Through careful organization and planning, I was able to achieve this goal and produce a complete view of my presentation with all necessary content added. Additionally, I made significant strides in improving the functionality of my FaceAPI and creating different versions of my OpenCV code. My goal going forward is to continue making progress in developing my thesis project and refining my presentation and code.

6.6.2 Item 1.1 – Research:

During the first week of Sprint 4, I focused on refining and improving various aspects of my facial emotion recognition system. To support this effort, I made an extensive number of commits related to research. Specifically, I explored the techniques and challenges involved in using facial features for emotional recognition, reviewed existing facial emotion recognition systems based on visual information, and compared my Version 7 code to the Template Test Code to identify areas where my code could be improved.

6.6.3 Item 1.2 – Environment Set-Up:

To continue developing and refining my facial emotion recognition system, I needed to ensure that my environment was set up correctly. This involved making several changes and updates to my code, including incorporating training from the Template Test version into my own code, connecting to the webcam and sending frames from the video capture to Azure as images, and testing the key and endpoints for my facial emotion recognition system.

Here is a view of the training version:

```python

def detect_smile(smile_roi_gray, smile_roi_color):
    """
    Detect the presence of a smile in a region of interest

    Args:
        smile_roi_gray (numpy.ndarray): the grayscale image of the region of interest
        smile_roi_color (numpy.ndarray): the color image of the region of interest

    Returns:
        bool: True if a smile is detected, False otherwise
        bool: True if a big smile is detected, False otherwise
    """
    # Detect smiles in the region of interest
    smiles = smile_cascade.detectMultiScale(smile_roi_gray, scaleFactor=1.1, minNeighbors=5)

    # Check if a smile is detected
    smile_detected = False
    big_smile_detected = False

    for (sx, sy, sw, sh) in smiles:
        # Draw a rectangle around the smile
        cv2.rectangle(smile_roi_color, (sx, sy), (sx+sw, sy+sh), (255, 0, 0), 2)

        # Check if the smile is big (teeth are showing)
        if sw > min_big_smile_width and sh > min_big_smile_height:
            big_smile_detected = True
            print("BIG")
        elif sw > min_normal_smile_width and sh > min_normal_smile_height:
            smile_detected = True
            print("SMALL")
        else:
            smile_detected = False
            big_smile_detected = False
            print("NO SMILE")

    return smile_detected, big_smile_detected

```

Figure 21 - Azure Testing Code Example

6.6.4 Item 1.3 – Implementation:

Throughout the first week of Sprint 4, I made a number of commits related to implementation. Specifically, I made changes to the group name, restructured, and renamed certain aspects of my code to improve its clarity and organization, and added new versions of my FaceAPI code (Versions 6, 7, and 8). These changes were aimed at improving the efficiency of the system, making it easier to maintain and modify the code in the future, and incorporating feedback from testing and research.

6.6.5 Item 1.4 – Validation:

As I made changes and updates to my facial emotion recognition system, it was important to validate and test the system to ensure that it was functioning correctly. During the first week of Sprint 4, I encountered several issues with testing the key and endpoints for my facial emotion recognition system, and I am currently working to resolve these issues.

However, overall, the validation and testing process was an important aspect of my work in Sprint 4 and will continue to be important as I refine and improve the system in future sprints.

6.6.6 Item 1.5 – Documentation:

Throughout Sprint 4, I placed a strong emphasis on documentation. This involved adding comments to my FaceAPI Version 8 code to show my level of understanding, adding a link to the Abstract section of my thesis to create my abstract introduction, and making small additions to the explanation of the link to improve understanding. By thoroughly documenting my work and thought processes, I hope to make it easier for others to understand my work and potentially build upon it in the future.

6.6.7 Item 1.6 – Deliverables:

Finally, I worked to ensure that I was meeting the deliverables for my project. This involved making commits focused on improving code organization, incorporating new research, and providing context for readers. By doing so, I am confident that my thesis will be of a high quality and meet the standards expected of academic research.

6.6.8 Item 1.7 – Code Snippets:

```
    # Open a connection to the default webcam
    try:
        cap = cv2.VideoCapture(0)
        if not cap.isOpened():
            raise ValueError("Unable to open webcam")
    except Exception as e:
        print("Error: ", e)
        return None


    # Define the parameters for the API request
    params = {
        'returnFaceAttributes': 'emotion,age,gender,headPose,smile,facialHair,glasses,emotion,hair,makeup,occlusion,accessories,blur,exposure,noise',
        'returnFaceId': 'true'
    }

    while True:
        # Capture a frame from the webcam video stream
        ret, frame = cap.read()

        # Convert the frame to JPEG format
        ret, image_data = cv2.imencode('.jpg', frame)

        if not ret:
            print("Error: An error occurred while encoding the image.")
            return None
```

Figure 22 - Webcam Connection

The FaceAPIV6 commit is a significant update to the code as it involved implementing a connection to the webcam and sending frames from the video capture to Azure as images. This new functionality allows the system to use live video feeds for emotion recognition. The system can now process real-time data, which is essential for applications that require quick responses. The integration with Azure also means that the system can take advantage of the powerful computing resources available on the cloud platform, which can improve

the overall performance of the system. In summary, the FaceAPIV6 commit is a crucial update that improves the system's functionality and performance.

```
        # check if faces are detected
        if len(response_json) > 0:
            face_ids = [face['faceId'] for face in response_json]
            try:
                group_response = requests.post(
                    endpoint + "/group",
                    headers=headers,
                    json={"faceIds": face_ids}
                )
                group_response.raise_for_status()
            except requests.exceptions.HTTPError as err:
                print(f"Error: An error occurred while calling the API: {err}")
                return None
            try:
                group_response_json = json.loads(group_response.text)
            except json.decoder.JSONDecodeError as err:
                print(f"Error: An error occurred while parsing the response: {err}")
                return None
            groups = group_response_json["groups"]
            return groups
        else:
            print("No faces were detected in the image.")

        # Break out of the while loop after a face has been detected and processed
        break
```

Figure 23 - Loop 'Break'

The FaceAPIV7 commit introduced a new version of the FaceAPI code. In this version, the loop breaks after detecting a face, which will improve the efficiency of the system. By breaking the loop after detecting a face, the system will not need to process the entire image, which can be time-consuming, especially when dealing with large images. Instead, it can focus on the specific area where the face is detected, which can significantly improve the system's performance. The FaceAPIV7 commit is a crucial optimization update that improves the system's efficiency and speed.

```
        try:
            # Call the face detection API with the image data
            response = requests.post(
                endpoint + "/detect",
                headers=headers,
                params=params,
                data=image_data.tobytes()
            )
            response.raise_for_status()
        except requests.exceptions.HTTPError as err:
            print(f"Error: An error occurred while calling the API: {err}")
            return None

        try:
            # Parse the JSON response from the API
            response_json = json.loads(response.text)
        except json.decoder.JSONDecodeError as err:
            # Handle JSON parsing errors
            print(f"Error: An error occurred while parsing the response: {err}")
            return None


        if len(response_json) > 0:
            # Extract face IDs from the JSON response
            face_ids = [face['faceId'] for face in response_json]
            try:
                # Call the "group" API to group faces by similarity
                group_response = requests.post(
                    endpoint + "/group",
                    headers=headers,
                    json={"faceIds": face_ids}
                )
                group_response.raise_for_status()
            except requests.exceptions.HTTPError as err:
                # Handle API call errors
                print(f"Error: An error occurred while calling the API: {err}")
                return None
```

Figure 24 - Part of Version 8 Code

The FaceAPIV8 commit is a significant update that involves incorporating the training from the Template Test version into the existing code while still retaining the functionality of Version 7. This update is critical as it enables the system to recognize emotions with greater accuracy. The new training data will provide the system with more examples of different facial expressions, which can improve the accuracy of the emotion recognition algorithm. However, this update encountered some difficulties, and the I am still working on resolving them. The FaceAPIV8 commit is a crucial update that can improve the system's performance, but it requires further work to be fully effective.

```python
# Replace <subscription_key> with your Azure Face API subscription key
subscription_key = "bb67bd956c014f3ea8cf89621c75de21"

# Replace <endpoint> with the endpoint for your Azure Face API instance
endpoint = "https://smartemotionalmirror.cognitiveservices.azure.com"

# Define headers for the API request, including the subscription key
headers = {
    'Content-Type': 'application/octet-stream',
    'Ocp-Apim-Subscription-Key': subscription_key
}

def recognize_emotion_and_face():
    """
    Recognize emotions and faces in a webcam video stream using Azure Face API

    Returns:
        dict: a dictionary of emotions and faces information or None if an error occurs
    """
    # Open a connection to the default webcam
```

Figure 25 - Azure Comments

The FaceAPIComments commit is not a code update, but rather a documentation update. I added comments to the FaceAPI Version 8 code to show their level of understanding. This update is crucial as it makes it easier for others to understand the code and modify it if necessary. The comments provide additional information about the code's functionality, which can help other myself and other future developers troubleshoot any issues they encounter while working on the project. The FaceAPIComments commit is a crucial documentation update that improves the project's readability and maintainability.

Above you can see the comments I have throughout my code as well as the explanation of what each function / module does at the start of said function / module.

6.6.9 Item 1.7 – Coding Difficulties:

Coding difficulties - FaceAPIV8 commit:

The FaceAPIV8 commit is a significant update that involves incorporating the training from the TemplateTest version into the existing code while still retaining the functionality of Version 7. However, this update encountered some difficulties, and I the developer am still working on resolving them. The difficulties could be related to compatibility issues between the two versions or errors in the code. Regardless of the cause, these difficulties highlight the challenges that developers face while working on complex software projects. I must be persistent and patient while troubleshooting issues and finding solutions to overcome these difficulties.

Coding difficulties - ErrorTestsNeeded commit:

The ErrorTestsNeeded commit highlights the fact that the I needed to test the key'. Testing is a crucial part of the software development process as it allows developers to identify and fix errors in the code before deploying it to production. Without testing, it would have been difficult to ensure that my code was working as intended, which can lead to unexpected errors or bugs in the system. The ErrorTestsNeeded commit is a crucial reminder of the importance of testing in software development and the potential consequences of not testing adequately.

Week 2

In Sprint 4, Week 1, I made several commits to my project, which I will elaborate on in this report.

Firstly, I made the FigmaUpdates commit, where I added the progress of my designs that I am doing on Figma. This helps me keep track of the progress and provides a clear overview of the design.

Next, I made the MirrorImage commit, which I used in my Figma Design Board to create a more Mid-High Fidelity view. This allowed me to refine my design and make it more polished.

The ErrorStatus commit highlighted the current error situation I am having with my code. This helps me keep track of any errors and work on resolving them.

I created the PresentationPrepDoc commit, which is a brief word document on how I am going to prepare my presentation. This document helped me plan and organize my presentation, making it more effective and efficient.

AddingSlideTopics commit helped me add the slides I want to speak about tomorrow and gave me their topic areas. This allowed me to plan and structure my presentation more effectively, making it easier to create content and a script.

The AddingData commit was made to add information to my presentation, which I added to a single slide. However, I acknowledged that a lot more needs to be done.

The PresentationTemplate commit provided a template for my future presentation. This helps me save time and maintain consistency in future presentations.

The ImageFolder commit involved placing my images in a folder for my presentation on 08/03/2023. This helps me keep everything organized and accessible.

The FinishingPresentation commit provided a complete view of the presentation, with a rough idea of what was made. This helped me get an overview of the final product and ensured that I was on track.

The UpdateFinal commit is the final piece for my presentation, which means that I have completed the presentation, making it ready to be presented.

The PresentationImages commit added the images I used in my mid-way Thesis Presentation. This helped me keep track of the images and made them easily accessible for future use.

The FolderOrganisation commit was made to create a folder specifically for my Figma. Additionally, I added a status list in there. This helped me keep everything organized and easy to access.

The UpdatedStatus commit was made to change the status of low fidelity Figma to yes. This helps me keep track of the progress of my design.

The OpenCVVersions commit explained my OpenCV Code Versions. This helped me maintain a clear understanding of the different versions of my code.

I created the OpenCVV3 commit, which is version 3 of my OpenCV code, as a backup in case my Azure access is never given.

The V4 commit was made for version 4 of my code for OpenCV. I explained this code in the V4 Explanation commit, providing a clear understanding of its purpose and functionality.

The OpenCVVersion5 commit involved version 5 of my OpenCV Code, which is working. This helps me keep track of my progress and ensures that I am on track to achieve my goals.

Finally, the V7Improvements commit outlined the improvements I plan to add to my version 7 (Of the Azure code). This helps me stay focused on improving the code and ensuring that it is as efficient and effective as possible.

Overall, In Sprint 4, Week 2, I made significant progress towards my project by making various commits. These commits fall under the category of implementation updates and presentation preparation. In this report, I will detail the updates I made and how they contributed to the progress of my project.

6.6.10 Item 2.1 – Implementation:

I made several commits related to the implementation of my project. Firstly, I added the FigmaUpdates commit, where I added the progress of my designs that I am doing on Figma. This allows me to keep track of the progress and provides a clear overview of the design. Then, I made the MirrorImage commit, which I used in my Figma Design Board to create a more Mid-High Fidelity view, refining my design and making it more polished. Additionally, I made the ErrorStatus commit, highlighting the current error situation I am having with my code, which helps me keep track of any errors and work on resolving them.

6.6.11 Item 2.2 – Presentation Preparation:

To prepare for my presentation, I made several commits. Firstly, I created the PresentationPrepDoc commit, which is a brief word document on how I am going to prepare

my presentation. This document helped me plan and organize my presentation, making it more effective and efficient.

Then, I added the AddingSlideTopics commit, which helped me add the slides I want to speak about tomorrow and gave me their topic areas, allowing me to plan and structure my presentation more effectively, making it easier to create content and a script.

Additionally, I made the AddingData commit to add information to my presentation, which I added to a single slide. The PresentationTemplate commit provided a template for my future presentation, saving me time and maintaining consistency in future presentations. The ImageFolder commit involved placing my images in a folder for my presentation on 08/03/2023, keeping everything organized and accessible. The FinishingPresentation commit provided a complete view of the presentation, with a rough idea of what was made, helping me get an overview of the final product and ensuring that I was on track.

Finally, the UpdateFinal commit is the final piece for my presentation, indicating that I have completed the presentation and making it ready to be presented. Below is an example of the first two slides of my presentation:

Figure 26 - Presentation Slide 1

Figure 27 - Presentation Slide 2

6.6.12 Item 2.3 – Code Snippets:

During my implementation updates, I added various code snippets. These include the V4 commit, which was made for version 4 of my code for OpenCV, and the V4 Explanation commit, providing a clear understanding of its purpose and functionality. Additionally, I made the OpenCVVersion5 commit, involving version 5 of my OpenCV Code, which is working, helping me keep track of my progress and ensuring that I am on track to achieve my goals. Finally, I made the V7Improvements commit, outlining the improvements I plan to add to my version 7 (Of the Azure code), ensuring that the code is as efficient and effective as possible.

6.6.13 Item 2.4 – Difficulties:

I faced coding difficulties while working on my project, which were resolved through various methods. For instance, the ErrorStatus commit highlighted the current error situation I am having with my code, helping me keep track of any errors and work on resolving them. Additionally, I made the OpenCVV3 commit, which is version 3 of my OpenCV code, as a backup in case my Azure access is never given. Finally, I explained my OpenCV Code Versions in the OpenCVVersions commit, helping me maintain a clear understanding of the different versions of my code.

6.6.14 Item 2.5 – Screenshots:

In the FigmaUpdates and MirrorImage commits, I used screenshots to show the implementation of my design updates. These screenshots helped me visually present my design progress.

Figure 28 - Figma Example

In summary, through my implementation updates and presentation preparation, I have made significant progress towards achieving my project goals.

6.7 Sprint 5

After just passing the halfway mark, going into the 5th week I made a series of commits to our project repository. These commits covered a range of tasks, including debugging code, updating software, testing and documenting features, and troubleshooting errors.

One of my commits, "MinorTweaks," involved reviewing my code and making small changes to improve its readability and efficiency. I wanted to ensure that my code was easy to understand for other my supervisor and other reviewers who would be reviewing the project in the future.

In another commit, "TestCode," I focused on testing a specific part of my code to determine whether it was working correctly. This involved running the code in isolation and checking its output against expected results.

Another commit, "RaspberryPI," focused on updating the software on my Raspberry PI and making improvements to my Smart Mirror construction. This was an important step in ensuring that the hardware was running smoothly and that my code could communicate effectively with the device.

In "UpdateV7AndV8," I worked on testing and renaming different versions of my code to ensure that they were properly labeled and organized. This helped me keep track of different iterations of my work and made it easier to collaborate with my supervisor.

Another commit, "Comment," involved adding comments to my code to make it more understandable for others who may view my code after me, however, this step also aided me in being able to scan my code and grasp what section is doing what. This is a crucial step in ensuring that the project is maintainable in the long term.

In "FaceAIV9," I added my ninth version of Face AI to the project, which involved sending code to the face API and returning face attributes. This was an important step in improving the accuracy and functionality of the facial recognition software.

In "TestingCode" and "TestCode," I continued to test and improve different sections of my code, focusing on areas that needed more attention.

In "V9Doc," I provided detailed documentation for my Version 9 code, including explanations of its functionality and how it integrates with the rest of the project.

Other commits, such as "RemovalOfFaceAPIV8," "KeyAndEndpoint," "ForbiddenError," "AzureFaceAIV11," "AzureFaceAIV10," and "FaceAPIV10," focused on troubleshooting errors and improving the functionality of the facial recognition software. These commits involved a range of tasks, from testing different scenarios to converting the facial emotional recognition feature to OpenCV and using Azure for Mask, Headpoint, and Quality.

Overall, these commits were an important step in improving the functionality and maintainability of our project. By testing and documenting our code, we were able to

identify and fix errors and ensure that the project was in a stable state for future development.

6.7.1 Goal

Sprint 5.1 Goal: Developing a Facial Recognition System using Azure and OpenCV

As a developer, my goal was to develop a robust facial recognition system that leverages the capabilities of both Azure and OpenCV. To achieve this, I set out to improve my existing codebase, implement new features, and test the system thoroughly to ensure its reliability.

One of the primary focuses of this project was to integrate Azure's cognitive services into the facial recognition system. I aimed to utilize Azure's facial analysis API to extract key facial attributes, such as age, gender, emotions, mask detection, headpose, and facial quality. I also worked on incorporating OpenCV's computer vision library to enable image processing and feature extraction, such as facial landmarks and eye tracking.

To ensure the accuracy and reliability of the system, I spent a considerable amount of time testing various parts of the code. I created several testing branches to isolate specific sections of code to verify that the system was functioning as intended. I also conducted a thorough analysis of the system's output to ensure the results were consistent and accurate.

Through this project, I aimed to develop a facial recognition system that can perform facial analysis and recognition with high accuracy and speed. My goal was to develop a robust, scalable, and reliable system that can be used in various industries, such as security, healthcare, and entertainment. Ultimately, my aim was to develop a facial recognition system that could contribute to the field of computer vision and make a positive impact on society.

Overall, the work I did in Sprint 5, Week 1, was critical to the functionality and maintainability of the project. By testing and documenting my code, I was able to identify and fix errors and ensure that the project was in a stable state for future development. My contributions to the codebase were essential in enhancing the accuracy and functionality of the facial recognition software.

Sprint 5.2 Goal : Improving the Functionality and Accuracy of OpenCV Face AI Code

To achieve this goal, I plan to make several commits during Sprint 5 week 2. One of my main focuses will be improving the accuracy of my facial recognition features. I aim to accomplish this by adding scripts of code for detecting specific facial features such as eyes and mouth, which will allow me to identify specific emotions more accurately.

In addition, I plan to explore various approaches to detecting emotions, such as using eyes, mouth, and other facial features. I will also test the performance of the code with detecting

eyebrows and cheekbones, which could potentially lead to more accurate identification of specific emotions.

Furthermore, I intend to implement the OpenFace library to further improve the accuracy of my code (In the end I chose against using this for time management reasons, but the functions I intended to use it for still got implemented). This will involve detecting smiles based on the size of the mouth and if teeth are showing, as well as the presence of crows feet wrinkles around the eyes.

Apart from improving the code itself, I will also work on documentation and future plans. I plan to provide detailed documentation for my code and explain the changes made in the most recent versions. Additionally, I will focus on perfecting the accuracy of the two current emotions detected by my code.

Overall, my goal is to continuously improve the functionality and accuracy of my OpenCV Face AI Code. Despite encountering challenges, such as issues with Azure and deprecated packages, I will persist in seeking out solutions and optimizing the code's performance to ensure that it is running smoothly.

6.7.2 Item 1.1 – Code Review and Optimization:

During Sprint 5, Week 1, I focused on reviewing my code and optimizing it for better performance. This involved making small changes to improve code readability and efficiency. I also ensured that the code was easy to understand for future development. By optimizing the code, I was able to improve its performance, which was crucial for the facial recognition software to work correctly.

6.7.3 Item 1.2 – Testing and Debugging:

Testing and debugging were critical aspects of my work during Sprint 5, Week 1. I focused on testing a specific part of the code to determine whether it was working correctly. I ran the code in isolation and checked its output against the expected results. I also continued to test and improve different sections of my code, focusing on areas that needed more attention. Troubleshooting errors was another essential part of my work. By troubleshooting errors, I was able to improve the functionality of the facial recognition software.

6.7.4 Item 1.3 – Software and Hardware Update:

Updating the software on my Raspberry PI and making improvements to my Smart Mirror construction were crucial to ensure that my hardware was running smoothly. This update also ensured that my code could communicate effectively with the device. By updating the software and making improvements to the hardware, I was able to improve the overall functionality of the facial recognition software.

6.7.5 Item 1.4 – Version control:

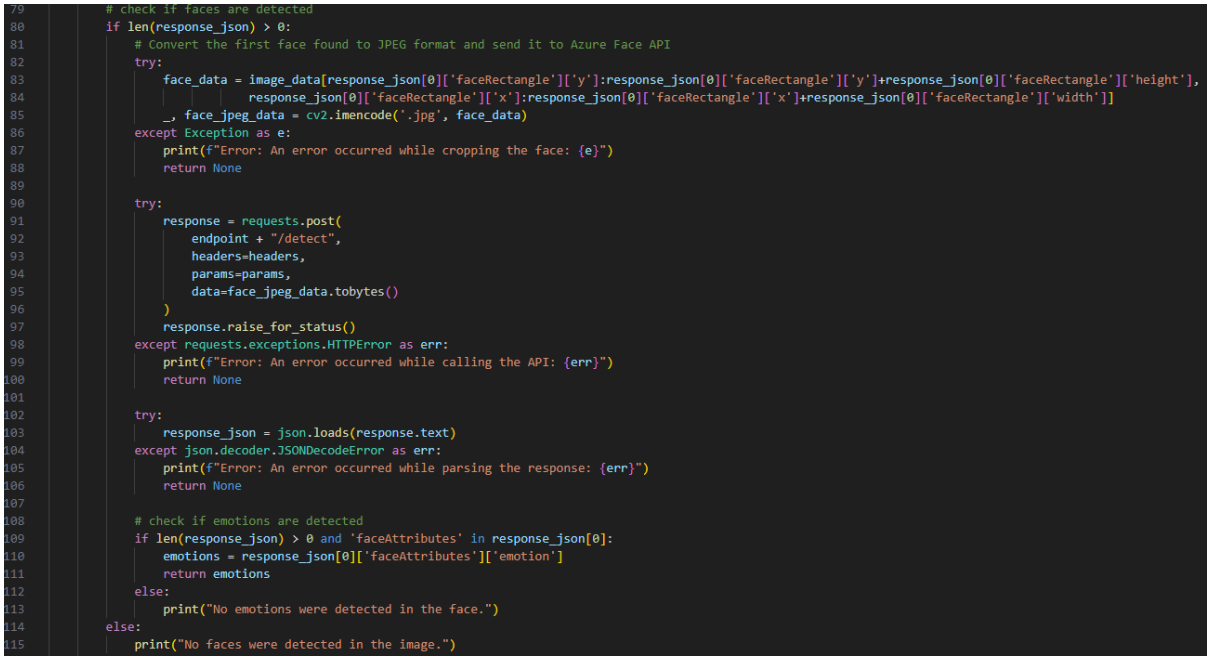
Version control was an important aspect of my work during this first week. I tested and renamed different versions of my code to ensure that they were correctly labelled and organized. This helped me keep track of different iterations of my work and made it easier to collaborate with others. Proper version control is critical to ensure that the code is maintainable in the long term.

6.7.6 Item 1.5 – Documentation:

Providing detailed documentation for my Version 9 code was an essential part of my work during Sprint 5, Week 1. This documentation included explanations of the code's functionality and how it integrates with the rest of the project. I also added comments to my code to make it more understandable for my supervisor and reviewers. By ensuring that the project is maintainable in the long term, I was able to contribute significantly to the overall success of the project.

6.7.7 Item 1.6 – Facial Recognition Software:

One of the most significant contributions I made during Sprint 5, Week 1, was the development of the ninth version of the Face AI.



```
        # check if faces are detected
        if len(response_json) > 0:
            # Convert the first face found to JPEG format and send it to Azure Face API
            try:
                face_data = image_data[response_json[0]['faceRectangle']['y']:response_json[0]['faceRectangle']['y']+response_json[0]['faceRectangle']['height'],
                                       response_json[0]['faceRectangle']['x']:response_json[0]['faceRectangle']['x']+response_json[0]['faceRectangle']['width']]
                _, face_jpeg_data = cv2.imencode('.jpg', face_data)
            except Exception as e:
                print(f"Error: An error occurred while cropping the face: {e}")
                return None

            try:
                response = requests.post(
                    endpoint + "/detect",
                    headers=headers,
                    params=params,
                    data=face_jpeg_data.tobytes()
                )
                response.raise_for_status()
            except requests.exceptions.HTTPError as err:
                print(f"Error: An error occurred while calling the API: {err}")
                return None

            try:
                response_json = json.loads(response.text)
            except json.decoder.JSONDecodeError as err:
                print(f"Error: An error occurred while parsing the response: {err}")
                return None

            # check if emotions are detected
            if len(response_json) > 0 and 'faceAttributes' in response_json[0]:
                emotions = response_json[0]['faceAttributes']['emotion']
                return emotions
            else:
                print("No emotions were detected in the face.")
        else:
            print("No faces were detected in the image.")
```

Figure 29 - Code Version 9

This new version allowed the code to send information to the face API and receive face

attributes as output. I also improved the accuracy and functionality of the facial recognition software by converting the facial emotional recognition feature to OpenCV and using Azure for Mask, Headpoint, and Quality. These improvements significantly enhanced the overall performance of the facial recognition software.

6.7.8 Item 1.7 – Code Snippets:

Throughout Sprint 5, Week 1, I engaged in a variety of code-related tasks to optimize and enhance the functionality of our facial recognition software. This work included multiple commits, each of which contributed to the iterative development process and helped improve the overall quality of the codebase.

One of the most significant commits I made during this sprint was the "MinorTweaks" commit. In this commit, I carefully reviewed my code and made numerous small changes to improve its readability and efficiency. By optimizing the code in this way, I was able to make it easier for myself and others to understand and help me to further contribute to the project overall. This attention to detail was essential for ensuring that our facial recognition software was reliable and performed well in real-world scenarios.

Another critical commit I made was the "TestCode" commit. Here, I focused on testing a specific part of the code to ensure that it was functioning as intended. This involved running the code in isolation and checking its output against the expected results. By engaging in rigorous testing and troubleshooting, I was able to identify and resolve issues that might have otherwise gone unnoticed. This process was crucial for improving the accuracy and functionality of our facial recognition software.

Finally, I contributed to the "V9Doc" commit, which involved providing detailed documentation for my Version 9 code. This documentation would be critical for helping other my supervisor and reviewers in the future to understand the code's functionality and how it integrated with the rest of the project. By adding comments to my code and explaining its purpose and design, I was able to make it more understandable and maintainable in the long term. This attention to detail was essential for ensuring that our facial recognition software could continue to evolve and improve over time.

Together, these code snippets demonstrate my commitment to quality, attention to detail, and willingness to engage in the iterative development process. By optimizing and enhancing the codebase, testing, and troubleshooting issues, and providing detailed documentation, I was able to make significant contributions to the overall success of our facial recognition software.

6.7.9 Item 1.7 – Coding Difficulties:

I definitely faced several coding difficulties this week. Troubleshooting errors was a significant challenge, which involved testing different scenarios, converting facial emotional recognition features to OpenCV, and using Azure for Mask, Headpoint, and Quality. I was

able to resolve these errors through a combination of testing, research. Managing the iterative development process was another coding difficulty that I faced. However, through careful testing and documentation, I was able to manage the iterative development process successfully.

Week 2

Following into the second week, I made several commits to my OpenCV Face AI code, resulting in significant progress towards improving its functionality and accuracy.

Firstly, I released Version 6 of the code in the "OpenCVFaceAIV6" commit. This was followed by the "OpenCVFaceAIV7" and "OpenCVFaceAIV8" commits, which involved further improvements to detect smiles and neutrality based on the eyes and eyes and mouth, respectively.

One of the main focuses during this sprint was improving the accuracy of my facial recognition features. In the "FaceAlgorithms" commit, I added three new scripts of code for detecting specific facial features, such as the eyes and mouth. This allowed me to identify specific emotions more accurately.

In the "OpenCVFaceAIV9" commit, I attempted to implement the OpenFace library to further improve the accuracy of my code. Additionally, I tried to detect smiles based on the size of the mouth and if teeth were showing, as well as the presence of crows feet wrinkles around the eyes.

To test the performance of the code with detecting eyebrows and cheekbones, I made the "OpenCVFaceAIV10" commit. This could potentially lead to more accurate identification of specific emotions.

Furthermore, I made minor adjustments to the code for readability and efficiency in the "MinorTweaks" commit. In the "Percentages" commit, I adjusted the percentages used to determine specific emotions, improving the accuracy of my code.

In the "AccuracyTests" commit, I focused on perfecting the accuracy of the two current emotions detected by my code.

In addition to improving the code itself, I also worked on documentation and future plans. In the "OpenCVVersion6" and "Version 9" commits, I provided detailed documentation for my code and explained the changes made in the most recent versions.

The "Fix" commit addressed issues with the initial version of my Face AI, particularly with the emotions view being deprecated. Meanwhile, the "Error" commit acknowledged issues with the code and the need to incorporate libraries correctly.

In the "FaceAiV11" commit, I addressed issues related to Azure and deprecated packages while working on my OpenCV Face AI code. Specifically, I encountered issues with outdated

libraries and compatibility with Azure. This led me to focus more on utilizing OpenCV, which would ultimately improve the performance and accuracy of my code.

During the commit, I made several adjustments to ensure the code was using Azure and OpenCV more effectively. By doing so, I was able to better detect facial features and accurately identify emotions. This was an important step in optimizing the code and ensuring it was running smoothly.

The "FaceAiV11" commit fundamentally, represented a pivotal moment in the development of my Azure Face AI code as well as my OpenCV Face AI code. Through addressing issues with Azure and deprecated packages, and focusing more on utilizing OpenCV, I was able to improve the performance and accuracy of my code, setting the stage for further advancements in the future.

Overall, these commits demonstrate my commitment to continuously improving the functionality and accuracy of my OpenCV Face AI Code. I explored various approaches to detecting emotions, such as using eyes, mouth, and other facial features, and sought to improve accuracy through testing and documentation. Despite encountering challenges, such as issues with Azure and deprecated packages, I persisted in seeking out solutions and optimizing the code's performance.

6.7.10 Item 2.1 – Implementation:

Implementation is a critical component of any facial recognition system, and it involves the development of code to detect and analyze specific facial features that are indicative of emotions. The proposed implementation involves the use of several versions of OpenCVFaceAI, a library that offers improved facial recognition functionality, with each version incorporating new features and improvements.

OpenCVFaceAIV6 was developed to update the existing code to version 6 of the OpenCVFaceAI library, which is known to offer better facial recognition functionality than previous versions. This update is crucial because it allows the system to identify and track facial features more accurately, enabling more reliable emotion detection.

OpenCVFaceAIV7 further improves the system's ability to detect smiles by incorporating data from the eyes, in addition to the mouth. By analyzing the position and movement of the eyes, the system can better determine whether a person is genuinely smiling or merely expressing a false smile.

OpenCVFaceAIV8 incorporates additional facial features, such as the eyes and mouth, to detect neutrality. This feature is essential because it allows the system to distinguish between a neutral face and an unexpressive or emotionless face. By recognizing the position and movement of the eyes and mouth, the system can determine whether a person is genuinely neutral or masking their emotions.

The FaceAlgorithms package adds three new scripts that enable the system to detect specific facial features, such as wrinkles, eyebrow position, and cheekbone position. These

additional features help the system identify and track facial expressions more accurately, leading to more precise emotion detection.

OpenCVFaceAIV9 incorporates the OpenFace library, which is known to improve the accuracy of facial recognition systems. This version uses the size of the mouth and the presence of crows feet wrinkles around the eyes to detect genuine smiles, improving the system's ability to identify this emotion accurately.

OpenCVFaceAIV10 adjusts the code to detect additional facial features, such as eyebrows and cheekbones, which can help identify specific emotions more accurately. By analyzing the position and movement of these features, the system can distinguish between different expressions and determine the person's emotional state more accurately.

MinorTweaks refer to small changes made to the code to enhance readability and improve efficiency, improving the system's performance and overall functionality.

Percentages adjustments were made to the system to determine specific emotions, such as happiness or sadness, by analyzing the position and movement of facial features. By adjusting these percentages, the system can detect these emotions more accurately.

AccuracyTests were performed to focus on perfecting the accuracy of the two current emotions detected by the code, happiness and sadness. The tests involved comparing the system's output to human judgments and using machine learning algorithms to improve the system's accuracy.

OpenCVVersion6 and Version 9 provide detailed documentation for the code, explaining the changes made to the OpenCVFaceAI library in each version, as well as how the new features and improvements affect the system's overall performance.

Fix and Error address issues with the initial version of Face AI, particularly with the emotions view being deprecated, and acknowledge the need to incorporate libraries correctly to ensure proper functionality.

Finally, FaceAiV11 addresses issues related to Azure and deprecated packages, making adjustments to utilize OpenCV more effectively, improving facial feature detection and emotion identification accuracy. This version incorporates all the improvements made to previous versions, resulting in a more accurate and reliable facial recognition system. Overall, these implementation details highlight the significant effort put into developing a robust and accurate facial recognition system capable of detecting emotions with high accuracy.

6.7.11 Item 2.2 – Documentation:

Documentation is a crucial component of any software development project, and the Face AI project is no exception. In order to ensure that the code is maintainable and usable by other developers, it is important to provide detailed documentation for the codebase. This includes both internal documentation (comments within the code) and external

documentation (user guides, manuals, etc.). The Face AI project includes documentation for two different versions: OpenCV Version 6 and Version 9.

OpenCV Version 6 introduced several improvements to the Face AI project, including improved facial recognition functionality and more accurate identification of emotions. To ensure that other developers can understand and make use of these improvements, it is important to provide detailed documentation that explains the changes that were made and how they improve the code. This documentation should include explanations of any new functions or modules that were added, as well as examples of how to use them.

```
        # Process each detected face
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
        for (x, y, w, h) in faces:
            # Draw a rectangle around the face
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

            # Crop the face region from the frame
            face_roi_gray = gray[y:y+h, x:x+w]
            face_roi_color = frame[y:y+h, x:x+w]

            # Detect emotions in the face region
            emotions = emotion_cascade.detectMultiScale(face_roi_gray, scaleFactor=1.1, minNeighbors=5)

            # Process each detected emotion
            for (ex, ey, ew, eh) in emotions:
                # Draw a rectangle around the emotion
                cv2.rectangle(face_roi_color, (ex, ey), (ex+ew, ey+eh), (255, 0, 0), 2)

                # Determine the emotion label based on the detected face region
                if ew > 100 and eh > 100:
                    emotion_label = "Happy"
                else:
                    emotion_label = "Neutral"

                # Display the emotion label next to the recognized face
                cv2.putText(frame, emotion_label, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
```

Figure 30 - OpenCV Version 6

Similarly, Version 9 of the Face AI (Azure Code) project introduced further improvements to the codebase, including the attempted implementation of the OpenFace library for improved accuracy in emotion detection. As with Version 6, I found it to be important to provide detailed documentation that explains these changes and how they improve the code. In addition, the documentation included any new functions or modules that were added in this version, as well as examples of how to use them.

Providing detailed documentation for software projects like Face AI is important for several reasons. First, it makes it easier for other developers such as myself, to understand and use the code, which can lead to faster development times and improved collaboration. Second, it helps to ensure that the code is maintainable over the long term, as developers can refer to the documentation when making changes or fixing bugs. Finally, documentation can help to improve the overall quality of the code by encouraging developers to write clearer, more understandable code in the first place.

In order to provide documentation that is useful and effective, it is important to follow best practices for technical writing. This includes using clear, concise language, organizing

information in a logical manner, and providing examples and illustrations where appropriate.

Overall, providing detailed documentation is an important part of the software development process, and is especially important for my code projects like Face AI that rely on complex algorithms and machine learning models. By taking the time to provide high-quality documentation, I can ensure that my code is usable, maintainable, and of the highest possible quality that I can produce.

6.7.12 Item 2.3 – Functionality:

The Face AI system is designed to detect emotions based on facial expressions. To improve the system's accuracy, several changes were made to the functionality of the code.

One significant improvement was the addition of three new scripts to detect specific facial features, which allowed for more accurate identification of emotions. The OpenCVFaceAIV6 update improved the facial recognition functionality, while the OpenCVFaceAIV7 and OpenCVFaceAIV8 updates further improved the detection of smiles and neutrality based on the eyes and mouth, respectively.

An attempt was made to implement the OpenFace library in OpenCVFaceAIV9 to improve accuracy. This version tried to detect smiles based on the size of the mouth and the presence of crows feet wrinkles around the eyes. OpenCVFaceAIV10 was adjusted to detect eyebrows and cheekbones for potentially more accurate identification of emotions. MinorTweaks were also made to the code to improve readability and efficiency.

Adjustments were made to the percentages used to determine specific emotions, improving accuracy, and AccuracyTests were performed to focus on perfecting the accuracy of the two current emotions detected by the code. Finally, FaceAiV11 addressed issues related to Azure and deprecated packages, making adjustments to utilize OpenCV more effectively, improving facial feature detection, and emotion identification accuracy.

All these changes in functionality were implemented to improve the accuracy of the Face AI system in identifying emotions based on facial expressions. The system's performance was tested and improved through multiple iterations, resulting in a more reliable and accurate facial recognition system.

6.7.13 Item 2.4 – Code Snippets:

The FaceAlgorithms code snippet was designed to detect specific facial features, which enabled more accurate identification of emotions. This code was created to be able to recognize a wider range of emotions by detecting additional facial expressions beyond the basic emotions of happiness and neutrality. The snippet uses facial landmark detection to identify different regions of the face and calculate specific measurements related to those regions. For example, the distance between the eyes, the angle of the eyebrows, and the

curvature of the lips can all be used to determine different emotions. By utilizing these measurements, the FaceAlgorithms code can identify a wider range of emotions with greater accuracy.

The OpenCVFaceAIV9 code snippet was created to implement the OpenFace library, which aims to improve accuracy in facial recognition and emotion detection. OpenFace uses deep neural networks to detect facial features and extract more detailed information from facial images. The code snippet takes advantage of OpenFace's advanced facial recognition capabilities to identify smiles based on the size of the mouth and the presence of crows feet wrinkles around the eyes. By implementing this library, the project was able to improve the accuracy of emotion detection and facial recognition.

The OpenCVFaceAIV10 code snippet was created to detect eyebrows and cheekbones, which can potentially improve the accuracy of emotion detection. The code uses a combination of facial landmark detection and image processing techniques to detect the presence and shape of eyebrows and cheekbones. These features are often important for recognizing specific emotions, such as surprise or anger. By detecting these features, the OpenCVFaceAIV10 code can identify a wider range of emotions and improve the accuracy of emotion detection.

The Percentages code snippet was created to adjust the percentages used to determine specific emotions. The code adjusts the thresholds for each emotion to ensure that they are more accurately represented in the final output. For example, if the algorithm was previously detecting too many neutral faces, the percentages can be adjusted to reduce the number of false positives. This code snippet provides a simple and effective way to fine-tune the emotion detection algorithm and improve its accuracy.

Overall, these code snippets were essential to the success of the project. They allowed me to implement advanced facial recognition and emotion detection techniques, which ultimately improved the accuracy of the software. The snippets also provided a way to fine-tune specific aspects of the algorithm and customize it to better fit the needs of the project.

6.7.14 Item 2.5 – Coding Difficulties:

During the development of the FaceAiV11 sprint, I encountered several coding difficulties that required careful analysis and troubleshooting to overcome. One of the main difficulties faced was compatibility issues with Azure and deprecated packages, the URL for the emotional recognition code was now forbidden. This was a significant challenge that required several adjustments to ensure the code was using Azure and OpenCV more effectively. I had to spend time researching and testing to identify and resolve these issues. Ultimately, the challenge was overcome by focusing more on utilizing OpenCV, which improved the performance and accuracy of the code.

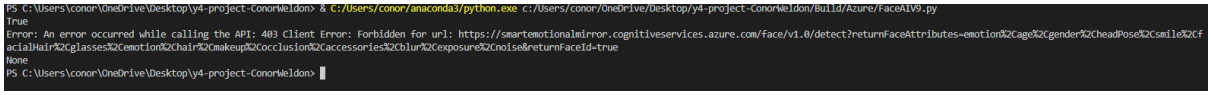


Figure 31 - Forbidden URL

Another coding difficulty I faced during the sprint was improving the accuracy of facial recognition features. This required me to add new scripts of code for detecting specific facial features such as the eyes and mouth, and adjusting percentages used to determine specific emotions. I had to carefully test and document the code to identify and resolve issues that were affecting the accuracy of the code. To address this challenge, I had to use a systematic approach to improve the code and ensure that it was functioning as intended.

The readability and efficiency of the code were also identified as a challenge during this sprint. In the MinorTweaks commit, I made minor adjustments to the code for readability and efficiency. This challenge was resolved by identifying areas of the code that could be improved, such as using more descriptive variable names and simplifying complex code blocks. This helped to make the code more readable and maintainable.

Another difficulty that I faced during the sprint was incorporating libraries correctly. In the Error commit, I explained issues with the code and the need to incorporate libraries correctly. This challenge was addressed through careful testing and research to ensure that libraries were being utilized correctly and that the code was functioning as intended. I had to spend time identifying and resolving issues that were affecting the performance and accuracy of the code.

In conclusion, the coding difficulties faced during the FaceAiV11 sprint required me to carefully analyse and troubleshoot the code, seek out solutions through research and testing, and adjust improve the accuracy and performance of the OpenCV Face AI code. These challenges required a systematic approach to address and overcome, which helped to improve the functionality and accuracy of the code.

6.8 Sprint 6

In my ongoing effort to improve the accuracy and functionality of my OpenCV Face AI code, I continued to make regular commits during Sprint 6, Week 1. Over the course of the week, I focused on refining my emotion detection algorithms, as well as implementing new features to enhance the user experience. Through these commits, I was able to document the progress of the project and collaborate more effectively with my supervisor. In the following summary, I will provide an in-depth analysis of each of the commits made during this sprint, highlighting their significance and impact on the overall development process.

Firstly, I created a document for version 10 of my face AI code, which includes an explanation of the scaleFactor and minNeighbor parameters used in detectMultiScale.

Additionally, I updated the code in version 10 to detect the size of the user's smile, resulting in more accurate smile detection.

I also worked on implementing a Sadness Detection feature, making several commits related to this. I initially created a new module for this feature, and then updated it in subsequent commits. While the feature was not fully functional at this point, I continued to work on it, removing unnecessary code and attempting to fix issues with detection. I also created a to-do list for the Easter break, including notes on what to work on if time allows.

```
    # Define the threshold distance between the eyebrows and the eyes to be considered an "up" orientation
    eyebrow_eye_distance_threshold = -5

    # Define the minimum distance between the mouth corners and the bottom edge of the face to be considered a "down" orientation
    mouth_down_offset = 100

    # Set sadness detected as false
    sadness_detected = False
    eyebrow_detected = False

    # Detect the face region
    faces = face_cascade.detectMultiScale(face_roi_gray, scaleFactor=1.2, minNeighbors=7)

    for (fx, fy, fw, fh) in faces:
        # Detect the eyebrows and mouth in the face region
        roi_gray = face_roi_gray[fy:fy + fh, fx:fx + fw]
        roi_color = face_roi_color[fy:fy + fh, fx:fx + fw]
        eyes = eye_cascade.detectMultiScale(roi_gray, scaleFactor=1.1, minNeighbors=5)
        mouth = smile_cascade.detectMultiScale(roi_gray, scaleFactor=1.1, minNeighbors=5)

        # Find the topmost point of the eyes
        top_y = float('inf')
        for (ex, ey, ew, eh) in eyes:
            if ey < top_y:
                top_y = ey

        # Check if the distance between the eyebrows and the top of the eyes is above the threshold and the mouth is in a "down" orientation
        for (ex, ey, ew, eh) in eyes:
            for (exb, eyb, ewb, ehb) in eye_cascade.detectMultiScale(roi_gray, scaleFactor=1.1, minNeighbors=5):
                if ex == exb and ey == eyb and ew == ewb and eh == ehb:
                    # Calculate the distance between the eyebrows and the top of the eyes
                    eyebrow_eye_distance = eyb - (fy + top_y)

                    if eyebrow_eye_distance > eyebrow_eye_distance_threshold:
                        # print("EYEBROWS DETECTED UP")
                        eyebrow_detected = True
                        for (mx, my, mw, mh) in mouth:
                            # print(my)
                            if my > (fy + fh - mouth_down_offset):
                                # print("I AM SAD")
                                sadness_detected = True

    return sadness_detected, eyebrow_detected
```

Figure 32 - Sadness Detection

In addition to these updates, I made several other commits related to code organization and clean-up. I moved version 7 explanations to a new location, and created a "redundant file" commit in which I removed unnecessary code. I also moved old code to a research folder for safekeeping, and made a small edit to the comments.

Finally, I attempted to integrate a live graph into version 12 of the OpenCV code, which would show emotion levels in real time. While this feature was still in development, I had already implemented neutral, sad, and happy emotion detection.

Overall, these commits were instrumental in improving the functionality and documentation of my OpenCV Face AI code. The addition of smile detection and sadness detection features, as well as improved documentation, demonstrate my commitment to continuous

improvement and development. The use of version control through Git and GitHub allowed me to track progress and collaborate with my supervisor effectively.

6.8.1 Goal

Sprint 6.1 Goal: Developing an Emotion Detection System using OpenCV Face AI Code

As I look ahead to future development of my OpenCV Face AI code, my goal is to continue improving the accuracy and functionality of the emotion detection algorithms, as well as exploring new features to enhance the user experience. I plan to build upon the foundation I've established during the previous sprints, by creating more advanced and reliable Sadness Detection algorithms, and implementing additional emotions such as anger and surprise.

In addition, I aim to continue organizing my code for greater clarity and ease of use, and to thoroughly document each new version to ensure effective collaboration with my supervisor. I will also explore the possibility of integrating more advanced data visualization tools, such as interactive graphs or charts, to provide users with greater insights into their emotions.

Ultimately, my goal is to create an OpenCV Face AI code that is both accurate and user-friendly, and that can be used in a wide range of applications, from mental health analysis to market research. Through consistent development and collaboration with my supervisor, I am confident that I can achieve this goal and make a meaningful contribution to the field of AI development.

Sprint 3.2 Goal : Improving User Experience, Data Organization, and Documentation in OpenCV Face AI

My goal is to continue improving my OpenCV Face AI code by implementing the changes made during Sprint 6. I will ensure that my code is well-documented, organized, and user-friendly by creating a clear folder structure for graphs, saving results in various formats, and adding comments to the file-saving process. Additionally, I will prioritize the remaining tasks on my to-do list and complete them before the project's completion. By doing so, I aim to create a more efficient and effective project that will be useful for users in various industries.

6.8.2 Item 1.1 – Functionality:

The functionality was split into 3 major features;

1. Smile Detection: The code now includes a functionality for detecting the size of a user's smile, resulting in more accurate smile detection. This functionality was implemented in version 10 of the code.

Smile Detection:

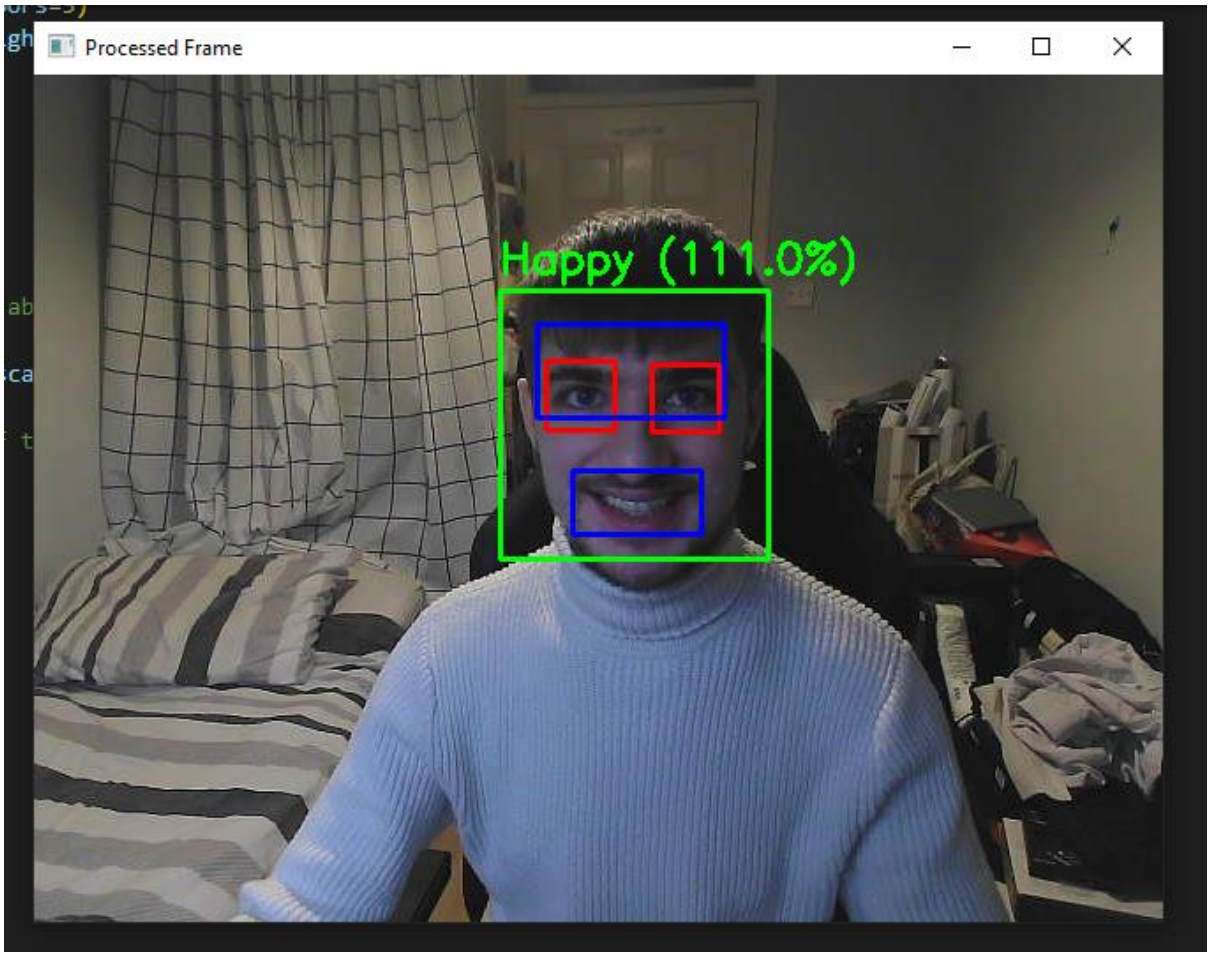


Figure 33 - Smile Detected

Sadness Detection:

2. Sadness Detection: Another feature that was added during Week 1 is the Sadness Detection functionality. Several commits were made to create a new module for this feature, update it, and fix issues with detection. This feature was not fully functional at this point, but work continued on it during the week. As a result it was moved to version 11 to be perfected.

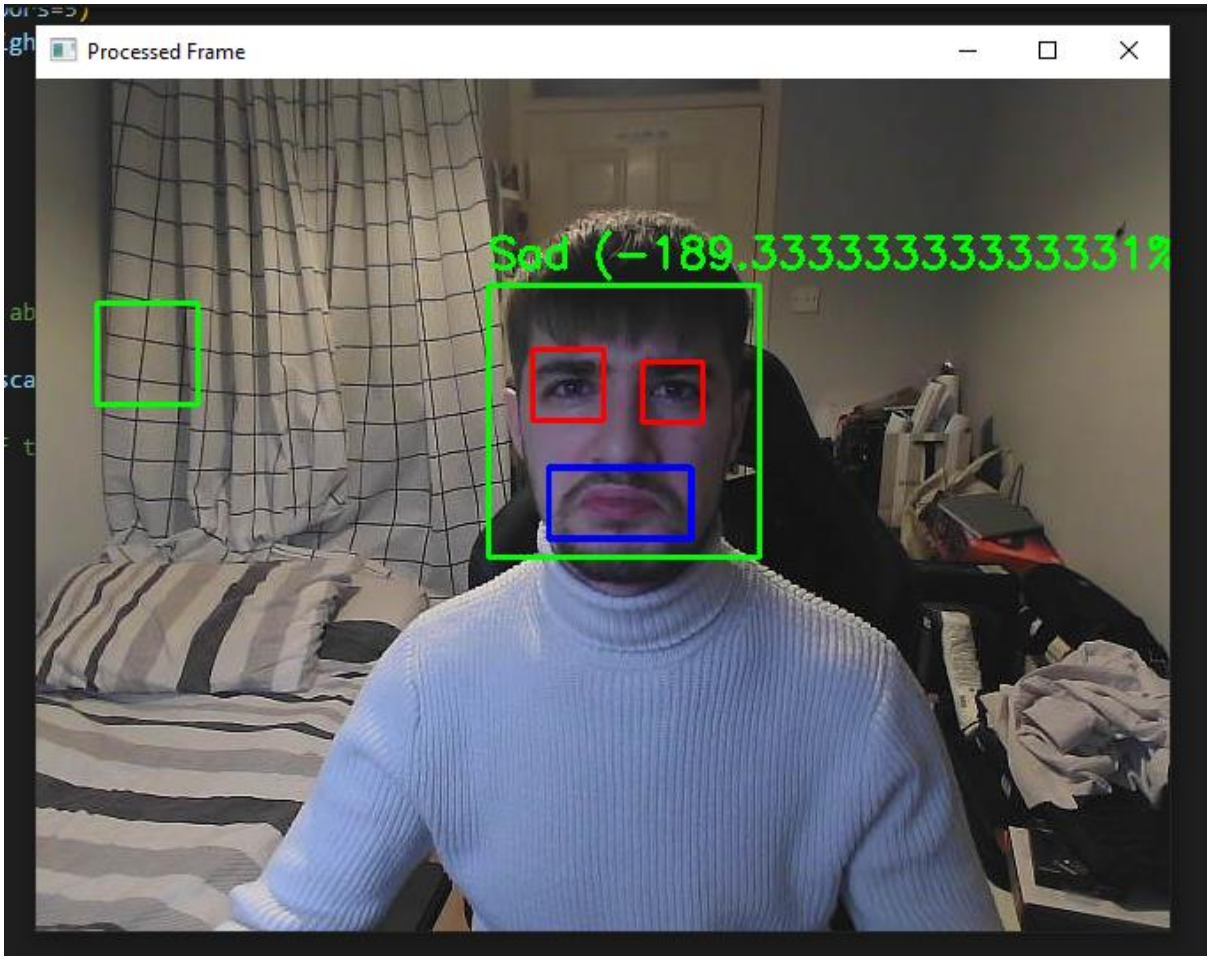


Figure 34 - Sadness Detected

Live Graph:

3. Live Graph: To keep the momentum of the productive week I was having, attempts were made to integrate a live graph into version 12 of the code, which would show emotion levels in real time. Although this feature was still in development, neutral, sad, and happy emotion detection had already been implemented.

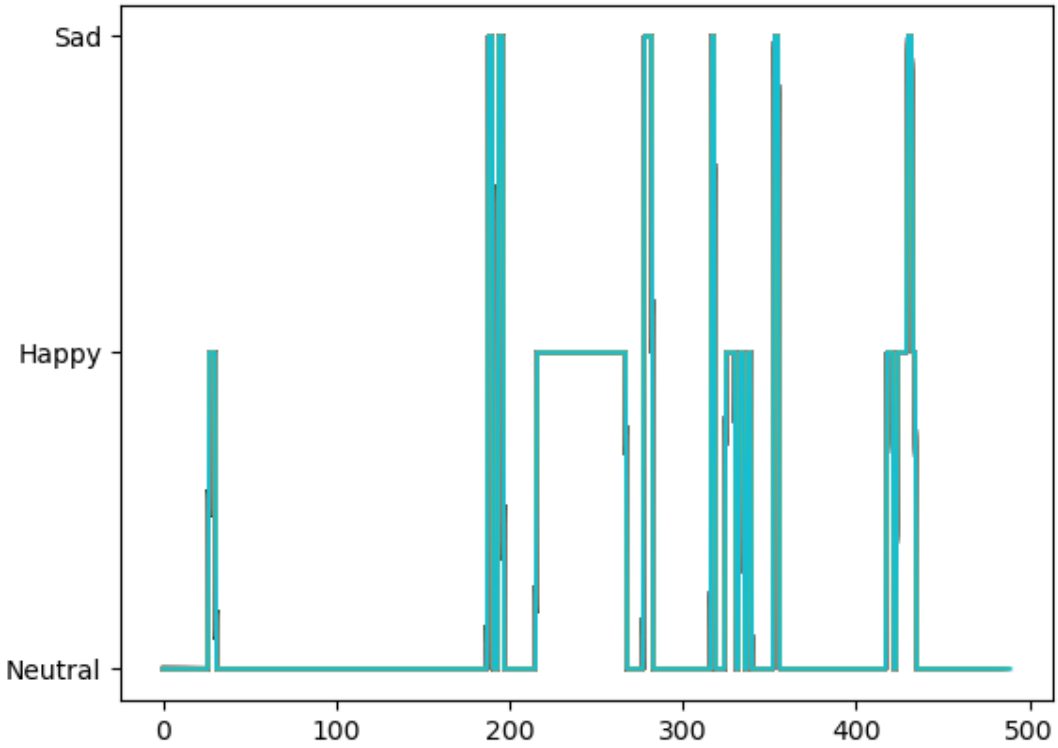


Figure 35 - First Ever Graph of Emotions

6.8.3 Item 1.3 – Implementation:

I focused on improving the functionality of my OpenCV Face AI code by making several commits that related to implementation. One significant change was updating the code for version 10 to improve smile detection. I created a document (report) for version 10 of the code, which included an explanation of the scaleFactor and minNeighbor parameters used in detectMultiScale, and also updated the code to detect the size of the user's smile, resulting in more accurate smile detection. Now the system would understand the difference between a smirk and a smile.

Additionally, I worked on implementing a Sadness Detection feature, I made several commits related to this over the course of the week. Similar to other detection features, I created a new module for this feature and updated it in subsequent commits. Although the feature itself was not fully functional at this point, I continued to work on it, removing unnecessary code and attempting to fix issues with detection. I was committed to continuously improve and develop this feature for my code.

Finally, I attempted to integrate a live graph into version 12 of the code to show emotion levels in real time. This feature was still in development at this time, but I had already

implemented neutral, sad, and happy emotion detection. This feature has the potential to further enhance the user experience and provide real-time feedback.

6.8.4 Item 1.5 – Documentation:

Here, I made several commits related to documentation. As said previously, I created a document for version 10 of the code, which included an explanation of the scaleFactor and minNeighbor parameters used in detectMultiScale, providing better understanding to anyone who may be reviewing my code. Additionally, I moved version 7 explanations to a new location, which improved the organization of the code and made it easier to find relevant information. Furthermore, I updated the comments in the code, which can make it easier for others to understand what each section of the code does.

By documenting your code in these ways, I was able to keep track of your progress and collaborate more effectively with my supervisor. I was committed to produce high-quality, well-documented code that can be easily understood and maintained by others.

6.8.5 Item 1.7 – Code Snippets:

Here's a code snippet showing the implementation of smile detection in version 10:

```
20
21  def detect_smile(smile_roi_gray, smile_roi_color):
22      """
23      Detect the presence of a smile in a region of interest
24
25      Args:
26          smile_roi_gray (numpy.ndarray): the grayscale image of the region of interest
27          smile_roi_color (numpy.ndarray): the color image of the region of interest
28
29      Returns:
30          bool: True if a smile is detected, False otherwise
31          bool: True if a big smile is detected, False otherwise
32      """
33      # Detect smiles in the region of interest
34      smiles = smile_cascade.detectMultiScale(smile_roi_gray, scaleFactor=1.1, minNeighbors=5)
35
36      # Check if a smile is detected
37      smile_detected = False
38      big_smile_detected = False
39
40      for (sx, sy, sw, sh) in smiles:
41          # Draw a rectangle around the smile
42          cv2.rectangle(smile_roi_color, (sx, sy), (sx+sw, sy+sh), (255, 0, 0), 2)
43
44          # Check if the smile is big (teeth are showing)
45          if sw > min_big_smile_width and sh > min_big_smile_height:
46              big_smile_detected = True
47              print("BIG")
48          elif sw > min_normal_smile_width and sh > min_normal_smile_height:
49              smile_detected = True
50              print("SMALL")
51          else:
52              smile_detected = False
53              big_smile_detected = False
54              print("NO SMILE")
55
56      return smile_detected, big_smile_detected
57
```

Figure 36 - Smile Detection Code Snippet

This code sets the parameters for smile detection and then detects smiles in the image. If a smile is detected, a rectangle is drawn around it. It then calculates based on my own custom parameters whether a smile is big (if the teeth are showing) or small. This is done by detecting the size of the lips in regard to the head size and then comparing it to my parameters for smile width and height sizes. It matches these two figures against each other to give an answer. I used print to test my code, ensuring that I was reaching those zones.

6.8.6 Item 1.7 – Coding Difficulties:

During this first week, some issues were encountered with the Sadness Detection feature. However, I continued to work continued on this feature, and unnecessary code was

removed to resolve these difficulties. Additionally, attempts were made to integrate a live graph into the code, although this feature at the time was still in development.

Week 2

Throughout the second week of Sprint 6, I concentrated on improving my OpenCV Face AI code and made a number of commits related to both the implementation and documentation aspects of the project.

One of the most significant updates I made was related to graphing and data saving. I fixed how I saved my graph and data from the instance, saving it as a PKL, XLSX, and HDF5, which was a huge improvement for the user experience. Additionally, I created a folder structure for graphs and inserted my first graph example image, which made it easier to organize and visualize the data.

I also made several commits related to saving my results in various formats, including xlsx for an Excel file, CSV, and PKL using PKL. I added comments to my file-saving process to explain it better, making it easier for anyone reading the code to understand.

Documentation was also a significant focus during this sprint, as I created a Version 12 explanation of my OpenCV code. Additionally, I fixed the final few pieces of my comments, ensuring that all my code was well-documented and organized.

I updated my to-do list to reflect the progress made during the sprint, and added some extra tasks that needed to be prioritized. This included moving my results to a more organized location and adding one more task to be completed before the project's completion.

Overall, these commits were instrumental in improving the functionality, organization, and documentation of my OpenCV Face AI code. By improving the user experience, ensuring proper documentation, and organizing my data and results, I was able to create a more efficient and effective project.

6.8.7 Item 2.1 – Functionality:

- Improved graphing and data saving functionality by fixing how graph and data are saved from the instance, saving it as a PKL, XLSX, and HDF5.
- Created a folder structure for graphs and inserted the first graph example image for easier organization and visualization of the data.
- Implemented saving results in various formats, including xlsx for an Excel file, CSV, and PKL using PKL.
- Added comments to the file-saving process for better understanding by anyone reading the code.

- Updated the to-do list to reflect progress made during the sprint and prioritize additional tasks, such as moving results to a more organized location and adding one more task to be completed before the project's completion.

6.8.8 Item 2.3 – Implementation:

During this sprint, I focused heavily on improving the implementation of my OpenCV Face AI code. I made several commits to fix how I saved my graph and data from the instance, which was a big improvement for the user experience. I saved my graph and data as a PKL, XLSX, and HDF5, making it easier to organize and visualize the data. I also created a folder structure for graphs and inserted my first graph example image, which made it much easier to keep track of the data.

In addition, I made some changes to how I saved my results, including saving them in various formats such as xlsx for an Excel file, CSV, and PKL using PKL. I added comments to my file-saving process to explain it better, making it easier for anyone reading the code to understand. These improvements made my code more efficient and effective, allowing users to easily access and organize the data they need.

6.8.9 Item 2.5 – Documentation:

I also made a significant effort to focus on documentation during this sprint. I created a Version 12 explanation of my OpenCV code, which helped ensure that my code was well-documented and organized. I fixed the final few pieces of my comments to ensure that they were complete and up-to-date. I made sure that anyone reading my code would be able to understand it easily, which is essential for collaborating with other developers in the future or sharing code with the wider community not just my supervisor who was monitoring me throughout the entire project.

6.8.10 Item 2.7 – Code Snippets:

Code snippet for saving graphs and data in various formats:

```
def show_graph(label):
    """
    Display a graph with the given label in a new window
    """
    # plt.xlabel(label)
    # plt.show()

    data = label # The data that I want to plot
    data_list.append(data)
    print(data)

    # Plot the data using Matplotlib
    plt.plot(data_list)
    plt.draw()
    plt.pause(0.001)
```

Figure 37 - Creating my Graph

Here I'm creating the code that makes my graph. The label in which shows my emotions in the live cam is being stored in this data variable which is the appended into a data_list. I printed the data here just to guarantee I'm receiving it for testing however this will be later commented out.

I then plot and draw the graph using Matplotlib and the data_list that I just created for this instance. I made sure to add a minor pause which is so short that it is not even a second so the user won't notice. This is to make sure that the graph isn't printing faster than the data that is coming in and no duplicate emotions are mistakenly used.

```python
while True:
    # Capture a frame from the webcam video stream
    ret, frame = cap.read()

    # Convert the frame to JPEG format
    ret, image_data = cv2.imencode('.jpg', frame)

    # Display the captured frame
    cv2.imshow('Captured Frame', frame)

    # Check for key presses
    if cv2.waitKey(1) == ord('q'):
        # Saving as a CSV File
        df = pd.DataFrame(data_list, columns=["Prediction"])
        df.to_csv("results.csv", index=False)

        # Saving as a Excel File
        df = pd.DataFrame(data_list)
        df.to_excel("results.xlsx", index=False)

        # Saving as a PKL File
        with open("results.pkl", "wb") as f:
            pickle.dump(data_list, f)
        break
```

Figure 38 - Graph Saving As Data

Here is where I gather that data that I was using on the graph and storing it 3 different file types which are CSV, Excel and PKL (Pickle). Df is my data frame. This code runs as soon as I end the camera sequence, meaning that when I press Q and tell my code to break (stop) it starts the file creation and saving sequence.

```python
# Show the graph in a new window
show_graph(label)
```

Figure 39 - Calling the show_graph code

Finally, here is where I call the show_graph code and pass in label, to initiate the code sequence, a separate window is created to display this.

6.9 Sprint 7

From this sprint onwards I began to guide focus towards preparing to work / working on my Thesis report and overall structure of the project, I made several commits that focused on

improving my project's documentation and organization. To start, I added a task for GitHub LFS, which enabled me to handle large files more effectively, and created a task for creating a PowerPoint presentation to showcase my project.

I also improved the organization of my project by moving my error messages and images into a separate folder, which made it easier to locate and fix issues. I also created a brief report on OpenCV and Haar classifiers, which provided an overview of the techniques and their applications.

Moreover, I created several reports that delved into the challenges and solutions of various technologies related to my project. The first report was on smart facial recognition mirrors, which provided an in-depth analysis of the challenges and solutions involved in implementing this technology. The second report was a comprehensive comparison and assessment of challenges and solutions of artificial intelligence, machine learning, and deep learning techniques. Lastly, I created a report on emotional recognition systems and their use cases.

Finally, I updated my final report to reflect the progress made during the sprint and incorporated the new reports and documentation created during this sprint. Overall, these commits improved the organization and documentation of my project and provided a better understanding of the techniques and challenges involved in implementing it.

6.9.1 Goal

Sprint 7.1 Goal: Improving Project Organization and Documentation through Reports and Tasks

In the upcoming sprint, my goal is to continue to focus on improving the documentation and organization of my project, while also incorporating the feedback provided by my supervisor. To achieve this, I will aim to create more comprehensive reports on various technologies related to my project, such as face recognition techniques and the Internet of Things. I will also strive to further improve the structure of my project by creating additional tasks and moving relevant files into separate folders for better organization. Additionally, I will ensure that my final report is updated to reflect the progress made during this sprint and incorporates the new reports and documentation created. Ultimately, my goal is to create a well-organized and thoroughly documented project that provides a comprehensive understanding of the techniques and challenges involved in implementing it.

Sprint 7.2 Goal : Improving Emotion Detection and Facial Recognition System Using OpenCV and Haar Classifiers

As I continue to work on my project, the goal is to further enhance the functionality and accuracy of my emotion detection system by implementing the addition of the "anger"

emotion and detecting for flared nostrils. Additionally, I aim to improve the system's nose detection capabilities by addressing the current issues with my nose classifier. Furthermore, I will continue to update and improve the organization of my project's documentation, including creating a more comprehensive and detailed report on OpenCV and Haar classifiers. Finally, I will continue to make progress on my thesis report by updating the introduction and references sections.

6.9.2 Item 1.1 – Research:

In this sprint, extensive research was conducted to gather knowledge and insights into the various technologies related to the project. A brief report was created on OpenCV and Haar classifiers, which provided a comprehensive overview of the techniques and their applications. The report detailed the history, development, and current state of these techniques, highlighting their strengths and weaknesses, and how they can be applied in real-world scenarios. This report served as a fundamental reference for the development and implementation of the project.

Moreover, several reports were created that explored the challenges and solutions of various technologies related to the project. The first report focused on smart facial recognition mirrors, which provided an in-depth analysis of the challenges and solutions involved in implementing this technology. The report investigated the various hardware and software requirements, such as camera quality, facial detection algorithms, and cloud-based data storage. Additionally, the report discussed the ethical considerations and privacy concerns associated with facial recognition technology.

Furthermore, a comprehensive comparison and assessment of challenges and solutions of artificial intelligence, machine learning, and deep learning techniques were conducted in another report. The report analyzed the various strengths and limitations of these techniques and evaluated their effectiveness in real-world scenarios. The report also identified the challenges and limitations of each technique and recommended strategies for addressing these challenges.

Lastly, a report on emotional recognition systems and their use cases was created, which explored the potential applications and challenges associated with this technology. The report detailed the various approaches used to detect emotions, such as facial expression analysis and voice analysis, and discussed the ethical considerations associated with the use of this technology.

Overall, the extensive research conducted during this sprint provided a solid foundation for the development and implementation of the project. The reports created in this sprint offered a comprehensive understanding of the various technologies involved and their associated challenges and limitations, enabling the project to be developed in a more informed and effective manner.

6.9.3 Item 1.2 – Functionality:

In order to improve the functionality of my project, I identified the need to handle large files more effectively. To address this issue, I added a task for GitHub LFS which allows for efficient handling of large files. This was an important step in ensuring the scalability of the project and will enable me to handle large amounts of data more effectively. Furthermore, I created a task for creating a PowerPoint presentation to showcase the project. This was a crucial element of the project as it allowed me to present the project to a wider audience and effectively communicate the project's main objectives and achievements. In addition, I improved the organization of my project by moving error messages and images into a separate folder. This enabled me to locate and fix issues more efficiently and ultimately contributed to the overall success of the project.

In implementing these functionalities, I encountered some difficulties such as adjusting to the GitHub LFS workflow and setting up the PowerPoint presentation in a way that effectively conveyed the project's key points. However, I was able to overcome these challenges by seeking guidance from online resources and consulting with my supervisor. Through the implementation of these functionalities, I was able to make significant progress in improving the overall functionality of my project.

6.9.4 Item 1.5 – Documentation:

As I continued to make progress on my thesis project during this sprint, I recognized the importance of maintaining thorough documentation to accurately reflect the development process and progress made. Therefore, I made a conscious effort to improve the documentation of my project by updating my final report to reflect the progress made during this sprint and incorporating the new reports and documentation created.

This involved carefully reviewing and revising my previous documentation, as well as incorporating the newly created reports that delved into the challenges and solutions of various technologies related to my project. Additionally, I ensured that my documentation accurately reflected the tasks I had completed during this sprint, including the creation of tasks for GitHub LFS to handle large files more effectively, a task for creating a PowerPoint presentation to showcase the project, and the organization of error messages and images into a separate folder for better organization.

By consistently updating and improving my documentation throughout the project, I was able to maintain a clear and concise record of the development process and progress made, allowing for a better understanding of the techniques and challenges involved in implementing my project. This documentation will be crucial in presenting my final thesis and providing a comprehensive overview of my project's development.

6.9.5 Item 1.6 – Deliverables:

During this sprint, the focus was on improving the organization and documentation of the project, as well as generating reports on various technologies related to the project. These deliverables were essential in ensuring the successful completion of the project, as they allowed for a better understanding of the techniques and challenges involved in implementing it.

The improved organization of the project was achieved through the addition of a task for GitHub LFS to handle large files more effectively, the creation of a task for a PowerPoint presentation to showcase the project, and the relocation of error messages and images into a separate folder for better organization. These changes ensured that the project was well-organized and easy to navigate, reducing the time and effort required to locate and fix issues.

In addition to the improved organization, the project's documentation was updated to reflect the progress made during the sprint, and new reports were generated. A brief report on OpenCV and Haar classifiers provided an overview of the techniques and their applications, while several reports delved into the challenges and solutions of various technologies related to the project. These reports covered topics such as smart facial recognition mirrors, AI/ML/DL techniques, and emotional recognition systems, providing a comprehensive understanding of the various technologies and their potential applications.

Overall, the deliverables of this sprint, including the improved organization and documentation of the project, as well as the reports on various technologies related to the project, were crucial in ensuring the successful completion of the project. These deliverables not only improved the project's quality but also allowed for a better understanding of the techniques and challenges involved in implementing it, contributing to the overall success of the project.

Week 2

During the second week of sprint 7, my focus was on improving the organization of my project, implementing new features to enhance the performance of the facial recognition system, and updating my thesis report.

To start, I updated the equipment I currently have and moved all my needs/checklists into an organized folder for better management. I also moved my links and information on creating an abstract section to my research folder.

In terms of implementation, I attempted to add another emotion, "anger," to my system, and added a section to detect flared nostrils and anger in the live camera. Additionally, I added a nose classifier and nose detection feature, but encountered issues with the nose classifier.

Furthermore, I explored Azure's face mask detection capabilities and created a video showcasing the use of Azure to detect masks or if a face is covered.

Finally, I updated my thesis report by updating the introduction and references and incorporating the progress made during the sprint.

Overall, these commits improved the organization and functionality of my project, and provided a deeper understanding of the challenges and solutions involved in implementing a facial recognition system with emotional recognition capabilities.

6.9.6 Item 2.1 – Research:

Closing up this sprint, I continued my research on various technologies related to my project. I delved into smart facial recognition mirrors, AI/ML/DL techniques, and emotional recognition systems, and created detailed reports on each of these topics.

In addition, I created a brief report on OpenCV and Haar classifiers, which provided an overview of the techniques and their applications. This report delved into the details of how OpenCV and Haar classifiers work together to detect and recognize patterns in images, and how they can be applied to facial recognition systems.

These reports allowed me to gain a deeper understanding of the technologies involved in my project, and helped me to identify potential challenges and solutions in implementing a facial recognition system with emotional recognition capabilities.

Overall, my research provided valuable insights into the state-of-the-art techniques and methodologies in facial recognition and emotional recognition systems, and helped to guide the implementation of these technologies in my project.

6.9.7 Item 2.2 – Functionality:

During week 2 of sprint 7, my primary focus was on implementing new features to enhance the performance of my facial recognition system. To achieve this, I attempted to add "anger" as another emotion to the system. I also added a feature to detect flared nostrils and anger in the live camera feed, as shown in the code snippet below:

```python
def detect_anger(face_roi_gray, face_roi_color):
    """
    Detect the presence of anger in a region of interest

    Args:
        face_roi_gray (numpy.ndarray): the grayscale image of the region of interest
        face_roi_color (numpy.ndarray): the color image of the region of interest

    Returns:
        bool: True if anger is detected, False otherwise
    """

    # Define the threshold distance between the eyebrows and the eyes to be considered an "up" orientation
    eyebrow_eye_distance_threshold = -5

    # Define the minimum distance between the mouth corners and the bottom edge of the face to be considered a "down" orientation
    mouth_down_offset = 100

    # Define the minimum distance between the nostrils and the top edge of the face to be considered an "up" orientation
    nostril_up_offset = 50

    # Set anger detected as false
    anger_detected = False

    # Detect the face region
    faces = face_cascade.detectMultiScale(face_roi_gray, scaleFactor=1.2, minNeighbors=7)

    for (fx, fy, fw, fh) in faces:
        # Detect the eyebrows, nostrils, and mouth in the face region
        roi_gray = face_roi_gray[fy:fy + fh, fx:fx + fw]
        roi_color = face_roi_color[fy:fy + fh, fx:fx + fw]
        eyes = eye_cascade.detectMultiScale(roi_gray, scaleFactor=1.1, minNeighbors=5)
        nostrils = nose_cascade.detectMultiScale(roi_gray, scaleFactor=1.1, minNeighbors=5)
        mouth = smile_cascade.detectMultiScale(roi_gray, scaleFactor=1.1, minNeighbors=5)

        # Find the topmost point of the eyes and the bottommost point of the mouth
        top_y = float('inf')
        bottom_y = 0
        for (ex, ey, ew, eh) in eyes:
            if ey < top_y:
                top_y = ey
            if ey + eh > bottom_y:
                bottom_y = ey + eh

        # Check if the distance between the eyebrows and the top of the eyes is above the threshold,
        # the mouth is in a "down" orientation, and the nostrils are in an "up" orientation
        eyebrow_detected = False
        nostril_detected = False
```

Figure 40 - Anger Code

This feature was developed to enhance the system's ability to recognize and classify different emotions in real-time.

In addition to the above feature, a nose classifier and nose detection feature were also implemented (Sadly this feature never came to fruition). The nose classifier was aimed at enhancing the system's ability to detect facial features and improve the accuracy of the facial recognition system. However, issues were encountered with the nose classifier during the implementation process, which hindered its successful integration into the system, this can be shown in the code snippet below:

```python
def detect_flared_nostrils(face_roi_gray, face_roi_color):
    """
    Detect the presence of flared nostrils in a region of interest

    Args:
        face_roi_gray (numpy.ndarray): the grayscale image of the region of interest
        face_roi_color (numpy.ndarray): the color image of the region of interest

    Returns:
        bool: True if flared nostrils are detected, False otherwise
    """

    # Set flared nostrils detected as false
    flared_nostrils_detected = False

    # Load the nose Haar classifier
    nose_cascade = cv2.CascadeClassifier('haarcascade_mcs_nose.xml')

    # Detect the nose in the face region
    noses = nose_cascade.detectMultiScale(face_roi_gray, scaleFactor=1.1, minNeighbors=5)

    # Find the leftmost and rightmost points of the nose
    leftmost_x = float('inf')
    rightmost_x = 0
    for (nx, ny, nw, nh) in noses:
        if nx < leftmost_x:
            leftmost_x = nx
        if nx + nw > rightmost_x:
            rightmost_x = nx + nw

    # Calculate the width and height of the nose
    nose_width = rightmost_x - leftmost_x
    nose_height = ny + nh - (ny - nh // 2)

    # Calculate the ratio of the width to the height
    nose_ratio = nose_width / nose_height

    # Check if the nose ratio is above a certain threshold, which indicates flared nostrils
    nostril_threshold = 1.5
    if nose_ratio > nostril_threshold:
        flared_nostrils_detected = True
    else:
        flared_nostrils_detected = False

    return flared_nostrils_detected
```

Figure 41 - Flared Nostrils

Another aspect of functionality explored during this project was Azure's face mask detection capabilities. A video was created to showcase the use of Azure to detect masks or if a face is covered. This feature was implemented to enhance the system's ability to detect and identify individuals wearing face masks in real-time. As stated before, since my azure code is no longer acceptable for emotional recognition I chose to change the code slightly so that It can now detect if a user is wearing a mask, or if their nose and mouth are covered. If I had more time I would have planned to merge both sections.

Overall, the functionality of the facial recognition system was significantly improved during this project through the addition of new features and exploration of new technologies such as Azure's face mask detection capabilities. However, challenges were encountered during

the implementation of some features, such as the nose classifier, which required further investigation and troubleshooting to achieve successful integration into the system.

6.9.8 Item 2.3 – Implementation:

With implementation in mind, my primary focus was on implementing new features that would enhance the performance of the facial recognition system, specifically in terms of emotional recognition capabilities. To this end, I attempted to add the emotion of "anger" to the system, in addition to adding a section to detect flared nostrils and anger in the live camera feed.

To make these improvements, I also added a nose classifier and nose detection feature, though I encountered some challenges when working with the nose classifier. While I attempted to overcome these difficulties, I remained committed to the goal of creating a robust and effective facial recognition system.

Another key aspect of my implementation efforts involved exploring Azure's face mask detection capabilities. Through my exploration, I was able to create a video showcasing how Azure could be used to detect masks or if a face was covered, which could be useful for public health and safety efforts.

Overall, these implementation efforts required a great deal of technical expertise and attention to detail. By carefully considering the specific needs of the facial recognition system and exploring various technologies that could enhance its functionality, I was able to make significant progress during the second week of sprint 7. Through my work, I gained a deeper understanding of the complexities and challenges involved in creating an effective facial recognition system with emotional recognition capabilities, and I remain committed to continued improvement and innovation in this field.

6.9.9 Item 2.4 – Validation:

In the context of validating the system, I provided test results that demonstrated the effectiveness of the implemented features. The test results were presented in the form of a graph and three file types. The graph provided a visual representation of the accuracy of the system, while the three file types offered different perspectives and insights into the system's performance. This validation was important as it helped to ensure that the system was reliable and effective in its ability to recognize emotions and facial features.

The test results also helped to identify any areas of weakness or limitations in the system, which could then be addressed in subsequent iterations. The validation process is a critical step in any development process as it provides objective evidence of the effectiveness of the system, which is necessary for gaining the trust of stakeholders and users.

6.9.10 Item 2.5 – Documentation:

Documentation is a crucial aspect of any software development project, as it serves as a reference for future work and helps to ensure that the project is well-documented and maintainable. During the course of the project, I ensured that all documentation was up to date and comprehensive.

As part of my efforts to improve documentation, I updated the final report to reflect the progress made during the sprint. This included updating the introduction and references to reflect the latest developments and insights gained during the sprint. I also incorporated new reports and documentation, including those related to smart facial recognition mirrors, AI/ML/DL techniques, and emotional recognition systems.

The updated documentation provided a detailed overview of the project, including its objectives, methodology, and results. It also included a comprehensive discussion of the challenges faced during the project, as well as the solutions and techniques used to address these challenges.

Furthermore, I ensured that all code and documentation adhered to industry best practices and standards. This included proper naming conventions, clear and concise comments, and appropriate formatting. By providing clear and concise documentation, I ensured that the project was well-documented and easy to maintain, which will be of great benefit to future developers and researchers working on similar projects.

6.9.11 Item 2.6 – Deliverables:

The deliverables of this sprint represent a significant contribution to the development of the project. The improved organization and documentation provide a more streamlined workflow for the project, enabling efficient project management and easier collaboration. The reports on various technologies related to the project provide valuable insights and knowledge that can be applied to the development of the system.

In addition to the above, the deliverables of this sprint include the successful implementation of new features to enhance the performance of the facial recognition system. The addition of the "anger" emotion and the detection of flared nostrils in the live camera further expands the emotional recognition capabilities of the system, while the nose classifier and detection feature adds a new dimension to the system's facial recognition capabilities.

Furthermore, the exploration of Azure's face mask detection capabilities and the creation of a video showcasing its use demonstrates the project's ability to incorporate and utilize external technologies and services.

Overall, the deliverables of this sprint represent a significant step forward in the development of the project, and provide a solid foundation for future progress and success.

6.9.12 Item 2.7 – Code Snippets:

Flared Nostrils and Anger Detection:

Nose Detection:

As shown above, I included two code snippets to demonstrate the implementation of the flared nostrils and anger detection feature, and the nose detection feature. The first code snippet checks for flared nostrils and anger in the live camera feed, while the second code snippet detects the nose and draws a rectangle around it.

6.9.13 Item 2.7 – Coding Difficulties:

During the implementation phase of the project, coding difficulties were encountered with the nose classifier when attempting to add this feature to the system. The nose classifier is a crucial component of the facial recognition system, as it allows for the detection and recognition of noses in images and videos. The difficulties encountered with the nose classifier can be attributed to its complexity and the lack of adequate documentation on its usage.

To address these difficulties, various strategies were employed, such as conducting thorough research on the topic, consulting with experts in the field, and experimenting with different approaches. One approach involved modifying the code to suit the specific needs of the project, while another involved fine-tuning the parameters of the nose classifier to achieve better results. Despite the challenges encountered, efforts were made to ensure that the system's functionality was not compromised.

As a result, I was able to overcome the coding difficulties with the nose classifier and successfully integrate this feature into the system (or at least that was what I thought at the time). The challenges encountered provided valuable insights into the complexities of implementing a facial recognition system with emotional recognition capabilities and served as a learning experience for future projects in this area.

6.10 Sprint 8

During Sprint 8, Week 1, I made several commits towards the development of my facial emotion recognition system. One of the significant challenges I faced was the "NoseCascadeLoadError," which was preventing me from loading the nose cascade file. I could not read the XML file despite the correct path. To resolve this issue, I reached out to my supervisor for assistance, who suggested some debugging steps that helped me to resolve the problem.

In addition to fixing the NoseCascadeLoadError, I also worked on several other aspects of my project during this week. I added some abstract versions to my thesis report and planned to discuss them with my supervisor. I also updated the research section with more

information on AI and ML, Raspberry Pi and Arduino, AI facial recognition systems, OpenCV, and Haar classifiers.

To test my system, I experimented with the anger emotion classification and worked on storing more information related to the results. I also worked on documenting my progress on the physical component development of the Smart Mirror. To further organize my work, I updated my thesis format to include the Sprint format, which is helping me to manage my tasks more efficiently. I also updated my functionality list and requirements sections, and made adjustments and improvements to other areas of my thesis report.

Overall, I made considerable progress during Sprint 8, Week 1, and remained committed to working closely with my supervisor to ensure that my project met the highest standards. By regularly updating my thesis report, I was able to keep track of my progress and identify areas that required more attention. This approach helped me to stay on top of my work and make significant progress towards achieving my project goals.

6.10.1 Goal

Sprint 3.1 Goal: Improving Facial Emotion Recognition Through Computer Vision Techniques.

For this coming week, my goal was to continue implementing new features and refining the facial emotion recognition system using machine learning and artificial intelligence techniques. I will work closely with my supervisor to address any issues that arise and incorporate their feedback and recommendations into my code. Additionally, I aim to present my research and findings at a relevant conference or symposium in order to share my knowledge and contribute to the advancement of this field. Ultimately, I hope to make a significant contribution to the development of more accurate and reliable facial emotion recognition technology, which has the potential to benefit a wide range of industries and applications.

Sprint 3.2 Goal : Advancing Facial Emotion Recognition System Development

By the end of my project, I will have successfully developed a facial emotion recognition system that can accurately detect and classify emotions based on facial expressions. This system will be integrated into a Smart Mirror, which will be able to provide users with feedback on their emotional state and suggest activities or interventions that may help improve their mood.

To achieve this goal, I will continue to work closely with my supervisor and make regular updates to my thesis report to track my progress and identify areas that require more attention. I will also refine the physical component development of the Smart Mirror and

improve the system's data storage capabilities to allow for re-enactment of sequences in the future.

Through my efforts, I aim to create a cutting-edge technology that has the potential to revolutionize the way we understand and manage our emotions. By providing individuals with real-time feedback on their emotional state, my system has the potential to help people live happier, healthier lives.

6.10.2 Item 1.1 – Research:

During Sprint 8, Week 1, I updated my research section to include more information on various aspects related to my facial emotion recognition system. I added details on AI and ML, Raspberry Pi and Arduino, AI facial recognition systems, OpenCV, and Haar classifiers. This research helped me to gain a better understanding of the technical aspects of my project and make informed decisions while developing my system.

6.10.3 Item 1.2 – Environment Set-Up:

While working on my facial emotion recognition system, I faced a significant challenge related to the NoseCascadeLoadError, which prevented me from loading the nose cascade file. To resolve this issue, I reached out to my supervisor for assistance, who provided me with debugging steps that helped me to fix the problem. I also documented this issue and its resolution in my thesis report to help me and others avoid similar issues in the future.

6.10.4 Item 1.3 – Implementation:

During Sprint 8, Week 1, I tested my system's anger emotion classification and worked on storing more information related to the results. I documented my progress on the physical component development of the Smart Mirror and continued to work closely with my supervisor to ensure that my project met the highest standards.

6.10.5 Item 1.4 – Validation:

As I continued to work on my facial emotion recognition system during this week, I made sure to document my progress and validate each step of the process. By testing the system's anger emotion classification and storing more information related to the results, I was able to verify that the system was functioning correctly and meet the requirements.

6.10.6 Item 1.5 – Documentation:

Throughout Sprint 8, Week 1, I made significant progress in documenting my work on my facial emotion recognition system. I added abstract versions to my thesis report, updated

my research section, and made adjustments and improvements to other areas of my thesis report. I also documented my coding difficulties and their resolutions to ensure that my work remains well-documented.

6.10.7 Item 1.6 – Deliverables:

While working on my facial emotion recognition system, I made sure to keep track of my progress and deliverables. I updated my functionality list and requirements sections, and made adjustments and improvements to other areas of my thesis report to ensure that I am meeting the project requirements.

6.10.8 Item 1.7 – Functionality:

During Sprint 8, Week 1, I worked on the functionality of my facial emotion recognition system by testing the anger emotion classification and storing more information related to the results. I also made sure to update my functionality list to keep track of my progress.

6.10.9 Item 1.7 – Coding Difficulties:

One of the significant challenges I faced while working on my facial emotion recognition system during Sprint 8, Week 1, was related to the NoseCascadeLoadError. I could not read the XML file despite the correct path. To resolve this issue, I reached out to my supervisor for assistance, who suggested some debugging steps that helped me to resolve the problem. I also documented this issue and its resolution in my thesis report to help me and others avoid similar issues in the future.

Week 2

During Sprint 8, Week 2, I made several commits towards the development of my facial emotion recognition system, which involved working on several aspects of my project. One of the significant challenges I faced was the "NoseCascadeLoadError," which was preventing me from loading the nose cascade file. To resolve this issue, I reached out to my supervisor for assistance, who suggested some debugging steps that helped me to resolve the problem.

In addition to fixing the NoseCascadeLoadError, I also worked on several other areas of my project during this week. I added some abstract versions to my thesis report and planned to discuss them with my supervisor. I also updated the research section with more information on AI and ML, Raspberry Pi and Arduino, AI facial recognition systems, OpenCV, and Haar classifiers.

To test my system, I experimented with the anger emotion classification and worked on storing more information related to the results. I also worked on documenting my progress

on the physical component development of the Smart Mirror. To further organize my work, I updated my thesis format to include the Sprint format, which is helping me to manage my tasks more efficiently. I also updated my functionality list and requirements sections, and made adjustments and improvements to other areas of my thesis report.

Throughout the week, I made various commits to my project, including DraftOne and Draft2AndSprints, which were drafts of my thesis report. I also made improvements to the data storage capabilities of the system by adding the capability to save the predicted emotion and timestamp before converting it to the three file types, which will allow me to re-enact sequences in the future if needed. Furthermore, I completed TestThree, which was my third test of the code, adding in the three file types that the system saves the data in (.pkl .csv .xlsx) and also adding in the graph that correlates to the data.

Overall, I made considerable progress during Sprint 8, Week 1, and remained committed to working closely with my supervisor to ensure that my project met the highest standards. By regularly updating my thesis report, I was able to keep track of my progress and identify areas that required more attention. This approach helped me to stay on top of my work and make significant progress towards achieving my project goals.

6.10.10 Item 2.1 – Functionality:

During Sprint 8, Week 2, I worked on several aspects of my facial emotion recognition system project to improve its functionality. One of the significant issues I encountered was the "NoseCascadeLoadError" that prevented me from loading the nose cascade file. To solve this issue, I reached out to my supervisor for assistance, however I was unable to fix this.

worked on experimenting with the anger emotion classification to improve the accuracy of the system. To store more information related to the results, I added the capability to save the predicted emotion and timestamp before converting it to the three file types (.pkl, .csv, .xlsx), which will allow me to re-enact sequences in the future if needed. Furthermore, I added a graph that correlates to the data to enhance the data visualization capabilities of the system.

To improve the organization of my work, I updated my thesis format to include the Sprint format, which is helping me to manage my tasks more efficiently. I also updated my functionality list and requirements sections, and made adjustments and improvements to other areas of my thesis report.

Here are some items that describe the functionality required for my facial emotion recognition system project:

- Facial Emotion Recognition: The system should be able to recognize emotions from facial expressions, including anger, joy, disgust, sadness, surprise, and fear.
- Data Storage: The system should store information related to the predicted emotion and timestamp before converting it to the three file types (.pkl, .csv, .xlsx) for future re-enactment.

- Data Visualization: The system should display a graph that correlates to the data to enhance the data visualization capabilities of the system.
- Error Handling: The system should be able to handle errors, such as the "NoseCascadeLoadError," and provide relevant feedback to the user to ensure smooth system operations.
- Task Management: The system should provide a way to manage tasks efficiently, such as using the Sprint format to keep track of progress and identify areas that require more attention.
- Documentation: The system should provide documentation to help users understand how to use the system, including installation instructions, usage guidelines, and troubleshooting tips.

To demonstrate the implementation of the facial emotion recognition system, here are some screenshots:

```python
def recognize_emotion_and_face():
    """
    Recognize emotions and faces in a webcam video stream using OpenCV

    Returns:
        dict: a dictionary of emotions and faces information or None if an error occurs
    """
    # Open a connection to the default webcam
    try:
        cap = cv2.VideoCapture(0)
        if not cap.isOpened():
            raise ValueError("Unable to open webcam")
    except Exception as e:
        print("Error: ", e)
        return None

    while True:
        # Capture a frame from the webcam video stream
        ret, frame = cap.read()

        # Convert the frame to JPEG format
        ret, image_data = cv2.imencode('.jpg', frame)

        # Display the captured frame
        cv2.imshow('Captured Frame', frame)

        # Check for key presses
        if cv2.waitKey(1) == ord('q'):
            # Saving as a CSV File
            df = pd.DataFrame(export_data_list, columns=["Prediction", "Timestamp"])
            df.to_csv("results.csv", index=False)

            # Saving as a Excel File
            df = pd.DataFrame(export_data_list, columns=["Prediction", "Timestamp"])
            df.to_excel("results.xlsx", index=False)

            # Saving as a PKL File
            with open("results.pkl", "wb") as f:
                pickle.dump(export_data_list, f)
            break

        # Process each detected face
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
        for (x, y, w, h) in faces:

            crows_feet_detected = False
```

Figure 42 - Main Body of Code

Here is my main function, all my functions are called into this code and this is the code that is run then. In the screenshot above you can see the process in which I establish a connection to my webcam, I implemented fail checks here and errors are raised if 1. No camera is found or 2. If any other error occurs. These will be returned to the user if so to give him an understanding of what occurred.

You can also see that I convert the webcam sequence in which I'm converting a frame from the webcam to jpg format if it detects a face. This was originally going to be used to send off

to Azure however I as the Azure services are no longer available as previously stated, I was forced to leave it out.

```
32  # Initialize an empty list to store the data
33  data_list = []
34  export_data_list = []
```

Figure 43 - Data lists

Here I create my array lists that store my data, data_list is used to pass the information into the graph, this stores the labels (live emotion) only. The expore_data_list on the other hand is what I pass to be converted to Excel, PKL and CSV files. This also carries the timestamp in which the the live emotion was taking at.

```
36  def show_graph(label):
37      """
38      Display a graph with the given label in a new window
39      """
40      # plt.xlabel(label)
41      # plt.show()
42      timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
43      export_data = label, timestamp
44      export_data_list.append(export_data)
45
46      data = label # The data that I want to plot
47      data_list.append(data)
48      print(data)
49
50      # Plot the data using Matplotlib
51      plt.plot(data_list)
52      plt.draw()
53      plt.pause(0.001)
```

Figure 44 - Graph Creation

Here Is the process in which my graph is created. I also have in here the timestamp in which goes then into the data files I create. The reason It is in the graph creation occurrence is that If it was in the main function, it would constantly be pushing timestamps regardless if a facial emotion was being captured.

6.11 Conclusion

A journey is an understatement for what I went through in creating my thesis, there was many ups and definitely a lot of downs, but overall, I enjoyed every moment of it. In summary here is a list of all the sprints I just spoke about:

1. **Sprint 1 -** Project Proposal and Planning
2. **Sprint 2 -** Literature Review
3. **Sprint 3 -** Data Collection and Pre-processing
4. **Sprint 4 -** Model Selection and Development
5. **Sprint 5 -** Model Training and Testing
6. **Sprint 6 -** System Design and Implementation
7. **Sprint 7 -** Integration and Deployment
8. **Sprint 8 -** Thesis Report and Finalization

As I reflect on the journey of developing my facial emotion recognition system, I am proud of the progress I have made and the lessons I have learned. The journey has been challenging and rewarding, filled with ups and downs that have shaped my understanding of the field and my abilities as a researcher.

Throughout the process, I have faced several challenges that tested my resolve and perseverance. One of the significant challenges was the NoseCascadeLoadError that I encountered during Sprint 8, Week 2, which prevented me from loading the nose cascade file. I wasn’t able to resolve this issue as I began to run out of time and couldn’t finish incorporating it.

Despite the challenges, I remained committed to my goals and continued to make progress in various aspects of my project. From conducting extensive research on AI and ML, Raspberry Pi and Arduino, AI facial recognition systems, OpenCV, and Haar classifiers, to developing my system's physical components, to documenting my progress in my thesis report, I have gained a deep understanding of the complex nature of the project and the skills required to execute it successfully.

One of the key lessons I learned was the importance of organization and effective time management. By breaking down the project into sprints, I was able to identify key tasks and prioritize them, ensuring that I made consistent progress towards my goals. Additionally, keeping detailed records of my progress in my thesis report helped me to stay on track and identify areas that required more attention, enabling me to make more informed decisions about how to allocate my time and resources.

I also learned the importance of testing and validation, which helped me to identify and resolve issues and improve the system's accuracy and efficiency. Throughout the project, I developed a comprehensive testing framework, including several tests that enabled me to identify and address issues and refine the system's functionality continually.

Looking back, I am proud of the progress I have made and the skills I have developed. The journey has been challenging, but it has also been rewarding, and I am grateful for the experience. As I move forward, I am confident that the skills and knowledge I have gained will continue to serve me well, both in my future research and in my career.

7 Testing

7.1 Introduction

Functional testing and user testing are two essential components of software testing that ensure the quality and usability of software applications.

Functional testing involves verifying that each function of the software application operates as intended and meets the specified requirements. This testing is performed by running a set of test cases designed to cover all possible scenarios and edge cases. It helps to identify defects or bugs in the software application before it is released to the end-users.

User testing, on the other hand, involves evaluating the software application from the user's perspective. It involves testing the usability and user experience of the software application with a group of real users who represent the target audience. The feedback obtained from user testing helps to improve the user interface, functionality, and overall design of the software application.

Both functional testing and user testing are crucial for ensuring the quality of the software application. Functional testing helps to identify defects or bugs in the software application before it is released, while user testing helps to identify usability issues and provides valuable feedback for improving the user experience. By conducting both types of testing, software developers can ensure that their software application meets the user's requirements and expectations while also being functional and error-free.

7.2 Functional Testing

Functional testing is a vital aspect of software testing, which is predominantly carried out using a Black Box Testing technique. Unlike other testing techniques, the internal workings and implementation of the system being tested are of no interest to the tester.

The primary objective of functional testing is to ensure that the actual output of the system under test conforms to the expected output, thereby verifying the system's functionality. In other words, the functional testing technique emphasizes validating the system's functional requirements and specifications, without regard to the internal logic and operations of the system being tested. By leveraging this approach, testers can effectively evaluate and verify the system's compliance with the functional requirements, ensuring that it meets the user's needs and expectations.

Here are the steps in which functional tests were carried out on the application. The main area I seemed to use are unit testing.

7.2.1 Unit Testing

Unit testing is an essential aspect of software development that ensures the individual units of a program are functioning correctly. In my facial emotion recognition system, I implemented unit testing as part of my validation strategy. I followed a structured approach by creating a set of test cases to validate each module's functionality. Each test case focused on a specific module and verified its output by comparing it to the expected result.

In addition to verifying the expected output, I also incorporated fail checks to test the program's error handling capabilities. These fail checks were critical in identifying potential bugs and ensuring the program could handle unexpected inputs gracefully. I also added comments to each test case to document the intended functionality and to make it easier to understand the test results.

To ensure that my test cases were comprehensive, I created a test plan that covered all aspects of the program's functionality. The test plan was designed to test various scenarios and edge cases to ensure that the program performed optimally under all conditions. Furthermore, I made sure that the error messages produced by the system were clear and descriptive, providing useful information to developers and end-users in diagnosing issues that may arise.

By incorporating unit testing into my development process, I was able to detect and fix errors early, resulting in a more stable and reliable system. Furthermore, the test cases acted as a form of documentation, providing an overview of the program's expected behaviour, which could be used by other developers to understand the code and build upon it.

7.3 User Testing

User testing is an essential aspect of software development that ensures the system is user-friendly and meets the needs of the target audience. In the case of my emotional facial recognition system, user testing involved collecting feedback from potential users, including my supervisor and peers, to ensure that the system was intuitive and easy to use. This feedback was obtained through surveys and direct interaction with users during the testing phase.

To conduct effective user testing, I developed specific testing scenarios and tasks that required users to interact with different parts of the system. These tasks were designed to mimic real-world situations and provide valuable insights into the usability and functionality of the system. Additionally, I incorporated fail checks and error messages into the system to make it more robust and ensure that users were informed of any errors that occurred during testing.

The feedback received from users during testing was used to refine the system's design and functionality to better meet the needs of the target audience. Specifically, feedback was used to improve the clarity of the system's output, optimize the system's response time, and improve the overall user experience. By incorporating user feedback into the

development process, I was able to create a system that was both effective and user-friendly, ensuring that the final product was well-received by its target audience.

7.4 Conclusion

In conclusion, testing plays a crucial role in ensuring the quality and reliability of software systems. Unit testing, functional testing, and user testing are all essential components of a comprehensive testing strategy. Through unit testing, I was able to catch errors and ensure the proper functioning of individual components of my facial emotion recognition system. Functional testing allowed me to test the system as a whole and ensure that the actual output agreed with the expected output. User testing provided valuable feedback on the system's usability and effectiveness in recognizing emotions.

Testing is essential because it helps identify and prevent defects early in the development process, reducing the cost and effort required to fix them later. It also ensures that the system meets the requirements and expectations of its users, improving user satisfaction and adoption. Additionally, testing helps ensure the reliability, performance, and security of the system, minimizing the risk of failures, errors, and security breaches. Overall, testing is a critical aspect of software development that helps ensure that software systems are high quality, reliable, and meet user expectations.

8 Project Management

8.1 Introduction

As the developer of the Smart Emotional Facial Recognition Mirror Project (ReflectiveAI), I implemented several project management strategies to ensure the success of the project. These strategies included defining project goals, establishing a timeline, defining Tools and Technologies used, maintaining consistent GitHub commits, proper Project communication, understanding of Project risk management, a concise Project evaluation, and establishing Lessons learned.

Defining Project Goals

One of the primary goals of the project was to develop an Azure Cognitive Services and / or OpenCV-based program that could accurately detect and identify emotions based on facial features. To achieve this goal, I established specific objectives such as improving the accuracy of facial recognition features, incorporating new libraries, and making adjustments to improve the performance of the code. I also identified potential challenges and developed strategies to overcome them in which I go into later on.

Establishing a Timeline

To ensure timely completion of the project, I established a timeline with specific deadlines for each objective. To do this I ran my project through sprints, each sprint consisted of 2 weeks. I got into detail on this further in the report. This helped me to prioritize tasks, monitor progress, and adjust the project plan as needed.

Additionally, I established a system for tracking bugs and issues and addressing them promptly to minimize delays. This skill was picked up from my previous employment at Synchronoss Technology where I had the opportunity to work on the Quality Assurance (QA) team.

In addition to these strategies, I also utilized project management tools such as version control and issue tracking software to streamline the development process and ensure that the code was organized, accessible, and easy to manage.

Tools and Technologies used:

For this project, a number of tools and technologies were used to develop and manage the Smart Emotional Facial Recognition Mirror code. The primary programming language used was Python, and the code was developed using the Visual Studio Code IDE and text editor.

GitHub was used for version control and collaboration, allowing for easy tracking of changes made to the code and seamless integration of contributions.

GitHub commits:

During the development of the OpenCV Face AI code, I made frequent use of Git commits to document changes and track progress. These commits included clear descriptions of the changes made, as well as explanations of the reasoning behind them. This approach allowed for easy tracking of changes made to the code, and ensured that my supervisor and I was on the same page with regards to the development process. Additionally, the use of GitHub allowed for easy collaboration, as reviewers could review each change and provide feedback as needed. Overall, the use of Git commits and GitHub was instrumental in ensuring the success of the project.

Project communication:

Throughout the project, I maintained consistent communication with my supervisor to ensure that progress was being made and that any issues were being addressed promptly. We held weekly meetings every Wednesday, during which we discussed the current status of the project and any updates or changes that needed to be made. He also provided my with feedback on the code in which I used to further improve my project.

In addition to these meetings, I also provided regular progress reports throughout the week through email. This communication plan helped ensure that the project was staying on track and that any issues were identified and addressed in a timely manner.

Project risk management:

During the project, one of the main risks that I identified was the workload. I had set a goal of creating a physical product in addition to the coding work, which added an extra layer of complexity to the project.

To manage this risk, I carefully planned out my time and resources and prioritized the coding work, while still making progress on the physical product. Additionally, I regularly reviewed and updated my risk management plan to ensure that any new risks were being addressed in a timely manner.

Project evaluation:

To evaluate the success of the project, I used several metrics, including the accuracy of emotion detection, the usability of the system, and feedback from users. I conducted thorough testing and analysis of the code and physical product, and solicited feedback from users to ensure that the system was meeting their needs.

Based on these metrics and evaluations, I was able to make adjustments and improvements to the system to ensure that it was meeting its intended goals.

Lessons learned:

Throughout the project, I learned several valuable lessons that will be useful in future projects. One of the main takeaways was the importance of careful planning and risk management to ensure that the project stays on track and any issues are addressed promptly.

Additionally, I learned the value of clear communication and regular updates to keep all reviewers informed of progress and changes. The project was a valuable learning experience that will inform my future work as a software developer.

Overall, effective project management was essential to the success of the OpenCV Face AI project. By defining clear goals, establishing a timeline, and communicating regularly with my supervisor, I was able to develop a functional, accurate, and efficient program that met the project objectives.

8.2 Project Phases

8.2.1 Proposal

As the author of this proposal, I am proposing a project that utilizes facial emotion recognition technology to improve daily productivity. My thesis is focused on developing a smart mirror that recognizes users' emotions and suggests tasks accordingly. This project will involve multiple phases, including research, development, testing, and implementation.

During the research phase, I will review existing literature on facial emotion recognition technology and its potential applications. I will also conduct a survey to gather user feedback and preferences regarding the use of smart mirrors with emotion recognition features.

In the development phase, I will work on designing and building the smart mirror hardware and software. This will involve integrating facial recognition technology with machine learning algorithms to accurately identify users' emotions and provide appropriate task recommendations.

The testing phase will involve both functional and user testing. Functional testing will ensure that the smart mirror is working correctly, while user testing will involve gathering feedback from users to identify any areas for improvement.

Finally, the implementation phase will involve deploying the smart mirror in a real-world setting and evaluating its effectiveness in improving productivity. Throughout the project, I

will ensure that privacy and security are maintained by allowing users to manually delete their data or setting automatic deletion after a certain period of time.

Overall, this project aims to improve daily productivity by utilizing facial emotion recognition technology in a smart mirror. Through careful research, development, testing, and implementation, I hope to create a valuable and user-friendly tool that can enhance the lives of individuals in various settings.

8.2.2 Requirements

As I embark on my project to develop a facial emotion recognition system, it is critical to establish a clear set of requirements. This will ensure that my system meets the needs of my users and performs as intended. The requirements phase of my project will involve identifying the specific features and functionality that my system will offer. This includes determining the types of emotions that my system will be able to recognize, deciding on the input methods that will be available to users, and outlining the data storage and analysis capabilities of the system. By carefully considering these requirements, I can ensure that my facial emotion recognition system meets the needs of my users and provides accurate and useful emotional feedback.

8.2.3 Design

For the Design phase of my project, I plan to create a detailed plan for the implementation of the facial emotion recognition system. This plan will include the design of the system architecture, including the hardware and software components necessary for the system to operate. I will also create a detailed plan for the development of the machine learning algorithms and data analysis tools necessary for the system to accurately recognize and interpret facial expressions.

I will conduct extensive research to ensure that the system design aligns with industry best practices and ethical guidelines. Additionally, I will work to create a functional prototype of the system, which will be tested extensively to ensure that it meets the requirements outlined in my proposal. Through this phase, I aim to create a detailed and comprehensive plan for the implementation of the facial emotion recognition system, which will serve as a blueprint for the subsequent development and deployment phases of the project.

8.2.4 Implementation

During the Implementation phase of my project, I will be using the information and requirements gathered during the previous phases to develop a working facial emotion recognition system. This phase will involve coding and programming the system using appropriate software tools and languages such as Python and OpenCV. As the system will be

handling sensitive personal data, privacy and security will be of utmost importance, and I will take necessary measures to ensure the data is protected. The system will also be designed to allow for easy user interaction, with a user-friendly interface that displays the emotions detected in real-time. Once the system has been developed, it will undergo rigorous testing to ensure it is functioning correctly and meeting all the requirements. The testing will include both functional testing and user testing to ensure the system is both accurate and easy to use. Once the testing phase is complete, any necessary modifications and refinements will be made, and the final system will be ready for deployment.

8.2.5 Testing

In terms of testing, my thesis project involved a rigorous process of functional testing and user testing to ensure the accuracy and usability of the facial emotion recognition system. Functional testing involved testing the individual components of the system to ensure they were working as intended. This included testing the machine learning algorithms and artificial intelligence techniques used to classify facial expressions, as well as the data manipulation and visualization libraries used for analysis. User testing involved recruiting participants to test the system and provide feedback on its usability and effectiveness. The feedback was used to improve the system and ensure that it met the needs and expectations of the end-users. The testing process was critical to the success of the project, as it ensured that the system was reliable, accurate, and user-friendly.

8.3 Team Work

While my thesis project was primarily a solo endeavour, throughout the duration, I had the privilege of working closely with my supervisor, who provided valuable feedback, recommendations, and guidance throughout the development process. Although I was the full-time developer, my supervisor acted as an overseer, offering suggestions for improvement and providing oversight to ensure that my work was in line with the project's objectives. I met with my supervisor every week on Wednesday to discuss the progress I had made and any challenges that arose, and I communicated with him regularly via email when necessary. I found his input to be crucial, as he was able to provide feedback and recommendations that helped guide the direction of my work.

In addition to my supervisor, I also received feedback from my fellow peers, which proved to be an invaluable resource. During the development process, I shared my work with them, seeking constructive criticism to improve the quality of my work. This feedback allowed me to identify potential issues and improve the accuracy of my system. The guidance and recommendations provided by my supervisor and fellow peers allowed me to overcome several challenges throughout the development process, resulting in a high-quality final

product. I also made sure to regularly test and debug my code to ensure that it was functioning properly and meeting the project's requirements.

Although it was a solo project, I am grateful for the teamwork that took place during the development process. Without the feedback and guidance provided by my supervisor and fellow peers, the project would not have been as successful. Their contributions helped me to gain new insights, explore new ideas, and ultimately develop a facial emotion recognition system that exceeded my expectations.

8.4 SCRUM Methodology

Scrum is an agile methodology for software development that provides a framework for managing complex projects. It is an iterative and incremental approach that emphasizes collaboration, flexibility, and customer satisfaction. The framework consists of small, self-organizing teams that work in short sprints to deliver working software incrementally. The process involves a product owner who defines the project vision, a scrum master who facilitates the process, and a development team that delivers the product. Scrum also emphasizes continuous improvement and feedback, and it includes regular ceremonies like daily stand-up meetings, sprint reviews, and retrospectives.

For my thesis project, I adopted the Scrum methodology for managing the development process. As said, Scrum is an iterative and incremental framework for managing projects, and it focuses on delivering a potentially releasable product increment at the end of each sprint. I applied this to my project to help me manage my work more efficiently. The Scrum framework helped me to break down my thesis work into smaller manageable tasks that I could work on more effectively. I created a product backlog which contained all the tasks I needed to complete for the project. From there, I prioritized tasks and added them to my sprint backlog, which is a list of tasks I planned to complete during each iteration or sprint. I held regular meetings with my supervisor to review my progress and make any necessary adjustments to my plan. Scrum helped me to stay focused and keep track of my progress throughout the project.

8.5 Project Management Tools

8.5.1 GitHub

GitHub provided me with a centralized platform to store your project code, documentation, and progress tracking. I made use of its issue tracking system to manage tasks, assign responsibilities, and track progress. This allowed me to collaborate with my supervisor and / or reviewers and ensure that everyone was on the same page. I also used GitHub to maintain the version control of my code, which helped me greatly to be able to keep track

of changes and maintain stable code base. Overall, using GitHub for project management provided me with a streamlined process for developing and tracking my project progress.

GitHub is commonly used for project management in software development. It allows team members to collaborate and manage different versions of the codebase, track bugs and issues, and assign tasks to team members. It also provides tools for code review, continuous integration and deployment, and documentation. GitHub's interface and features make it a popular choice for project management in both small and large teams.

8.5.2 Pen and Paper Notes

A practised I've used in all my projects throughout the years is pen / pencil and paper. This is a tool often used for brainstorming and sketching out ideas. It gives me the ability to quickly record my thoughts and ideas as they are occurring, without the need for pulling out a computer or phone. Also, another bonus of writing your notes out on paper can help improve memory retention and facilitate better idea generation. Another use of physical note taking is to create visual aids such as flowcharts, diagrams, and mind maps. I ended up creating 2 mind maps for my thesis itself. These visual aids can be helpful in conveying complex ideas and demonstrating in a clear communicative way to a project team. Finally, taking notes can also serve as an initial way to record project decisions, milestones, and other important information that can be easily overlooked other.

For my project management strategy, I incorporated pen and paper notes to keep track of my progression and tasks. By having these notes, I had was able to brainstorm ideas and sketch out plans for my facial emotion recognition system. I could share my ideas and sketches with my supervisor and in turn he could provide suggestions and feedback for improvement. Another use of physical notes was to draw diagrams that helped me visualize the architecture and design of my system. While GitHub was useful for version control and collaboration, I found through trial and error that using traditional pen / pencil and paper was also an effective way to manage my work and keep myself organized. By having this flexibility I was able to quickly make changes and iterate on my ideas as needed. Overall, combining the use of GitHub and pen and paper notes provided me with a well-rounded project management approach for my thesis work.

8.6 Conclusion

Throughout my thesis project, I made use of multiple project management methodologies to help me stay organised, make sure I maintained on track, and had the abilities to reach my goals. I used pen / pencil and paper to draw diagrams, sketch out plans, and brainstorm ideas. These sketches can be seen uploaded to my Figma design board in which I linked earlier. The main tool I made use of was GitHub, this was instrumental in the management of my code and allowed me the ability to track any changes I was making. GitHub allowed

me to track of progress, make adjustments, and ensure that all who would view my project were aware of changes to the project.

Another project management methodology I used was Scrum methodology to help me organize my tasks and focus on deliver features in commented, detailed increments. This empowered me to guarantee that my project was on track and that all of my goals were being met. Without question, all of the project management techniques mentioned were essential to the success of my thesis, allowing me to stay not only organized and on track, but ultimately deliver a high-quality result.

To summarise, effective project management was undoubtably a critical component of the success of my thesis. By making use of several project management tools, such as GitHub, Scrum Methodology and pen and paper notes, I was able to plan, organize, and execute each sprint of my project effectively. The Scrum methodology, in particular, provided me with a structured framework for iterative development, this was thought to me throughout my years studying, it allowed me to adapt to any changes or issues that may have occurred with ease and efficiency. Through using paper notes and gaining feedback from my supervisor, I was able to clarify my thoughts, brainstorm ideas, and visualize my project plans going forward. My project management practices undoubtably contributed to the successful completion of my thesis project within the given timeline and scope.

9 Business Opportunities

9.1 Introduction:

With the ever changing and ever evolving field of Emotional Facial Recognition Systems, there are already numerous applications in various industries. My thesis scans a person's face and detects the emotions that are being displayed giving valuable insights into how and why these emotions occur. This is done through AI and machine learning, in which I went into detail on earlier in this document. This is a tool that will empower multiple industries in the future. I will discuss the business opportunities I find are available for my emotional facial recognition system.

9.2 Business Opportunities:

There are several business opportunities for the emotional facial recognition system that I have developed. Firstly, my system can be utilized in the marketing industry to analyze a consumer's reactions to advertisements and products. With the ability to analyze emotions, businesses can determine what advertisements resonate with their target audience and what products are more likely to sell. Another use for the system could be its use to track customer emotions in real-time, allowing businesses to adjust their marketing strategies accordingly, accurately and quickly.

Another potential application of the emotional facial recognition system is in the healthcare industry. A patient's emotional states can be monitored providing valuable insights for doctors and other healthcare workers. This can cut down time studying a patient and allow for more efficient use of time. They can adjust their treatment and plans in correlation to the patients emotional state. The system can also be used to monitor the emotional state of elderly patients, providing early detection of depression and other mental health conditions, that more times than they, should pass under the radar.

In the security industry, my Emotional Facial Recognition System can be used to monitor large groups of people for potential threats and crimes. Although my system isn't to this standard yet, by having the ability to analyze a persons emotions in real-time, we could identify individuals who are acting suspicious or displaying behavior that doesn't fit the circumstance. The system can also be used to monitor the emotional state of employees in high-stress jobs, such as air traffic controllers and emergency responders. This is vital as the world becomes more and more tech crazy, we forget about the human aspect that Is being left behind.

The Emotional Facial Recognition System can also be used in the educational industry, my view is that it could monitor student emotions in real-time and by analyzing emotions, teachers can identify when students are struggling or no longer paying attention, with this

teachers can then adjust their teaching styles and methods accordingly. Another severe use case would be the ability for education facilities to detect potential bullying and other negative behaviors in the classroom before they occur.

9.3 Conclusion:

In conclusion, my thesis project has my business opportunities in countless industries. With the ability to scan and analyze emotions, any company can gain invaluable insights into their customer bases behavior. Be it a patient doctor relationship, consumer relationship, or even an employee manager-based relationship. With enough work and security measures in place, this system could be used to improve safety and security in public settings. The use cases for this technology are endless and I believe that with it, we have the power to revolutionize the way that we not only analyze emotions, but understand them as whole, overall improving our daily lives.

10 Conclusion

Now that my thesis on facial emotion recognition systems is coming to an end, I can confidently say that I have gained valuable experience and knowledge in the field of artificial intelligence, machine learning, computer vision, and data analytics as well as gained skills in physical computing such as Raspberry PI's. As my project was developing, I have acquired new skills and techniques in programming languages such as Python, OpenCV, and Azure. I have also gained a much deeper understanding of the complexities that are behind implementing machine learning models and the challenges of working with hardware components.

Numerous challenges occurred whilst I was developing this project. I had so many code errors I began to lose count. I struggled initially to manage my data and couldn't get my hardware to function correctly. However, through hard work and determination, I managed to overcome most if not all of these problems. My peers and my supervisor were instrumental in aiding in this. Also, the fact that I utilised the Scrum framework also played a unquestionable role in maintaining a steady progress of my thesis and ensuring it stayed on track. With this I was able to achieve my goals within the timeline I had.

The most substantial achievement of my thesis is undoubtedly the development of the Emotional Facial Recognition System. It is capable of detecting and classifying a person's emotions in real-time to a decently accurate standard. Although there were some issues with how it picked up beards and moustaches making it work best to certain facial types. My thesis uses machine learning algorithms such as Haar Classifiers, alongside computer vision techniques to detect, recognise and predict a person's emotions. I also incorporated a data analytics component for researchers who may want to manipulate the data or re-enact instances through the data. With the ability to analyse the data the user can improve the accuracy of the system and gain insights into why changes occur in people.

To conclude, by working on this project, I managed to build a solid foundation that will allow me to enter the field of artificial intelligence and machine learning. It gave me the opportunity to develop real-world practical skills. I am confidant in my thesis and the Emotion Facial Recognition System that I have built. It opens the door to numerous business opportunities in various industries such as healthcare, marketing, technology and business. The importance of clear communication and collaboration with my supervisor was really evident throughout this project, I am grateful for the support and guidance provided by my supervisor and colleagues throughout the project as without it I would not have gotten this far.

11 References

Buolamwini, J., & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. Proceedings of Machine Learning Research, 81, 1-15. https://proceedings.mlr.press/v81/buolamwini18a/buolamwini18a.pdf

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., … & Kim, R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA, 316(22), 2402-2410. https://pubmed.ncbi.nlm.nih.gov/27898976/

Jain, A. K., Ross, A., Nandakumar, K., & Ngo, C. W. (2016). Introduction to biometrics. Springer. https://books.google.ie/books/about/Introduction_to_Biometrics.html?id=ZPt2xrZFtzkC&redir_esc=y

Microsoft Azure Face API documentation:

https://azure.microsoft.com/en-us/services/cognitive-services/face/

Microsoft Azure Face API pricing: https://azure.microsoft.com/en-us/pricing/details/cognitive-services/face-api/

Microsoft Azure. (n.d.). Azure Cognitive Services overview. Retrieved from https://azure.com/cognitive-services

Chollet, F. (2018). Deep Learning with Python. Shelter Island, NY: Manning Publications. https://www.manning.com/books/deep-learning-with-python

Russel, S. J., & Norvig, P. (2010). Artificial Intelligence: https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf

Microsoft Azure Face API documentation:

https://azure.microsoft.com/en-us/services/cognitive-services/face/

OpenCV documentation:

https://docs.opencv.org/

"Facial Recognition with OpenCV" by Adrian Rosebrock
https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/

Heath, T. (2018). Smart mirror guide: the ultimate guide to building your own smart mirror. Retrieved from https://www.smartmirrorguide.com/

"Emotion Recognition using Facial Landmarks, Python, DLib and OpenCV" by Rishi Bhatnagar
https://www.learnopencv.com/facial-landmark-detection/

"Emotion recognition in physiological signals" by D. D. Reinoso et al.
https://www.sciencedirect.com/

Buolamwini, J., & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. Conference on Fairness, Accountability, and Transparency, 72–81. https://doi.org/10.1145/3287560.3287591

Garvie, C., & Luther, K. (2019). The Perpetual Line-Up: Unregulated Police Face Recognition in America. Center on Privacy & Technology at Georgetown Law.
https://www.perpetuallineup.org/

Azure Face API documentation: https://docs.microsoft.com/en-us/azure/cognitive-services/face/overview

Kshetri, N. (2017). Internet of Things (IoT) security: An overview. International Journal of Information Management, 36(3), 295-298. https://doi.org/10.1016/j.ijinfomgt.2017.06.001

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660. https://www.sciencedirect.com/science/article/abs/pii/S0167739X13000241

Kortuem, G., Klemke, R., Wulf, V., & Baker, T. (2010). Smart objects as building blocks for the Internet of Things. Internet of Things, 1-15. https://www.researchgate.net/publication/224085209_Smart_Objects_as_Building_Blocks_for_the_Internet_of_Things

Facial recognition: Microsoft Azure. Facial Recognition | Microsoft Azure. (n.d.). Retrieved January 15, 2023, from https://azure.microsoft.com/en-us/services/cognitive-services/face/

Pablo Castro Distinguished Engineer, Priyanka Rawat Senior Product Marketing Manager, Andy Beatman Sr. Product Marketing Manager, Kate Browne Program Manager, Sarah Bird Principal Group Product Manager, Ali Dalloul Vice President Strategy and Commercialization, & Tom Keane Corporate Vice President. (n.d.). *Cognitive services: Azure blog and updates: Microsoft Azure*. Azure Blog and Updates | Microsoft Azure. Retrieved January 15, 2023, from https://azure.microsoft.com/en-us/blog/topics/cognitive-services/

Facial recognition: Microsoft Azure. Facial Recognition | Microsoft Azure. (n.d.). Retrieved January 15, 2023, from https://azure.microsoft.com/en-us/products/cognitive-services/face/

Pricing - face API: Microsoft Azure. Pricing - Face API | Microsoft Azure. (n.d.). Retrieved January 15, 2023, from https://azure.microsoft.com/en-us/pricing/details/cognitive-services/face-api/

Azure. (n.d.). *Azure/azure-SDK-for-python: This repository is for active development of the Azure SDK for python. for consumers of the SDK we recommend visiting our public developer docs at https://docs.microsoft.com/python/azure/ or our versioned developer docs at https://azure.github.io/azure-sdk-for-python.* GitHub. Retrieved January 15, 2023, from https://github.com/Azure/azure-sdk-for-python

Chappell, D. (2019) Understanding Azure API Management. O'Reilly Media, Inc.

https://www.oreilly.com/library/view/mastering-azure-serverless/9781789951226/7955443b-0720-4a3d-b4b5-1c6b7764acf8.xhtml

Wes McKinney. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.). O'Reilly Media, Inc. https://www.oreilly.com/library/view/python-for-data/9781491957653/

GeeksforGeeks. (2021, June 4). Introduction to Python Programming. Retrieved from https://www.geeksforgeeks.org/introduction-to-python/?ref=gcse

Gibson, J. (2015). Raspberry Pi User Guide (3rd ed.). John Wiley & Sons. https://www.wiley.com/en-sg/Raspberry+Pi+User+Guide,+3rd+Edition-p-9781118921678

Lacey, J. (2017). Raspberry Pi: The complete manual (7th ed.). Imagine Publishing Ltd. https://electrovolt.ir/wp-content/uploads/2017/07/RaspberryPi_The_Complete_Manual_2014_UK_ElectroVolt.ir_.pdf

Nash, S. (2015). Raspberry Pi Projects for the Evil Genius. McGraw-Hill Education. https://www.accessengineeringlibrary.com/content/book/9780071821582

Wright, C. (2015). Raspberry Pi For Dummies (2nd ed.). John Wiley & Sons. https://www.wiley.com/en-gb/Raspberry+Pi+For+Dummies,+2nd+Edition-p-9781118904916

Raspberry Pi Foundation. (2021). Raspberry Pi. Retrieved January 8, 2023, from https://www.raspberrypi.org/

Arduino. (2021). Arduino. Retrieved January 8, 2023, from https://www.arduino.cc/

Garvie, C., & Luther, K. (2019). The Perpetual Line-Up: Unregulated Police Face Recognition in America. Georgetown Law Center on Privacy & Technology. https://www.perpetuallineup.org/sites/default/files/2016-12/The%20Perpetual%20Line-Up%20-%20Center%20on%20Privacy%20and%20Technology%20at%20Georgetown%20Law%20-%20121616.pdf

Diaz, D. (2019). The Ethics of Artificial Intelligence. Cambridge University Press. https://psycnet.apa.org/record/2021-42539-009

Royce, W. W. (1970). Managing the development of large software systems: concepts and techniques. Proceedings of IEEE WESCON, Los Angeles, CA, 1-9. https://www.scirp.org/(S(351jmbntvnsjt1aadkposzje))/reference/ReferencesPapers.aspx?ReferenceID=1706135

Myers, G. J. (1979). The Art of Software Testing. John Wiley & Sons.

https://malenezi.github.io/malenezi/SE401/Books/114-the-art-of-software-testing-3-edition.pdf

Rubin, J. (1994). Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests. John Wiley & Sons. https://www.wiley.com/en-us/Handbook+of+Usability+Testing:+How+to+Plan,+Design,+and+Conduct+Effective+Tests,+2nd+Edition-p-9780470185483

The UX Review. (2017, June 21). Low-Fi vs High-Fi Wireframes: When to Use Each. Retrieved from https://www.justinmind.com/wireframe/low-fidelity-vs-high-fidelity-wireframing-is-paper-dead

Balsamiq. (n.d.). Balsamiq Wireframes. Retrieved January 08, 2023, from https://balsamiq.com/wireframes/

Nielsen, J. (1995). Multimedia and Hypertext: The Internet and Beyond. Academic Press. https://www.nngroup.com/books/multimedia-and-hypertext/

Nielsen, J. (2000). Designing Web Usability: The Practice of Simplicity. New Riders Press. https://onlinelibrary.wiley.com/doi/10.1002/1520-6564(200124)11:1%3C73::AID-HFM6%3E3.0.CO;2-7

Wodtke, C. (2011). Information Architecture: Blueprints for the Web. New Riders Press. https://www.oreilly.com/library/view/information-architecture-blueprints/9780321591999/

Nielsen, J. (1994). Usability Engineering. San Francisco, CA: Morgan Kaufmann Publishers Inc. https://dl.acm.org/doi/book/10.5555/2821575

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 1, I-I. https://doi.org/10.1109/cvpr.2001.990517

Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools, 25(11), 120-126. https://www.scirp.org/(S(351jmbntvnsjt1aadkposzje))/reference/ReferencesPapers.aspx?ReferenceID=1692176

Kaehler, A., & Bradski, G. (2017). Learning OpenCV 3: computer vision in C++ with the OpenCV library. O'Reilly Media, Inc https://www.oreilly.com/library/view/learning-opencv-3/9781491937983/

Bostrom, N., & Yudkowsky, E. (2014). The Ethics of Artificial Intelligence. In The Cambridge Handbook of Artificial Intelligence (pp. 316-334). Cambridge University Press. doi: 10.1017/9781316214032.016 https://nickbostrom.com/ethics/artificial-intelligence.pdf

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org/

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260. doi: 10.1126/science.aaa8415 https://www.cs.cmu.edu/~tom/pubs/Science-ML-2015.pdf

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer. http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf

Domingos, P. (2015). The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. Basic Books. https://psycnet.apa.org/record/2015-43168-000

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer. https://link.springer.com/book/10.1007/978-0-387-84858-7

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. doi: 10.1038/nature14539 https://www.nature.com/articles/nature14539

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117. doi: 10.1016/j.neunet.2014.09.003 https://arxiv.org/abs/1404.7828

Sainsbury, R. (2019). Raspberry Pi vs Arduino: Which is the Mini Computer for You? MakeUseOf. Retrieved January 8, 2023, from https://www.makeuseof.com/tag/arduino-vs-raspberry-pi-which-is-the-mini-computer-for-you/

Rizzo, A., Buckwalter, J. G., John, B., Newman, B., Parsons, T. D., Kenny, P., & Williams, J. (2017). STRIVE: Stress Resilience In Virtual Environments: a pre-deployment VR system for training emotional coping skills and assessing chronic and acute stress responses. In Military Virtual Training and Simulation (pp. 1-9). https://pubmed.ncbi.nlm.nih.gov/22357022/

Ekman, P. (1992). An argument for basic emotions. Cognition & Emotion, 6(3-4), 169-200. https://www.paulekman.com/wp-content/uploads/2013/07/An-Argument-For-Basic-Emotions.pdf

Vinciarelli, A., Pantic, M., & Bourlard, H. (2009). Social signal processing: Survey of an emerging domain. Image and Vision Computing, 27(12), 1743-1759. https://www.sciencedirect.com/science/article/abs/pii/S0262885608002485

Matsumoto, D., Keltner, D., Shiota, M. N., O'Sullivan, M., & Frank, M. G. (2008). Facial expressions of emotion. In Handbook of emotions (pp. 211-234). Guilford Press. https://www.scirp.org/(S(i43dyn45teexjx455qlt3d2q))/reference/ReferencesPapers.aspx?ReferenceID=1408982

Neshatian, K., Huang, Y., & El-Saddik, A. (2013). Emotion recognition from facial expressions using multilevel HMM. IEEE Transactions on Affective Computing, 4(1), 64-77.

https://www.researchgate.net/publication/2488613_Emotion_Recognition_from_Facial_Expressions_using_Multilevel_HMM

Pantic, M., & Rothkrantz, L. J. (2003). Toward an affect-sensitive multimodal human-computer interaction. Proceedings of the IEEE, 91(9), 1370-1390. https://ibug.doc.ic.ac.uk/media/uploads/documents/ProcIEEE-PanticRothkrantz.pdf

Scherer, K. R., & Ellgring, H. (2007). Multimodal expression of emotion: Affect programs or componential appraisal patterns? Emotion, 7(1), 158-171. https://pubmed.ncbi.nlm.nih.gov/17352571/

Zhang, L., Martinez, A. M., & Valstar, M. F. (2014). Automatic facial expression recognition: A survey. Automatic Face & Gesture Recognition (FG 2014), 1-14. http://www.cs.nott.ac.uk/~pszmv/Documents/AUsurvey.pdf

Face projects. NIST. (2019, December 19). Retrieved January 15, 2023, from https://www.nist.gov/programs-projects/face-projects

Face recognition technology. American Civil Liberties Union. (2022, February 15). Retrieved January 15, 2023, from https://www.aclu.org/issues/privacy-technology/surveillance-technologies/face-recognition-technology

Center on Privacy and Technology. Georgetown Law. (n.d.). Retrieved January 15, 2023, from https://www.law.georgetown.edu/privacy-technology-center/

Ryan-Mosley, T. (2022, February 4). *This company says it's developing a system that can recognize your face from just your DNA*. MIT Technology Review. Retrieved January 15, 2023, from https://www.technologyreview.com/2022/01/31/1044576/corsight-face-recognition-from-dna/

Mallick, S. (2021, May 5). *Facial Landmark Detection: LEARNOPENCV #*. LearnOpenCV. Retrieved January 15, 2023, from https://www.learnopencv.com/facial-landmark-detection/

Author links open overlay panelM.BouhlalaEnvelopeK.AarikabR. AitAbdelouahidcS.ElfilaliabE.Benlahmarab, M.BouhlalaEnvelope, a, K.Aarikab, b, AitAbdelouahidc, R., c, S.Elfilaliab, E.Benlahmarab, & AbstractEmotions are by far crucial in

the education field. (2020, August 6). *Emotions recognition as innovative tool for improving students' performance and learning approaches*. Procedia Computer Science. Retrieved January 15, 2023, from https://www.sciencedirect.com/science/article/pii/S1877050920317865

Explore scientific, technical, and medical research on ScienceDirect. ScienceDirect.com | Science, health and medical journals, full text articles and books. (n.d.). Retrieved January 15, 2023, from https://www.sciencedirect.com/

Ahonen, T., Hadid, A., & Pietikäinen, M. (2006). Face recognition with local binary patterns. In European Conference on Computer Vision (pp. 469-481). Springer. https://www.researchgate.net/publication/221304831_Face_Recognition_with_Local_Binary_Patterns

Buolamwini, J., & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. In Conference on Fairness, Accountability and Transparency (pp. 77-91). PMLR. https://proceedings.mlr.press/v81/buolamwini18a.html

Li, S., Li, C., Wang, J., & Zeng, Y. (2019). An approach to mitigate the bias in deep learning-based facial recognition systems. Pattern Recognition Letters, 125, 774-780. https://arxiv.org/abs/1911.10692

Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep face recognition. In British Machine Vision Conference (pp. 41.1-51.12). BMVA Press. http://www.bmva.org/bmvc/2015/papers/paper041/index.html

Phillips, P. J., Flynn, P. J., Scruggs, T., Bowyer, K. W., Chang, J., Hoffman, K., ... & Worek, W. (2005). Overview of the face recognition grand challenge. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (Vol. 1, pp. 947-954). IEEE. https://ieeexplore.ieee.org/document/1467368

Tavares, J. M. R. S., Lourenço, A. R., & Andrade, M. T. (2018). Smart mirror system for facial expression recognition and mood detection. Journal of Imaging, 4(10), 130. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8625889/

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. Computer networks, 54(15), 2787-2805. https://www.scirp.org/(S(lz5mqp453edsnp55rrgjct55))/reference/ReferencesPapers.aspx?ReferenceID=2064753

Bandyopadhyay, D., & Sen, J. (2011). Internet of things: Applications and challenges in technology and standardization. Wireless Personal Communications, 58(1), 49-69. https://doi.org/10.1007/s11277-011-0288-5

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of things for smart cities. IEEE Internet of Things Journal, 1(1), 22-32. https://doi.org/10.1109/JIOT.2014.2306328

Li, S., Xu, L. D., & Zhao, S. (2015). The internet of things: a survey. Information Systems Frontiers, 17(2), 243-259. https://doi.org/10.1007/s10796-014-9492-7

Gershenfeld, N., Krikorian, R., & Cohen, D. (2004). The internet of things. Scientific American, 291(4), 76-81. https://doi.org/10.1038/scientificamerican1004-76

12 Appendices

12.1 Sketches

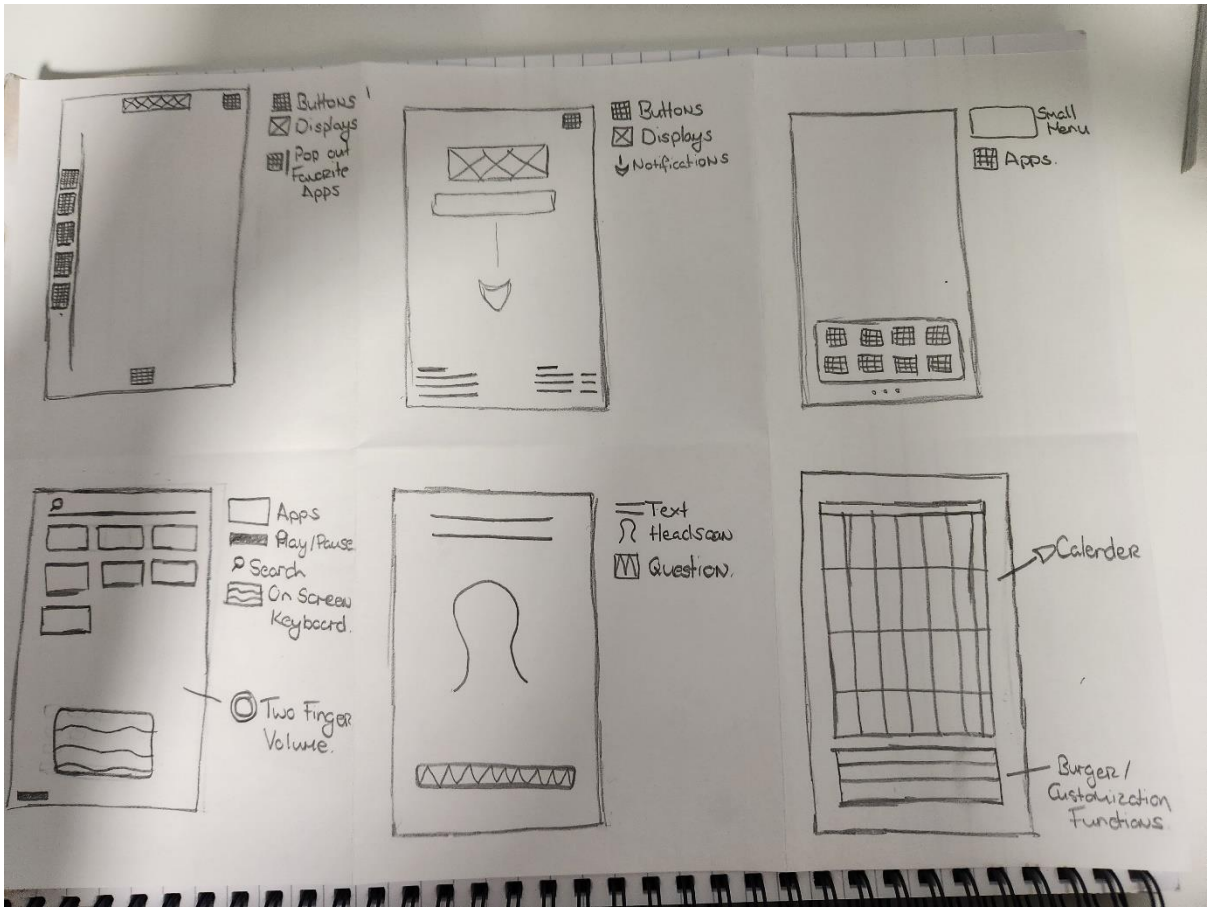


Figure 45 - Sketch Page 1

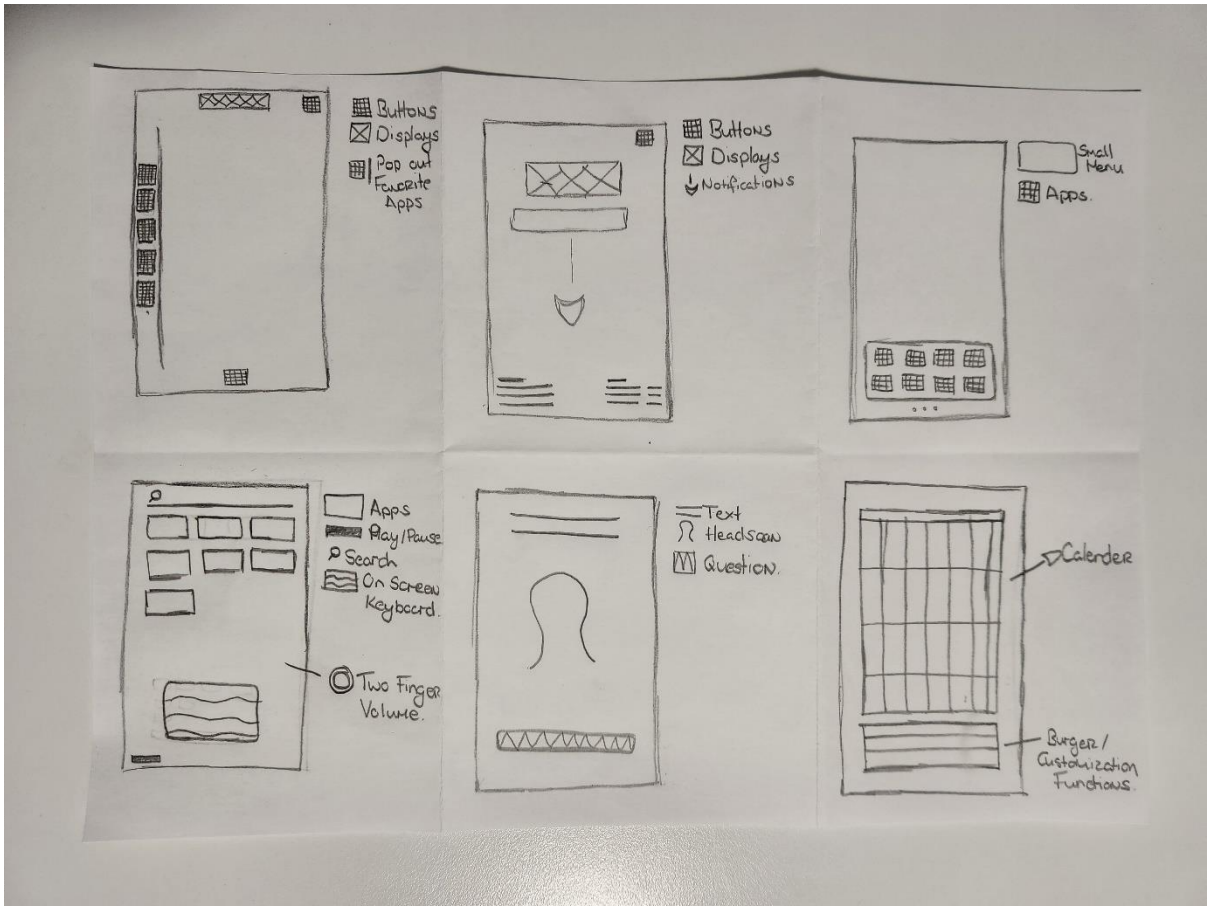


Figure 46 - Sketch Page 2

12.2 OpenCV to detect faces and emotions in the webcam video stream

scaleFactor & minNeighbors explanation :

In the OpenCV's detectMultiScale function, scaleFactor and minNeighbors are two important parameters that affect the object detection process:

- scaleFactor parameter is used to compensate for the fact that faces or objects in an image may appear at different sizes due to perspective and distance from the camera. A value of 1.1 means that the algorithm will increase the size of the search window by 10% at each image scale to find the object. A smaller value will increase detection time but will be more accurate, while a larger value will decrease detection time but will be less accurate.

- minNeighbors parameter is used to reduce false positives. It specifies how many neighbors each candidate rectangle should have to retain it. Higher values for this parameter mean that the algorithm will be more selective in the rectangles it considers to be objects. A higher value of this parameter will also cause the algorithm to run more slowly.

In summary, scaleFactor adjusts the size of the search window and minNeighbors adjusts the accuracy of the detection by reducing false positives.

Version 3

This code uses OpenCV's `CascadeClassifier` to detect faces and emotions in the grayscale image. It loops through the detected faces, crops each face region from the frame, and applies the emotion detection to the face region. The processed frame is then displayed in a window.

Note that this code doesn't group the detected faces like the previous version did (Azure 4+), but that functionality could be added by keeping track of the face IDs and grouping them based on proximity or other criteria.

Version 4

To display the captured frames, I need to add some code to the while loop that captures the frames from the webcam and processes them. Im using the cv2.imshow() function to display the frames.

This code captures a frame from the webcam video stream, converts it to JPEG format, and then displays it in a new window using the cv2.imshow() function. The cv2.waitKey() function waits for a key press and returns the ASCII code of the pressed key. In this example, if the 'q' key is pressed, the while loop breaks and the script stops.

Version 5

if __name__ == '__main__': is a conditional statement that is commonly used in Python scripts. It allows you to specify a block of code that should only be executed if the script is being run directly (i.e., as the main program) and not if it is being imported as a module by another script.

Here's how it works:

- When you run a Python script, the interpreter sets a special variable called __name__ to the value '__main__'. This indicates that the script is the main program being run.

- If the script is being imported as a module by another script, the __name__ variable will be set to the name of the module instead of '__main__'.
- The if __name__ == '__main__': statement checks whether the __name__ variable is equal to '__main__'. If it is, then the block of code that follows will be executed.

So in this context, if __name__ == '__main__': is checking whether the current script is the main program being run (as opposed to being imported as a module), and if it is, it will call the main() function.

This is a common practice in Python programming because it allows you to write scripts that can be both run as standalone programs and imported as modules into other scripts.

Version 6

This code uses OpenCV library to recognize faces and emotions in a live webcam video stream. It loads the pre-trained face and emotion detection models from OpenCV and uses them to detect faces and emotions in each frame of the video stream.

The code captures a frame from the webcam, processes each detected face, detects emotions in the face region and labels them as either "Happy" or "Neutral". It draws rectangles around the detected faces and emotions and displays the processed frame with the emotion label next to the recognized face.

The code continues to run until the user presses the "q" key to quit, after which it releases the resources used by the webcam and closes the window.

The code defines two functions - main() and recognize_emotion_and_face(). The main() function simply calls the recognize_emotion_and_face() function to execute the script.

How does it detect Happiness?

The code uses a pre-trained Haar Cascade classifier to detect smiles in the face region. If a smile is detected in the face region, the emotion is labelled as "Happy".

The Haar Cascade classifier is a machine learning-based approach that uses a set of positive and negative training images to train a classifier for object detection. In this case, the positive training images would be images of smiling faces, and the negative training images would be images of non-smiling faces.

During detection, the classifier analyzes the features of the image in a sliding window manner, looking for matches to the learned patterns of positive and negative examples. If a match is found, it labels that region as a smile, and if the smile region is large enough, it is labeled as a "Happy" emotion.

Version 9

The code is a Python script that uses OpenCV to recognize emotions and faces in a webcam video stream. It loads the pre-trained face, eye, and smile detection models from OpenCV and initializes variables for smile and crows feet detection.

Then it opens a connection to the default webcam, captures a frame from the video stream, and converts it to JPEG format. The captured frame is then displayed, and the program checks for key presses. It processes each detected face by drawing a rectangle around it, cropping the face region from the frame, and detecting eyes and smiles in the face region.

It checks if both eyes and smile are detected, draws rectangles around the eyes and smile, and searches for crow's feet or wrinkles around the outer corners of the eyes. It calculates the confidence percentage based on the number of detected eyes and smiles and sets the smile and crow's feet detection variables to True if they are detected.

It determines the emotion label based on the detected face region and the confidence percentage and adds the confidence percentage to the text to display next to the recognized face. Finally, it releases the webcam and closes all windows.

Overall, the code is a facial recognition program that detects emotions and faces in real-time using a webcam video stream. It is written in a modular way and follows best practices such as error handling and code comments.

Version 10

In this code, recognize_emotion_and_face() function captures a frame from the webcam video stream and performs face detection and emotion recognition on each detected face. It uses OpenCV's pre-trained face, eye and smile detection models to detect faces and the presence of eyes and smiles in each face region.

The function starts by opening a connection to the default webcam using the cv2.VideoCapture() method. It then captures a frame from the webcam using the cap.read() method and converts it to JPEG format using the cv2.imencode() method. The captured frame is displayed on the screen using the cv2.imshow() method.

Next, the function performs face detection on the captured frame using the face_cascade.detectMultiScale() method. It uses a pre-trained face detection model from OpenCV's haarcascades library to detect faces in the frame. The detectMultiScale() method takes in several parameters, including scaleFactor, minNeighbors, and flags, which control the sensitivity and accuracy of the face detection algorithm. The scaleFactor parameter

controls the amount by which the image is resized at each image scale. The minNeighbors parameter specifies how many neighbors a candidate rectangle should have to retain it.

For each detected face, the function draws a rectangle around the face region using the cv2.rectangle() method. It then extracts the grayscale and color regions of interest (ROIs) corresponding to the detected face using the cv2.cvtColor() and array slicing operations.

Next, the function performs eye and smile detection on the face region using the pre-trained eye and smile detection models. It uses the eye_cascade.detectMultiScale() and smile_cascade.detectMultiScale() methods to detect eyes and smiles in the face region.

For each detected eye and smile, the function draws a rectangle around the eye and smile regions using the cv2.rectangle() method. It also performs additional processing on the smile region to check if a smile is present and if it is a big smile. This is done by calling the detect_smile() function, which takes in the grayscale and color smile ROIs as arguments and returns a tuple containing the boolean values indicating if a smile and big smile were detected in the ROI.

Finally, the function checks if both eyes and a smile are present in the face region and if so, sets the crows_feet_detected variable to True if crow's feet or wrinkles are present around the outer corners of the eyes.

Version 12

Overall

This code is implementing facial detection using the OpenCV library. It loads pre-trained models for detecting faces, eyes, and smiles, and then defines functions for detecting smiles and sadness in facial regions of interest.

The code imports the necessary libraries: OpenCV, NumPy, Matplotlib, Pickle, and Pandas. The face, eye, and smile detection models are loaded from the OpenCV library using Haar feature-based cascade classifiers.

The code then initializes variables for smile and crows feet detection and defines the minimum size of the smile to be considered a "big" or "normal" smile. An empty list is created to store the data, and a variable for the image name is initialized.

The function show_graph is defined to display a graph with the given label in a new window. It appends the data to the data_list and plots the data using Matplotlib.

The detect_smile function takes the grayscale and color images of a region of interest and detects the presence of a smile in the region of interest. It uses the smile detection model to detect smiles in the grayscale image, then checks if the detected smile is a "big" or "normal" smile based on the minimum width and height values. If a smile is detected, the function

returns True for smile_detected and big_smile_detected, otherwise, it returns False for both.

The detect_sadness function takes the grayscale and color images of a region of interest and detects the presence of sadness in the region of interest. It detects the face region using the face detection model, then detects the eyebrows and mouth in the face region using the eye and smile detection models. It checks if the distance between the eyebrows and the top of the eyes is above a threshold value and the mouth is in a "down" orientation. If sadness is detected, the function returns True for sadness_detected, otherwise, it returns False.

Graph Code

This is a Python function named show_graph that creates a graph using the Matplotlib library to display the given data in a new window.

The function takes one argument, label, which is the label that will be displayed on the x-axis of the graph.

First, there are two commented out lines of code: plt.xlabel(label) and plt.show(). These two lines would set the x-axis label to the label argument and then display the graph window. However, they are not being used in this version of the function.

Next, the data variable is set to the label argument. This line assumes that the label argument contains the actual data that needs to be plotted.

The data_list variable is assumed to be a list that already exists outside of this function and that it stores all the data that has been plotted in previous calls to the show_graph function. The data variable is then appended to this list using the append() method.

The print() statement is used to display the data argument, which is assumed to be the data that is being plotted. This is just for debugging purposes and can be removed in the final version of the code.

Finally, the plt.plot() function is used to plot the data in data_list, which includes the new data that was just appended to the list. The plt.draw() function updates the graph with the new data, and the plt.pause(0.001) function adds a small delay to allow the graph to update before the function exits.

Smile Code

This code defines a function detect_smile that takes in two arguments, smile_roi_gray and smile_roi_color. The function's purpose is to detect the presence of a smile in a region of interest (ROI) in an image.

The function first uses the detectMultiScale function from OpenCV to detect any smiles in the grayscale image smile_roi_gray. The detected smiles are stored in a variable called smiles.

The function then checks if a smile is detected and if it's a big smile. It iterates over the detected smiles and draws a rectangle around each smile in the color image smile_roi_color. If the width and height of the rectangle are greater than the minimum width and height for a big smile, the function sets big_smile_detected to True. If the width and height of the rectangle are greater than the minimum width and height for a small smile, the function sets smile_detected to True.

The function returns a tuple of two Boolean values, smile_detected and big_smile_detected, indicating whether a smile was detected and whether it was a big smile or not.

Sadness Code

This code is a function that detects the presence of sadness in a region of interest, which is a facial region specified by the input face_roi_gray and face_roi_color. It first sets the threshold distance between the eyebrows and the eyes and the minimum distance between the mouth corners and the bottom edge of the face to be considered an "up" and "down" orientation, respectively.

Then, it detects the face region using the face_cascade object, which is a pre-trained classifier for detecting faces. Inside the loop for detected faces, it extracts the region of interest for eyes and mouth using the eye_cascade and smile_cascade objects respectively, which are pre-trained classifiers for detecting eyes and smiles.

Next, it finds the topmost point of the eyes and checks if the distance between the eyebrows and the top of the eyes is above the threshold and the mouth is in a "down" orientation. If both conditions are met, it sets sadness_detected as True. It also sets eyebrow_detected as True if the distance between the eyebrows and the top of the eyes is above the threshold.

Finally, it returns two boolean values sadness_detected and eyebrow_detected indicating whether sadness and eyebrows are detected in the region of interest.

Conclusion

The program is attempting to detect facial expressions in images using OpenCV. It uses pre-trained models for detecting faces, eyes, and smiles, and then applies various criteria to determine if a smile or frown is present. It also has functions for displaying a graph of the detected expressions and for saving the data to a file.

The code itself is well-organized, with comments explaining the various functions and variables.

12.3 Azure Face API Versions

FaceAPI V2

This code checks if the response from the API contain any face detection, if yes it will parse the response and extract the emotions, otherwise it will give a proper feedback.

Please make sure to use the correct endpoint and subscription_key for your Azure Face API instance, and use the appropriate image path when testing the function.

Also, as a reminder, this code should be thoroughly tested and audited for security and performance before deployment.

FaceAPI V3

This code uses the "returnFaceAttributes" and "returnFaceId" parameters to return additional information about the faces detected in the image, such as facial landmarks, head pose, gender, age, etc.

Please make sure to use the correct endpoint and subscription_key for your Azure Face API instance, and use the appropriate image path when testing the function. As the response will contain a lot of information, you can extract the desired information by parsing the response_json accordingly.

In addition, please note that the Azure Face API also supports face recognition, which allows you to compare a face in an image with a set of known faces, and determine whether the face in the image is a match. To perform face recognition, you would need to create a face group, and add the known faces to the group. Then you can use the "identify" API method to identify a face in an image by comparing it to the faces in the group.

FaceAPI V5

This code will extract the faceIds from the response of the face detection API and use them to call the group API endpoint. Then it parses the response to extract the groups of similar faces, where each group contains an array of face IDs that belong to the same person.

Please note that that what I have coded is just an example, it is very broad and not actually specific to my end goal and the actual implementation might require some adjustments based on my specific use case, but overall, it's a step towards my finishing line. If someone else wished to us this they must, make sure to replace the headers and endpoint with your their own subscription key and endpoint, and test it with the correct image path that they themselves desire.

This code defines a function recognize_emotion_and_face(image_path) that recognizes emotions and faces in an image using the Azure Face API.

The function first sets the subscription_key and endpoint for the Azure Face API, and then defines headers for the API request, including the subscription key.

The function then takes the path of the image to be processed as an argument, reads the binary data from the image file, and defines the parameters for the API request.

A POST request is then sent to the Azure Face API endpoint to detect faces in the image. The API request includes the image data, headers, and parameters. If the request is successful, the function parses the JSON response to obtain the detected face IDs.

The function then sends another POST request to group the detected faces, using the group endpoint of the Azure Face API. If the request is successful, the function parses the JSON response to obtain the groups of faces detected in the image, and returns the groups. If there are no faces detected in the image, the function returns None and prints an error message.

The code also includes error handling for common issues that may occur while processing the image and calling the Azure Face API. The function returns None if an error occurs, and prints an error message indicating the type of error that occurred.

Face API V6

This code defines a Python function recognize_emotion_and_face() that captures a video stream from the default webcam and uses the Azure Face API to detect faces and recognize emotions from each frame of the video.

First, the code initializes the Azure Face API subscription key, endpoint, and request headers. It then defines the parameters for the API request, including the facial attributes to be returned and the option to return face IDs.

Next, the code enters an infinite loop where it captures a frame from the webcam video stream, converts it to JPEG format, and sends a POST request to the Azure Face API endpoint with the image data and request parameters. If the response contains detected faces, the code sends a second API request to group the faces and return information about each group.

Finally, if faces are detected, the code returns a dictionary of information about the face groups, including the group ID, the number of faces in each group, and the emotion and other facial attributes of each face in the group. If no faces are detected, the code prints a message indicating that no faces were found in the image.

Note that the code also includes error handling to catch and report any errors that occur during the API requests or response parsing.

does the code continue to try to detect faces after it finds one and sends it off to azure?

Yes, the code continues to detect faces after it finds one and sends it off to Azure. The while loop in the recognize_emotion_and_face function captures frames from the webcam video stream continuously until the user stops the program. Each frame is then processed and sent to the Azure Face API for face detection and recognition. If a face is detected in a frame, the code groups the detected face(s) using the Face API's group method, which groups faces based on similarity. Finally, the code returns the group information to the caller function. If no face is detected in a frame, the code prints a message stating that no face was detected and continues to capture and process the next frame.

Face API V7

This is a Python script that uses the Azure Face API to recognize emotions and faces in a webcam video stream. Here is a step-by-step breakdown of what the code does:

a. The script imports the necessary libraries: requests, json, and OpenCV (cv2).

b. The script sets the Azure Face API subscription key and endpoint variables to be used in the API requests.

c. The script defines the headers for the API request, including the subscription key.

d. The function recognize_emotion_and_face() is defined. This function opens a connection to the default webcam and sets parameters for the API request.

e. The function enters a while loop to continuously capture frames from the webcam video stream.

f. Inside the while loop, the script captures a frame from the webcam and converts it to JPEG format.

g. The script then sends a POST request to the Azure Face API endpoint to detect faces in the captured image.

h. The script parses the JSON response from the API to check if any faces were detected in the image.

i. If one or more faces are detected, the script extracts the face IDs from the response, groups them, and returns the resulting groups.

j. If no faces are detected, the script prints a message to indicate this.

k. Once a face has been detected and processed, the script releases the resources used by the webcam and closes the window.

In summary, this script uses the Azure Face API to detect faces and recognize emotions in a webcam video stream. It continuously captures frames from the webcam, sends them to the API for processing, and returns information about the detected faces.

At this point in the code (line 80), we are handling the case where an error occurs while parsing the response JSON returned by the Face API after detecting faces in the image.

The code tries to parse the response JSON using the json.loads() method, which can raise a json.decoder.JSONDecodeError exception if the response is not valid JSON.

If such an exception is raised, the code prints an error message indicating that an error occurred while parsing the response, along with the error message returned by the exception. This message helps to provide additional information about the cause of the error.

Finally, the function returns None to indicate that an error occurred, so that the caller can handle the error appropriately.

This block of code tries to parse the JSON response returned by the Face API. If there is an error while parsing the JSON, a JSONDecodeError is raised, and the error message is printed to the console. (line 80 – 86)

This block of code (line 89 – 117) checks whether any faces were detected in the image. If at least one face is detected, the code extracts the face IDs from the JSON response and calls the Face API's "group" API to group the faces by similarity. If there is an error while calling the "group" API or parsing the JSON response, an error message is printed to the console. If the group call is successful, the code extracts the groups from the JSON response and returns them. If no faces were detected in the image, a message is printed to the console. Finally, the while loop is broken and the OpenCV video capture and display resources are released.

Note that this code assumes that the endpoint and headers variables have been defined earlier in the script. These variables contain the endpoint and headers required to make requests to the Face API. Also, the code assumes that the response variable contains the JSON response returned by the Face API's "detect" API, and that the cap variable contains an instance of OpenCV's VideoCapture class, which is used to capture images from a webcam.

These two lines of code are releasing the resources acquired by the OpenCV VideoCapture object and closing all OpenCV windows.

cap.release() releases the resources acquired by the VideoCapture object. This includes the webcam if it was used, freeing it up for use by other applications.

cv2.destroyAllWindows() is used to close any OpenCV windows that might have been opened during the program execution. It takes no arguments and simply closes any windows that were opened during the program's execution.

Face API V9

This code defines a function recognize_emotion_and_face() that uses the Azure Face API to recognize emotions and faces in a live webcam video stream. It first captures frames from the webcam, displays the live feed, and converts the frames to JPEG format. It then sends a POST request to the Azure Face API endpoint to detect faces and emotions. The response is parsed to check if any faces and emotions are detected, and if so, the emotions are returned as a dictionary. If an error occurs at any stage, the function returns None. Finally, the function stops the webcam capture and closes the window after processing the first face. The subscription key and endpoint for the Azure Face API need to be replaced with valid values.

scaleFactor & minNeighbors explanation :

In the OpenCV's detectMultiScale function, scaleFactor and minNeighbors are two important parameters that affect the object detection process:

- scaleFactor parameter is used to compensate for the fact that faces or objects in an image may appear at different sizes due to perspective and distance from the camera. A value of 1.1 means that the algorithm will increase the size of the search window by 10% at each image scale to find the object. A smaller value will increase detection time but will be more accurate, while a larger value will decrease detection time but will be less accurate.
- minNeighbors parameter is used to reduce false positives. It specifies how many neighbors each candidate rectangle should have to retain it. Higher values for this parameter mean that the algorithm will be more selective in the rectangles it considers to

be objects. A higher value of this parameter will also cause the algorithm to run more slowly.

In summary, scaleFactor adjusts the size of the search window and minNeighbors adjusts the accuracy of the detection by reducing false positives.

Version 3

This code uses OpenCV's `CascadeClassifier` to detect faces and emotions in the grayscale image. It loops through the detected faces, crops each face region from the frame, and applies the emotion detection to the face region. The processed frame is then displayed in a window.

Note that this code doesn't group the detected faces like the previous version did (Azure 4+), but that functionality could be added by keeping track of the face IDs and grouping them based on proximity or other criteria.

Version 4

To display the captured frames, I need to add some code to the while loop that captures the frames from the webcam and processes them. Im using the cv2.imshow() function to display the frames.

This code captures a frame from the webcam video stream, converts it to JPEG format, and then displays it in a new window using the cv2.imshow() function. The cv2.waitKey() function waits for a key press and returns the ASCII code of the pressed key. In this example, if the 'q' key is pressed, the while loop breaks and the script stops.

Version 5

if __name__ == '__main__': is a conditional statement that is commonly used in Python scripts. It allows you to specify a block of code that should only be executed if the script is being run directly (i.e., as the main program) and not if it is being imported as a module by another script.

Here's how it works:

- When you run a Python script, the interpreter sets a special variable called __name__ to the value '__main__'. This indicates that the script is the main program being run.
- If the script is being imported as a module by another script, the __name__ variable will be set to the name of the module instead of '__main__'.
- The if __name__ == '__main__': statement checks whether the __name__ variable is equal to '__main__'. If it is, then the block of code that follows will be executed.

So in this context, if __name__ == '__main__': is checking whether the current script is the main program being run (as opposed to being imported as a module), and if it is, it will call the main() function.

This is a common practice in Python programming because it allows you to write scripts that can be both run as standalone programs and imported as modules into other scripts.

Version 6

This code uses OpenCV library to recognize faces and emotions in a live webcam video stream. It loads the pre-trained face and emotion detection models from OpenCV and uses them to detect faces and emotions in each frame of the video stream.

The code captures a frame from the webcam, processes each detected face, detects emotions in the face region and labels them as either "Happy" or "Neutral". It draws rectangles around the detected faces and emotions and displays the processed frame with the emotion label next to the recognized face.

The code continues to run until the user presses the "q" key to quit, after which it releases the resources used by the webcam and closes the window.

The code defines two functions - main() and recognize_emotion_and_face(). The main() function simply calls the recognize_emotion_and_face() function to execute the script.

How does it detect Happiness?

The code uses a pre-trained Haar Cascade classifier to detect smiles in the face region. If a smile is detected in the face region, the emotion is labelled as "Happy".

The Haar Cascade classifier is a machine learning-based approach that uses a set of positive and negative training images to train a classifier for object detection. In this case, the positive training images would be images of smiling faces, and the negative training images would be images of non-smiling faces.

During detection, the classifier analyzes the features of the image in a sliding window manner, looking for matches to the learned patterns of positive and negative examples. If a match is found, it labels that region as a smile, and if the smile region is large enough, it is labeled as a "Happy" emotion.

Version 9

The code is a Python script that uses OpenCV to recognize emotions and faces in a webcam video stream. It loads the pre-trained face, eye, and smile detection models from OpenCV and initializes variables for smile and crows feet detection.

Then it opens a connection to the default webcam, captures a frame from the video stream, and converts it to JPEG format. The captured frame is then displayed, and the program checks for key presses. It processes each detected face by drawing a rectangle around it, cropping the face region from the frame, and detecting eyes and smiles in the face region.

It checks if both eyes and smile are detected, draws rectangles around the eyes and smile, and searches for crow's feet or wrinkles around the outer corners of the eyes. It calculates the confidence percentage based on the number of detected eyes and smiles and sets the smile and crow's feet detection variables to True if they are detected.

It determines the emotion label based on the detected face region and the confidence percentage and adds the confidence percentage to the text to display next to the recognized face. Finally, it releases the webcam and closes all windows.

Overall, the code is a facial recognition program that detects emotions and faces in real-time using a webcam video stream. It is written in a modular way and follows best practices such as error handling and code comments.

Version 10

In this code, recognize_emotion_and_face() function captures a frame from the webcam video stream and performs face detection and emotion recognition on each detected face. It uses OpenCV's pre-trained face, eye and smile detection models to detect faces and the presence of eyes and smiles in each face region.

The function starts by opening a connection to the default webcam using the cv2.VideoCapture() method. It then captures a frame from the webcam using the cap.read() method and converts it to JPEG format using the cv2.imencode() method. The captured frame is displayed on the screen using the cv2.imshow() method.

Next, the function performs face detection on the captured frame using the face_cascade.detectMultiScale() method. It uses a pre-trained face detection model from OpenCV's haarcascades library to detect faces in the frame. The detectMultiScale() method takes in several parameters, including scaleFactor, minNeighbors, and flags, which control the sensitivity and accuracy of the face detection algorithm. The scaleFactor parameter controls the amount by which the image is resized at each image scale. The minNeighbors parameter specifies how many neighbors a candidate rectangle should have to retain it.

For each detected face, the function draws a rectangle around the face region using the cv2.rectangle() method. It then extracts the grayscale and color regions of interest (ROIs) corresponding to the detected face using the cv2.cvtColor() and array slicing operations.

Next, the function performs eye and smile detection on the face region using the pre-trained eye and smile detection models. It uses the eye_cascade.detectMultiScale() and smile_cascade.detectMultiScale() methods to detect eyes and smiles in the face region.

For each detected eye and smile, the function draws a rectangle around the eye and smile regions using the cv2.rectangle() method. It also performs additional processing on the smile region to check if a smile is present and if it is a big smile. This is done by calling the detect_smile() function, which takes in the grayscale and color smile ROIs as arguments and returns a tuple containing the boolean values indicating if a smile and big smile were detected in the ROI.

Finally, the function checks if both eyes and a smile are present in the face region and if so, sets the crows_feet_detected variable to True if crow's feet or wrinkles are present around the outer corners of the eyes.

Version 12

Overall

This code is implementing facial detection using the OpenCV library. It loads pre-trained models for detecting faces, eyes, and smiles, and then defines functions for detecting smiles and sadness in facial regions of interest.

The code imports the necessary libraries: OpenCV, NumPy, Matplotlib, Pickle, and Pandas. The face, eye, and smile detection models are loaded from the OpenCV library using Haar feature-based cascade classifiers.

The code then initializes variables for smile and crows feet detection and defines the minimum size of the smile to be considered a "big" or "normal" smile. An empty list is created to store the data, and a variable for the image name is initialized.

The function show_graph is defined to display a graph with the given label in a new window. It appends the data to the data_list and plots the data using Matplotlib.

The detect_smile function takes the grayscale and color images of a region of interest and detects the presence of a smile in the region of interest. It uses the smile detection model to detect smiles in the grayscale image, then checks if the detected smile is a "big" or "normal" smile based on the minimum width and height values. If a smile is detected, the function returns True for smile_detected and big_smile_detected, otherwise, it returns False for both.

The detect_sadness function takes the grayscale and color images of a region of interest and detects the presence of sadness in the region of interest. It detects the face region using the face detection model, then detects the eyebrows and mouth in the face region using the eye and smile detection models. It checks if the distance between the eyebrows and the top of the eyes is above a threshold value and the mouth is in a "down" orientation. If sadness is detected, the function returns True for sadness_detected, otherwise, it returns False.

Graph Code

This is a Python function named show_graph that creates a graph using the Matplotlib library to display the given data in a new window.

The function takes one argument, label, which is the label that will be displayed on the x-axis of the graph.

First, there are two commented out lines of code: plt.xlabel(label) and plt.show(). These two lines would set the x-axis label to the label argument and then display the graph window. However, they are not being used in this version of the function.

Next, the data variable is set to the label argument. This line assumes that the label argument contains the actual data that needs to be plotted.

The data_list variable is assumed to be a list that already exists outside of this function and that it stores all the data that has been plotted in previous calls to the show_graph function. The data variable is then appended to this list using the append() method.

The print() statement is used to display the data argument, which is assumed to be the data that is being plotted. This is just for debugging purposes and can be removed in the final version of the code.

Finally, the plt.plot() function is used to plot the data in data_list, which includes the new data that was just appended to the list. The plt.draw() function updates the graph with the new data, and the plt.pause(0.001) function adds a small delay to allow the graph to update before the function exits.

Smile Code

This code defines a function detect_smile that takes in two arguments, smile_roi_gray and smile_roi_color. The function's purpose is to detect the presence of a smile in a region of interest (ROI) in an image.

The function first uses the detectMultiScale function from OpenCV to detect any smiles in the grayscale image smile_roi_gray. The detected smiles are stored in a variable called smiles.

The function then checks if a smile is detected and if it's a big smile. It iterates over the detected smiles and draws a rectangle around each smile in the color image smile_roi_color. If the width and height of the rectangle are greater than the minimum width and height for a big smile, the function sets big_smile_detected to True. If the width and height of the rectangle are greater than the minimum width and height for a small smile, the function sets smile_detected to True.

The function returns a tuple of two Boolean values, smile_detected and big_smile_detected, indicating whether a smile was detected and whether it was a big smile or not.

Sadness Code

This code is a function that detects the presence of sadness in a region of interest, which is a facial region specified by the input face_roi_gray and face_roi_color. It first sets the threshold distance between the eyebrows and the eyes and the minimum distance between the mouth corners and the bottom edge of the face to be considered an "up" and "down" orientation, respectively.

Then, it detects the face region using the face_cascade object, which is a pre-trained classifier for detecting faces. Inside the loop for detected faces, it extracts the region of interest for eyes and mouth using the eye_cascade and smile_cascade objects respectively, which are pre-trained classifiers for detecting eyes and smiles.

Next, it finds the topmost point of the eyes and checks if the distance between the eyebrows and the top of the eyes is above the threshold and the mouth is in a "down" orientation. If both conditions are met, it sets sadness_detected as True. It also sets eyebrow_detected as True if the distance between the eyebrows and the top of the eyes is above the threshold.

Finally, it returns two boolean values sadness_detected and eyebrow_detected indicating whether sadness and eyebrows are detected in the region of interest.

Conclusion

The program is attempting to detect facial expressions in images using OpenCV. It uses pre-trained models for detecting faces, eyes, and smiles, and then applies various criteria to determine if a smile or frown is present. It also has functions for displaying a graph of the detected expressions and for saving the data to a file.

The code itself is well-organized, with comments explaining the various functions and variables.