Reimagining the student services

Tariq Horan

N00202437

Supervisor: Mohammed Cherbatji

Second Reader: John Montayne

Year 4 2023/24

DL836 BSc (Hons) in Creative Computing

Abstract

This project aimed to construct a system that allows students at IADT to efficiently access information regarding the institute's services. Initiated by a group within IADT, it was identified that the existing student services page was inadequate and required redevelopment. Subsequently, the project proposal was submitted to and won the N-TUTORR competition, securing funding for its development. With a focus on inclusivity, the application caters to neurodivergent students' needs. This report will outline the project's progression and the resulting system. The development process encompassed various stages, including planning, design, implementation, and testing. Testing was conducted iteratively throughout and after implementation, revealing positive results regarding usability and functionality.

Acknowledgements

I would like to take this opportunity to give a special thanks to my supervisor for this project Mohammed Cherbatji, his guidance, expertise, and encouragement have been invaluable throughout this journey. Additionally, I am grateful to my lecturers, Stefan Berrios and John Montayne, whose valuable input and support helped keep me motivated throughout the project. Lastly, I would like to express my gratitude to my family and friends who provided constructive criticism, helping me to refine my work and grow as a professional.

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: Tariq Horan

Signed

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

Table of Contents

1 Introduction

The application to be built is a student services platform for the Institute of Art Design and Technology, which allows students to find the information they need. There currently is a student services that this project will aspire to improve. This will be built using React JS for the front end with Tailwind CSS styling, a content management system (CMS) called Strapi as the backend, the data will be stored in an SQL database and the application will be hosted on Azure. The main component of this application will be search. This will work by implementing a search framework called Algolia. Search is the powerhouse behind the application. It is being used to solve the problems currently being experienced in the current student services offered by IADT.

There are multiple users for this application, each with their respective requirements. Firstly, students need to be able to find information quickly. The admins of each student service will need to be able to log in to Strapi and have full create, read, update, and delete (CRUD) functionality over their respective content. There are multiple levels for each user, there are students that may require special attention so there is a major emphasis on accessibility in this application. The admins may have multiple colleagues on their team who need access to content, and this will be achieved by assigning roles in Strapi.

As this project is built for an institution the developer must follow best practices when it comes to development so when the application is ready for production it meets the ambitious standards set by the institute.

This report details the entire process of creating this project from start to finish. The preliminary sections being research, requirements, and design, started before any coding was considered, this gave the developer a clear idea of how the project will go and a deeper understanding of the research topic being search algorithms which aided in the development of this project. This section delved into the various techniques and methods behind search algorithms and why they are essential for information retrieval. The requirements section explored the various components of the project that will be necessary, there were functional and non-functional requirements which would culminate into a final project. The developer then took a more comprehensive approach to how the application will work both behind the scenes and how it will be delivered to users so it will be easy to use, this helped bring the project to fruition.

The project used SCRUM methodologies this is shown in the implementation chapter which discusses and displays how the application was built through code and how the developer followed a structure to stay on track from start to finish. The project changed from the initial idea and the implementation chapter shows exactly how the developer dealt with these changes.

The testing section is a rigorous evaluation of how the application preforms in different circumstances; this chapter shows how the app will be used by an array of users which aided the development as the feedback was used to further improve the user experience. The application code was tested, and the integrity of the application was examined.

2 Research

2.1 Introduction

A search algorithm is a process used to find data in a collection of data. It is a fundamental procedure in computing. Often when searching for data the main factor of the application being fast or slow relies on the search algorithm. (Rouse, 2017)

The idea of searching for information you want to find has been around since the early 1990's. Since then, it has come on leaps and bounds from its origins of simply searching databases to complex algorithms and neural networks that makes finding information easier.

2.2 What is a search algorithm?

A search algorithm is a process used to find data in a collection of data. It is a fundamental procedure in computing. Often when searching for data the main factor of the application being fast or slow relies on the search algorithm. (Rouse, 2017)

2.3 How do search algorithms work?

An algorithm searches for a location, of the first occurrence of a word within a pattern of characters. During the search procedure, the characters of the given word are matched starting with the last character of the word. By starting the word match at the end of the pattern the algorithm navigates through the text in larger segments. (Mahnke, 2014)

This means that more information is gained by matching the pattern from the right than from the left. (Boyer, 1997) Imagine that the word is placed on top of the left-hand end of word pattern so that the first characters of the two words are aligned. This is the character which is aligned with the last character of the word marking the end point for the matching process. (M ahesh, 2020)

There are two categories for search algorithms, informed and uninformed search algorithms. Informed algorithms use heuristics which are rules it calculates the cost of a certain path to a solution, to make better decisions that will bring the algorithm closer to the solution by understanding which path to a solution is more desirable. (Vadapalli, 2023) Uninformed algorithms also known as blind algorithms, do not use any information to support the search it considers each path equally and without bias. (Randall, 2020)

There are multiple ways to do this process and it is important to pick the right search algorithm for a given problem. This literature review will explore what search algorithms there are and how they work.

2.4 Types of search algorithms

a. Linear Search: Is as a sequential search algorithm, meaning it starts at one end and goes through each element of a list until the desired element is found, otherwise the search continues till the end of the data set is reached. (GeeksforGeeks., 2023) This is a popular algorithm as it is straightforward because it the data does not have to be ordered but will not be the optimal solution for large datasets as it checks each element till the correct element is found, this is because each element is seen as a match. (Terziev, 2021) If there requested element is found the search is successful therefore the index of the element is returned, otherwise is has failed and no match found is what is returned. Below is a visual representation of how it goes through each element in the array.

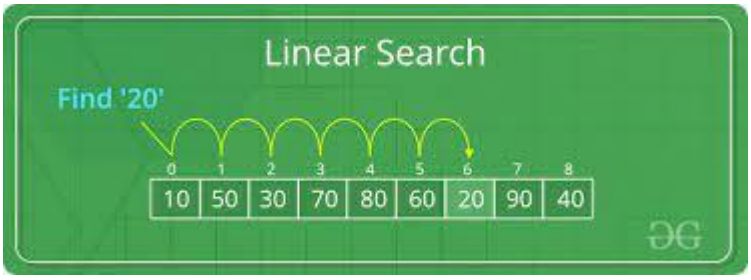


Figure 2.1 How the linear search algorithm checks element.

b. Sentinel Linear Search: This is like linear search, but the number of element checks is reduced making it a bit smarter. (GeeksforGeeks., 2023) Why? Because it uses a sentinel value that is used as a checkpoint at the end of the list. This makes sure the algorithm always searches for the right number in the list and ensures that the algorithm does not go out of "bounds." (Terziev, 2021) This algorithm is especially useful because of its improved efficiency due to the dentinal value but that requires more memory which in some scenarios can be a major drawback.

c. Binary Search: Is a searching algorithm used when an array has been sorted as it splits the search interval in two. This algorithm is different in that it will start in the middle and work out whether to go to the first half of the list or the second half. (Terziev, 2021) The algorithm does this by comparing the requested element to the element in the middle of the list if the requested element is smaller, it must be in the first half of

the list as it is ordered. (GeeksforGeeks., 2023) It will then repeat the splitting process thill the element is found. This makes the algorithm very efficient as it takes less time to find the solution, but the list must be ordered and if it is not extra memory is taken up trying to sort it. Below is a visual representation of how the algorithm splits the data in two.

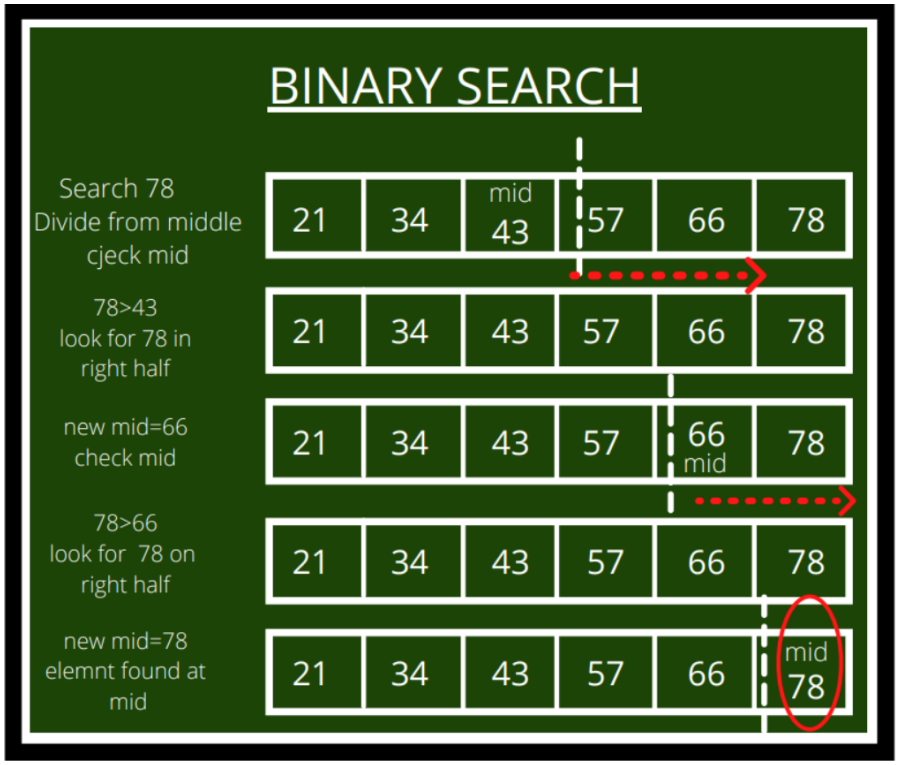


Figure 2.2 How the binary search algorithms divide the list.

d. Meta Binary Search | One-Sided Binary Search: This algorithm is an improvement to binary search; it is an improvement because it reduces the number of comparisons. (GeeksforGeeks., 2023) Instead of going to the middle of the array the algorithm will make comparisons from the beginning and end for example and from there narrow down the possible locations for the element. Like binary search it is good for efficiency but again the list needs to be sorted. It also is not most effective if the answer is not on one end of the list is the solution is in the middle regular binary search would be better. (Terziev, 2021)

e. Jump Search: The algorithm will step through the list in blocks and will search in each block. The step number represents each block, and it is usually calculated by getting the square root of the list. (GeeksforGeeks., 2023) The algorithm then goes through the blocks and compares it to the current and next block and if the element falls in between the range, it will search the current block for the element using linear

search. This algorithm is good for large datasets as it steps through the data in greater increments. Like algorithms mentioned above it requires the list to be sorted and the efficiency of the step has a considerable influence on whether the algorithm is efficient.

f. Harmony Search: This is a music algorithm, it will imitate the harmonies in music, each harmony will be stored in harmony memory. (Sultana, 2017) From this memory new harmonies will be created by bringing pre-existing ones together. This harmony will be evaluated and updated until the best solution is found. The solution will be the parameters used in the algorithm. This algorithm is good for solving music problems as it uses less maths equations which can complicate things.

2.5 Applications of search algorithms

g. Information Retrieval (IR): Information retrieval is the response of the data after data has been requested by a user. (Blair, 2003) There are multiple techniques used to improve the accuracy of the data that is returned. Search algorithms are used in each of the IR techniques explored below.

   i. Indexing: The IR system will create an index for each document in a collection of data. (Amit, 2001) This gives the data more structure which is particularly important as it impacts the accuracy and speed of the response based on specific keywords or terms.

   ii. Ranking: Priority is assigned to each document in a collection of data. This gives the results returned by the algorithm a particular order, most to least relevant. This is another useful technique used in IR. (Amit, 2001)

   iii. Querying: The process when a user requests data from an information system this is called querying. The algorithm will use the techniques mentioned above to return the best results to the user. (Amit, 2001)

h. Optimization: is trying to find the least computational cost for a problem. Search algorithms are useful in this field as it uses methods like hill climbing to find the most optimal solution. Hill climbing is the process of going towards the most optimal solution by tweaking its parameters to find the “peak” also known as the optimal solution. (Codecademy, 2023) The algorithm will start at a random point

and look at its neighbours to go to higher points. A real-world instance of this would be finding the most optimal route for a delivery driver.

2.6 Optimising search algorithms

This is the process of tweaking parameters in the algorithm for better performance leading to better results. The goal of this is to improve the algorithms' ability to navigate through the search space, the search space is all the workable solutions in a collection of data. (Engelbrecht, 2007)

2.7 How to optimise search algorithms

i. Pruning: By reducing the size of the search space by getting rid of unnecessary paths that are not worth exploring further. This can be done by using two methods, forward checking, and back jumping. Forward checking examines the entire path of a future choice before deciding and eliminates branches that violate any constraints. Back jumping goes through different paths and once a dead end is reached the algorithm will backtrack to the last path with potential desirable paths this helps remove unnecessary paths in a search space. (development, 2023)

ii. Parallelization: By now we now an algorithm is a series of steps that take place starting with an input and after some computation some output. Parallelization in algorithms is making it so the algorithm can execute multiple steps synchronously. These steps can take place on different machines or devices and the output from each step is combined to form the result. (Tutorialspoint.com., 2023)

2.8 Search algorithms in Artificial Intelligence (AI)

Search algorithms help with search issues in AI. (Javatpoint., 2022) A search issue is made up of three things the search space, start, and goal state. As mentioned before the search space is the workable solutions in a collection of data, the start state is where the algorithm starts, and the goal state is where the algorithm should finish.

AI agents streamline artificial intelligence. An agent does jobs to reach an objective and make other tasks that may lead to a desired outcome. AI agents find the best solution it does this by taking all other solutions into account. Search algorithms in artificial intelligence are used to find the best viable solutions for AI agents. (Shah, 2023)

2.9 Neural Networks and how they work.

Neural networks are a key role in machine learning and are the catalysts of deep learning algorithms. Neural networks are made up of multiple layers, there is an input layer which is what is feed to the network, some hidden layers are also in the network which basically are

functions that make computations on the previous layer and reduce the size of the data before being passed on to the next layer, lastly there is an output layer which is what comes out of the network, the data that is passed through each layer and then output can be an array of things it depends on the algorithm. Neural networks use training data to develop knowledge and increase the accuracy of the network as it learns. (IBM, 2023) Once an input layer has been passed weights are assigned to the layer. Weights give importance to every piece of data in the layer. All inputs are then multiplied by the assigned weights and then summed. Now the input layer is passed to an activation function, an activation function determines the output of the input layer, it is now passed to the next layer as an input, this is called a feedforward neural network. Below is an example of how the neural network would look.

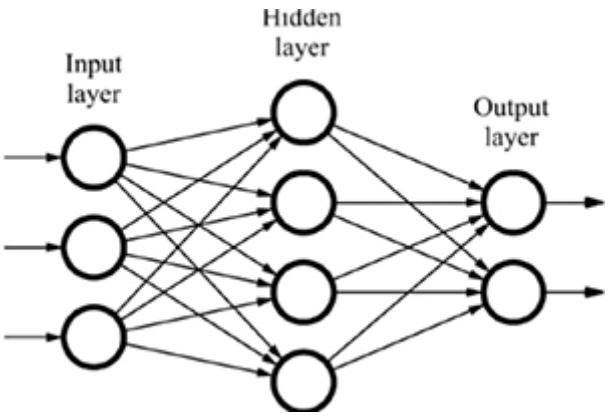


Figure 2.3 How a simple Neural Network works.

i. Neural search

This is new advancement in search, it uses artificial intelligence to create neural networks and there is a machine learning model that learns through training. (Algolia, 2023) Neural Search can decide what content should be shown to the user. Neural Search is also an advance in IR that automates the design of artificial neural networks., Neural Search trains itself to improve the accuracy of the model. This keeps developers from interacting with it manually. (Team Aisera, 2022) Neural search leverages deep learning models to improve search results. Instead of relying just on matching keywords in a data collection, neural search understands context and semantics which enhances the search. This is done by converting text into numerical representations called embeddings, the model captures meaning and relationships between words. When a request is made by the user, the system locates nearby embeddings in its data collection, ensuring that the results returned

are contextually relevant. This leads to more accurate and intuitive search outcomes in comparison to older techniques.

j. Search frameworks.

   By now you can see that there is a lot that goes on in the background when it comes to search. Luckily, there are frameworks that do the work for us they use the methods mentioned above and provide developers with a streamlined way of implementing search. Notable frameworks at the forefront of search are Algolia, Bloomreach and Coveo.

2.10 Summary

The area of search and its inner workings is huge. There is a lot of work that goes on behind the scenes that is constantly evolving to improve the area in metrics like accuracy and efficiency.

From where search started to where it is now there has been a lot of change and it will continue to evolve with new advancements in neural networks being the powerhouse for new developments.

Search is an integral part of most people's lives it is used every day for both simple and complicated tasks. Sometimes users do not even know they are using search.

3 Requirements

3.1 Introduction

In this chapter, the author will discuss how the application will be built and what is required. This chapter will look at similar applications to show the good things that will be replicated and the parts of the application that are not good will be avoided. There is also a section where the requirements are modelled, this will focus on the different users that will use the application. After those sections, there are functional and non-functional requirements that also tie in with the testing section. This will paint a picture of how the application will work in its most basic form and how it will be tested. Lastly, there are feasibility and conclusion sections which dive into how attainable the project is and how the requirements section affects the project.

3.2 Requirements gathering

3.2.1 Similar applications

This application is being developed for IADT (Institute of Art, Design Technology). To find similar applications it is relevant to look at other college websites and see how they deliver a similar product to the IADT student services. There is a common theme in each application, the search for information a user is looking for is not as efficient as it should be. There are a lot of links on each page that bring you to another page with a link. The overall feel of the applications is not consistent, there is no tone that invites users to use it.

Some applications like the student services for The National College of Art and Design have a completely distinct experience, for example, the first page you see is simple yet inviting. The pages for each service do not have a lot of text and they have concise headings which make it easier to find the content you are looking for.

Figure 3.1 The University College Cork student services page.

This is a page with a lot of links, the search is not prominent and does not search for content in student services just the whole website.

Career Services

The **Career Services** exists to support the career development of UCC students through the provision of advisory, work placement and graduate recruitment services.

Their mission is to help UCC students to achieve their career objectives by providing access to one-to-one careers advice and coaching, employability skills development classes, workshops and events, work experience placement opportunities and employment or postgraduate opportunities in Ireland and abroad.

The Career Services are based on the Second Floor of the Hub, with advisors having rooms on the Ground Floor.

Disability Support Service

The **Disability Support Service** (DSS) in UCC provides supports for students who have a disability, specific learning difficulty or who entered UCC through the DARE scheme.

If you did not apply through the DARE scheme but are wondering if you can register with the DSS, please check out www.accesscollege.ie for a list of supported disabilities and relevant documentation required.

The DSS is based on the First Floor of the Hub.

Figure 3.2 Example of a text heavy student services page

This text on this page is ok but there are another eleven sections on the same page just like this which makes the page too text heavy.

Figure 3.3 Cards that show the supports offered

The homepage has a warm feeling, there is consistency in the layout and design which makes it more inviting to users.

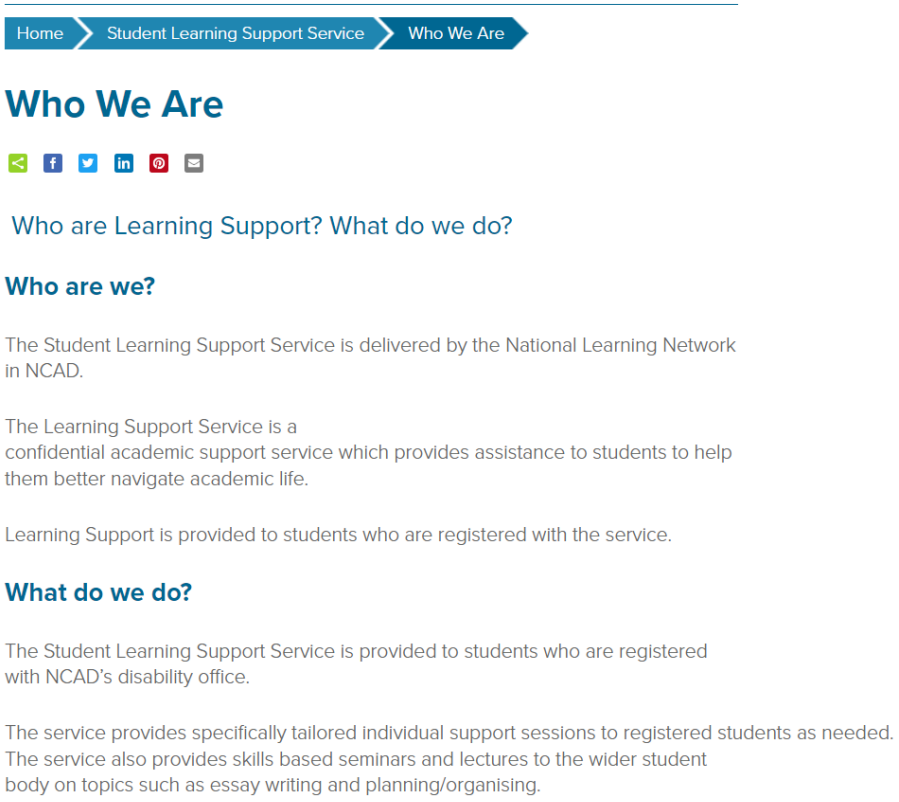


Figure 3.4 Information page with good structure

Simple yet effective services page. Instead of having large bodies of text, there are simple and informative short sentences but there is still room for improvement as the colour of the text is hard to read.

This section shows that no application is perfect, each has its pros and cons and will require constant development to cater to the service's needs.

3.2.2 Student services website for IADT

The front end is created using a react framework, the code is structured in a way to improve efficiency and maintainability, by having files for pages and separate files for components that will be imported making the code cleaner. There are many packages used in this project, but the main package is Axios, Axios is used to interact with Application Programmable Interfaces (APIs). By using this package, the front end will be able to retrieve information from Strapi. The application will be designed in such a way that users find what they need quickly, it will be a responsive mobile-first application that works well on any screen size.

Strapi lets users add content and the front end dynamically displays the content through API calls, but the application requires the content on the website to be searched by users. Algolia solves that by searching the content that is in Strapi, there is a pipeline between Strapi and Algolia so when a user uploads content in Strapi it is then sent into the Algolia index and now on the front end, the students can search for information. All the information is stored in an SQL database which is not interacted with directly.

An issue in this project is the bookings that users will need to make, since the users are all in the same Azure directory ideally a user will be able to log in using their student emails, the problem here is that IADT has certain permissions on the directory that may restrict this functionality. There is also a lot of development behind creating a booking system, a solution to this problem would be to make users make bookings through Outlook on a person's calendar on the student services. This will require each person in the student services to make a booking page in Outlook.

Each of the technologies mentioned above is hosted except for Algolia. Azure provides the hosting. There is a container that has the application stored in it, the front end, CMS, and database. The container hosted on Azure allows for more security and scalability, it also streamlines the deployment, because now when a change is made to anything in the container it automatically redeploys.

3.2.3 Survey

As you can see above there are lots of requirements, to gather more data surveys were conducted, and admins of each service completed this survey. The goal of the survey is to understand each service's needs and from that the project decisions going forward will be more informed.

The following are the responses from the survey.

1. What is the name of your support service

6 Responses

ID ↑	Name	Responses
1	Suzanne Keily	Student Health
2	Brian Keane	Student Learning Centre
3	Carly Salter	Student Counselling Service
4	Loreto Meagher	Student Learning Centre
5	Dawn OConnor	IADTCareers (all one word like that please - it is how I have it on my Careers Portal!)
6	Sinead Mc Entee	Access Office

Figure 3.5 Name of service. Question 1

The first question was an easy one to get started. This information is important because now we have established the name of each service and the number of services. This shows scalability is important, if another service is added there needs to be a straightforward process to set it up quickly so users can interact with it on the application.

The second question is simple but provides a lot of information, now the contacts for each service are being set. This not only gives the contacts but also how the application might be structured so that users can find this data easily.

2. List main points of contact for that service

6 Responses

ID ↑	Name	Responses
1	Suzanne Keily	Suzanne Keily,Institute Nurse contact details studenthealth@iadt.ie or 012394460 Health center administrator (no name yet as new person being appointed) same contact details.
2	Brian Keane	Loreto Meaghar- Chartered Psychologist. Brian Keane- Assistant Psychologist
3	Carly Salter	Person 1: Name: Carly Salter Pronouns: she/her Contact: studentcounselling@iadt.ie - no phone number
4	Loreto Meagher	Loreto Meagher Chartered Psychologist loreto.meagher@iadt.ie 01 239 4790 Brian Keane Assistant Psychologist learningdevelopment@iadt.ie 087 102 3215 (9am - 5pm)
5	Dawn OConnor	Name: Dawn O'Connor Role: Careers and Employability Email: dawn.oconnor@iadt.ie Phone Number - 01 239 4670
6	Sinead Mc Entee	1. Sinead McEntee Access Officer 01 2394628 Sinead.mcentee@iadt.ie or access@iadt.ie 2. Tracey Morgan Access Office Administration Support 01 239 4628?? tbc tracey.morgan@iadt.ie

Figure 3.6 Points of contact for each service. Question 2

Now the questions will be more open, which gives each service a chance to express what they think needs to be relayed to the users, this will set the tone for each service and again it helps develop the application in both data collection and overall application structure. The main takeaway from the response to this question is adaptability.

3. This support service can help you with... / Reach out to us if you need help with...

6 Responses

ID ↑	Name	Responses
1	Suzanne Keily	We are here etc., help you with 1.Arranging an appointment to see the Doctor or Nurse. 2.Treating any injuries or accidents you require treatment for. 3.To discuss any medical issue you may have and designing a treatment plan for you.
2	Brian Keane	- Provides psychological learning support to student. -Provides academic learning support to the student to allow them to become independent in their learning. -Delivers workshops throughout the trimester to assist students with their learning experience (Time management, stress and wellness programs, academic writing, coping with distraction) -Assist students experiencing difficulties related to ADHD whether through assisted learning, referrals and screenings, offering tools and resources to allow them to address certain difficulties.
3	Carly Salter	Student counselling provide short-term psychological support to any registered IADT student. Student counselling operates on a six-session model, with signposting to external services as appropriate. Student Counselling is located in the Carriglea building, in room C006a. We are also available online through Teams video call. An initial appointment is an opportunity to further understand your present circumstance and to explore which supports you may need. Students attending Student Counselling frequently present with concerns of anxiety, low mood, family and other relationship dynamics, grief, gender and

Figure 3.7 What can the service help with. Question 3

This ties in with the previous question but the responses to this question are shorter. This is important because keywords can be extracted from the responses, which can be used for tagging content. Tagging content can help with things like filtering and more importantly searching.

4. List the main supports you offer

6 Responses

ID ↑	Name	Responses
1	Suzanne Keily	Sexual health Mental health General physical health Dressings and administration of medication/injections We do blood tests.
2	Brian Keane	- Psychological support -Academic support -Person centered support surrounding their academic and psychological needs. -Provide support to the overall team and department and ensuring good communication between them when supporting a student.
3	Carly Salter	One-to-one counselling sessions Referral to external medium-longer term supports
4	Loreto Meagher	• Getting Started on your Assignment • Managing Procrastination • Breaking Down Your Assignment Brief • Essay Writing • Thesis Proposal and Thesis Writing Skills • Research Methods and Finance modules • Organisation and Time Management • Notetaking • Presentation Skills • Critical Thinking Skills • Exam Preparation
5	Dawn OConnor	Building your Personal Profile (All years and all interest areas) Knowing your competencies / skills (All years and all interest areas) Creating your unique path - preference assessments (All years and all interest areas) Employer events (All years and all interest areas)

Figure 3.8 Main supports offered. Question 4

This is an app for students, but students are a large group by asking who is eligible for the support the students can be divided into subgroups and now the application can cater to their specific needs, how? This can be done by ensuring there is a flow for every type of user so they can reach their goals as quickly as possible.

5. Who is eligible for your support?

6 Responses

ID ↑	Name	Responses
1	Suzanne Keily	Any registered student is eligible to attend Student Health including medical card holders Eramus students
2	Brian Keane	Any person registered as a student with one of the courses at IADT.
3	Carly Salter	All registered full-time, part-time and certificate course IADT students. No diagnosis necessary.
4	Loreto Meagher	All IADT students are welcome to use our service. We also work with lots of neurodiverse students including those with Autism, ADHD, Dyslexia and Dyspraxia
5	Dawn OConnor	All students, graduates and staff of IADT.
6	Sinead Mc Entee	Students who enter via the HEAR/DARE route, QQI route, mature students or other students who identify themselves as belonging to a target group (listed above) The student assistance fund is open to ALL students to apply who will then be assessed on their individual financial circumstances.

Figure 3.9 Who is eligible for the support. Question 5

Like before there will be an element of adaptability, not every service will be the same which is a good thing but how can the application meet the requirements of each service without restricting them from creating what they want? Again, it goes back to structure, the question is there additional information that your support page needs, and its responses enforce the idea of flexibility. For example, the author can add multiple sections of content, but they must not exceed five hundred characters. This lets the author create what they want but supports consistency throughout the application for each service.

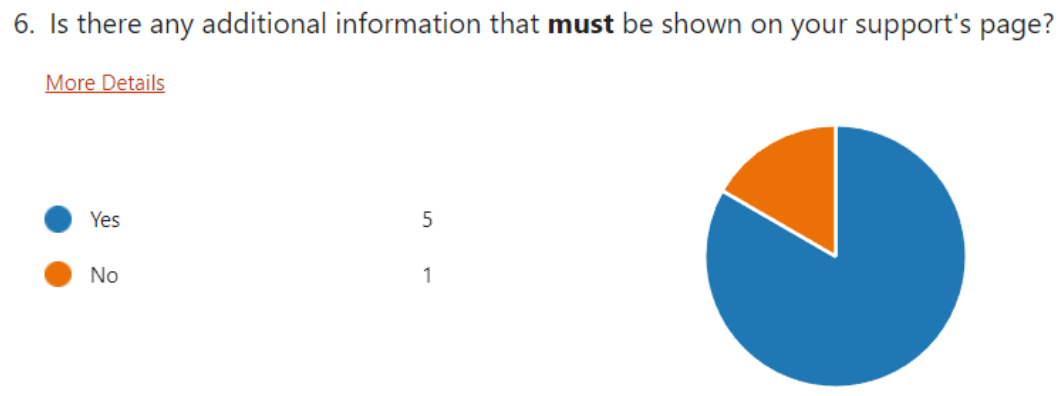


Figure 3.10 Is there additional information that must be shown on the support page. Question 6

All this information is beneficial now when building the application there is little to no guesswork when meeting requirements. The surveys have aided in the development process

and will be a constant point of call when working through current ideas proposed for the project. Now the challenge of understanding the different users and how they will use the app, the next section will go into personas which will solve the problem of user flows.

3.3 Requirements modelling

3.3.1 Personas

Figure 3.11 Persona picture 1

Christian O'Keefe 1st year business student

Christian is an 18-year-old first-year student in IADT who has autism and ADHD. Christian lives close to the college and plans to cycle to campus, He has not met his classmates yet and knows no one in the course. His first class is in the Carriglea building in the college, but he has not yet visited the campus, so he does not know his way around.

Christian would like a way to find out all this information, finding services that apply to him is also important to him. He does not like reading large bodies of text as he finds it overwhelming. Lastly, he needs to be able to go on the app whether he is on his phone or his laptop without going through the IADT website every time.

Solution: These measures will be implemented to meet Christian's needs. There will be a map of the college that is clearly labelled with each major location, things like lecture buildings bike stations canteens will be shown on the map. The application homepage will be up to date with what is happening on campus things like first-year focus will be featured content so Christian can maybe meet friends and socialize with his peers, there will be an entire page dedicated to student services/supports so Christian can find them quickly, the content on the pages will be summarized so Christian can easily find what he is looking for, he can also use the search box that will be at the top of each page to help find content. The app will be a mobile-first app so he will be able to download it on his phone from the browser.

Sara Walsh 3rd year psychology student

Figure 3.12 Persona picture 2

Sara is in her penultimate year in IADT, she loves the college and finds her course remarkably interesting, many of Sara's friends went on placement or Erasmus last year but Sara could not find a placement and felt she would get homesick if she went on Erasmus, so she stayed in IADT and completed a project instead.

Sara is looking to do a placement this year, but she is worried she will not get one, she believes that her CV is not up to a good standard and since she has not any previous experience, she is finding it hard to get offers. Because of this she has been stressed about college, and it is starting to influence her grades, Sara would like to get help so she can finish the year strong.

Solution: To help Sara get the best of her time at IADT she will have to contact the appropriate services/supports when Sara searches for "career help" she will see the careers service and be able to book an appointment so she can get help with her CV and placement opportunities. Now the problem of her grades, Sara needs to contact her year tutor and tell them about her situation so they can figure out how to fix it effectively. There will be a list of each course in the college and in that list, she can find the year's tutor's contact information so she can reach out.

James Taylor senior lecturer and student health coordinator

Figure 3.13 Persona picture 3

James has been a lecturer at IADT for 12 years now, he is very enthusiastic about his job and loves helping students. He plays a significant role in making sure students understand the importance of health and thinks that students need to visit the student nurse more when they have problems.

James is frustrated at how long it takes to change the content on the student health page as he must ask the IADT admins to go in and make changes. He would like to update the content so it reflects the latest research and recommended guidelines, he also wants to add in a new section that he brainstormed with his team, it is a quick health tips section that aims the give quick health tips that students may not know. Currently, if a student wants to see the student

nurse, they must email James would like to have a way for students to book appointments with the student nurse.

Solution: James will have direct access to the CMS so there is no intermediary he has CRUD functionality over the content, there will also be a booing page where students can book appointments that he can customize to his needs.

Sylvia Jones Junior Lecturer student health writer

Figure 3.14 Persona picture 4

This is Sylvia's second year as a lecturer, and she has been added to the student health team. She is extremely excited about the challenge and is enthusiastic about her role as a writer.

Sylvia must send her new articles to James to be approved and James is busy with teaching and other duties, sometimes it takes him a while to see it and he has not yet made any suggestions or changes to her work, so the process is redundant.

Solution: James will be an admin in the CMS. The admin can assign roles to users, Sylvia will be assigned the writer role so she can go in and create or update content on the page now the process is more streamlined.

3.3.2 Functional requirements

Requirement ID	Requirement Definition
Admin	
FR.1.0	CRUD content in CMS
User	
FR.2.0	Search Content on site
FR.4.0	Make a booking to service
FR.5.0	Download the website as an app

3.3.3 Non-functional requirements

Requirement ID	Requirement Definition
FR.1.0	Elevated level of accessibility
FR.2.0	Good user experience
FR.3.0	Appealing design
FR.4.0	Responsive layout

3.3.4 Use Case Diagrams

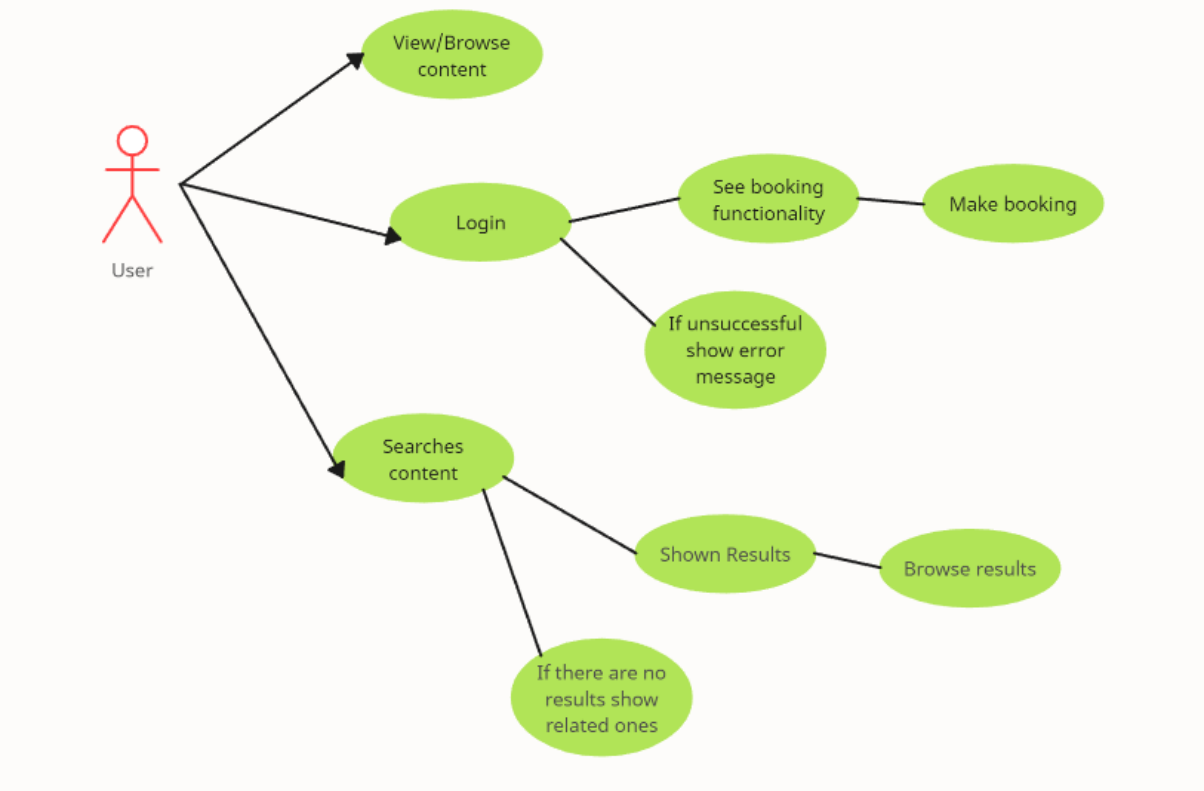


Figure 3.15 Use case diagram showing user flows

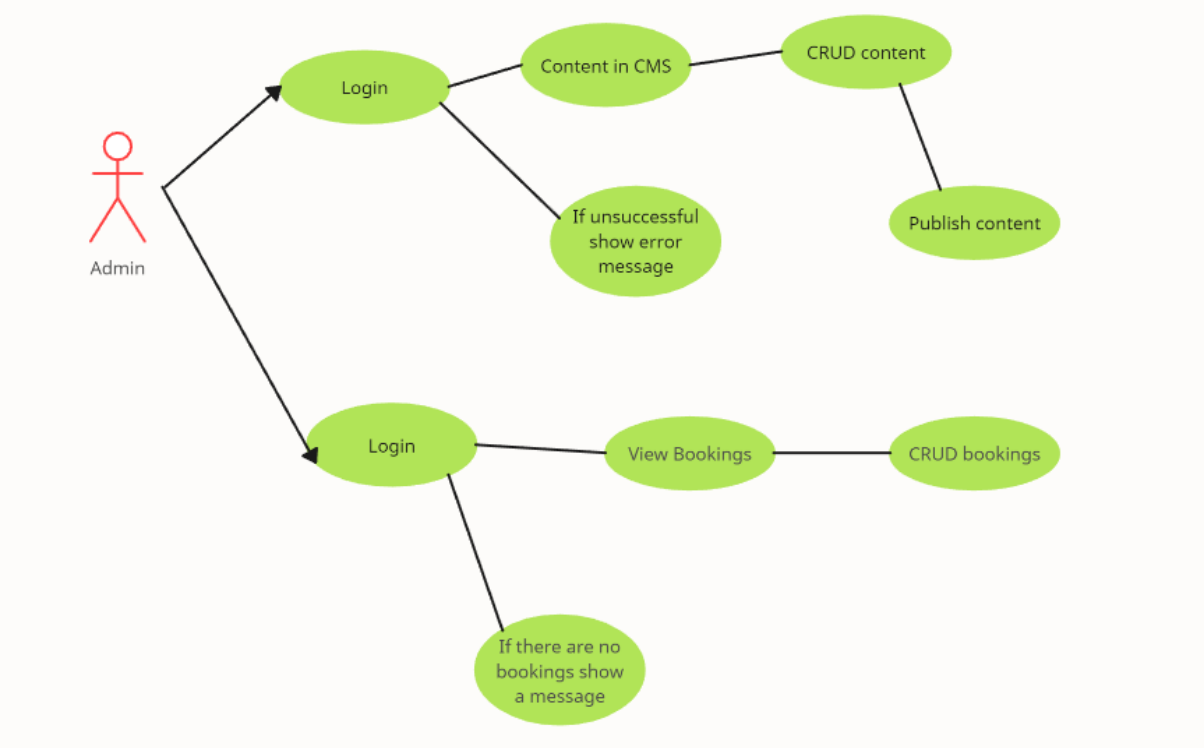


Figure 3.16 Use case diagram showing admin flows

3.4 Test plan.

Testing the application is going to ensure it is ready for production. The application needs to be of a high standard as this is an application that will be used by a lot of people and be representing IADT. To maintain the high standard the application will be tested by examining user interactions and user feedback and there will be unit tests that will test the code and prevent any bugs from occurring so there are no errors in production.

3.4.1 User testing

User tests will be conducted continuously once the application has been developed to an acceptable standard, an acceptable standard is going to be when the functional requirements are implemented and ready to be tested. The user testing will consist of the developer asking a user to complete a task as they watch and record the results, the user will be asked to talk through each of their actions as they try to complete the tasks. The developer will also ask the user questions after the tasks are completed to get an understanding of how they think the app works and any changes that should be made.

3.4.2 Unit testing

There are several functional requirements that the application relies heavily on, to ensure the application runs smoothly the features need to be tested before they are used by users, the

unit tests will play out different scenarios that a user might take to make sure whatever actions are taken do not result in errors.

3.4.3 Test cases.

Test Suite ID	TC01
Test Case ID	TC01
Test Case Summary	Users searching for content
Related Requirement	FR2.0
Prerequisites	Algolia Framework integrated on the front end
Test Script / Procedure	Select the search bar, type in a keyword
Expected Result	The user should be shown the content related to what they are searching and be able to view the resource
Created By	Tariq Horan
Date of Creation	16/01/2024

Test Suite ID	TC02
Test Case ID	TC02
Test Case Summary	An admin using the CMS for the CRUD functionality
Related Requirement	FR1.0
Prerequisites	Admins have access to the hosted version of CMS
Test Script / Procedure	Create content, read content, update content, and delete content
Expected Result	The admin should be able to CRUD content with no errors
Created By	Tariq Horan
Date of Creation	16/01/2024

Test Suite ID	TC03

Test Case ID	TC03
Test Case Summary	Users will log in using their student emails so they can see the booking page
Related Requirement	FR3.0
Prerequisites	The user has a student email
Test Script / Procedure	Click log in button and fill out the details
Expected Result	The user will log in with no errors
Created By	Tariq Horan
Date of Creation	16/01/2024

Test Suite ID	TC04
Test Case ID	TC04
Test Case Summary	A user will make a booking to a service
Related Requirement	FR4.0
Prerequisites	The user has logged in using the student email
Test Script / Procedure	Find the desired meeting type and time them to make a booking
Expected Result	The user will make a booking to a service successfully
Created By	Tariq Horan
Date of Creation	16/01/2024

Test Suite ID	TC05
Test Case ID	TC05
Test Case Summary	Download the website as an app
Related Requirement	FR5.0
Prerequisites	The user has access to the website
Test Script / Procedure	Click the download button built into chrome
Expected Result	The app will be downloaded
Created By	Tariq Horan
Date of Creation	16/01/2024

3.5 Feasibility

This project relies heavily on search as that is seen as the primary requirement, this will be done through Algolia. Algolia has a free tier that will be used to develop the project and will be what is used in the final application if the free tier limits are not exceeded, this makes the project more feasible because if Algolia was not free another search framework would have to be considered. Algolia will have to communicate with the CMS to get content, the developer has done some research into how the content in Strapi will be put into Algolia, and some tutorials and videos show how to achieve this. Creating a booking system is quite comprehensive and will take a big chunk of the project development time, an alternative to this would be an external booking service that lets users make bookings to service in IADT, another solution would be to use the Outlook booking pages that are built into outlook, some of the members of the college already use this feature in Outlook.

3.6 Conclusion

The sections in this chapter dived into the relevance, importance, and expectations of the project. This chapter has provided a deeper insight into how the project will work and the overall requirements of the project.

The steps needed to complete the project are clear because of the planning and data gathering done in the preliminary stages which took place well before any technical development so now the process will be easier. All the steps taken have strengthened my understanding and confidence in completing this application, errors and complications will arise, but they are expected.

4 Design

4.1 Introduction

Many features will need to be implemented for the project, for all these features to work well there is an emphasis on how the application is designed before development. This section will take both the research and requirements sections into account when designing the application.

4.2 Program Design

4.2.1 Technologies

The technologies being used to create this application are:

- React JS: This is going to be the frontend library. This library was chosen because the developer has previous experience and knows the library well, this will make the development process easier and allow for more time to be allocated to other features that are more challenging. Another major focus in this project is accessibility, the developer knows most of the best practices when it comes to accessibility and there is also a lot of documentation in this area.
- Tailwind CSS: The framework that will be used to design the front end will be Tailwind, Tailwind is a design framework that will allow the developer to make the user's experiences better. Tailwind has an abundance of documentation and allows developers to create striking interfaces that are inviting to users.
- Strapi: This is the content management system where the information will be retrieved. Strapi is built using Node.js which works very well with React. This is a popular framework used to build applications it has a built-in backend that handles operations like logging in. The CMS is very customizable and through research can meet the requirements of this project about content manipulation and plugin integration.
- Azure: Is a cloud computing platform made by Microsoft that offers services like web hosting and it also allows applications to use Microsoft single sign-on services. These services are requirements for the project and this platform allows the React front end and Strapi backend to be hosted and communicate with each other. There is also a database that will contain all the content uploaded which will be hosted on Azure too.

- Algolia: The search framework that users will use to find content on the application. This is easy to use, fast, and scalable the framework has a very generous free tier that allows an application to make 10,000 requests per month which is plenty for the project.

The developer picked these technologies because they are compatible with each other, the other technologies that were considered would have made the application break as they cannot work together due to technology conflicts like deprecated packages. The other framework considered was Vue JS for the front end, this is an exceptionally good front-end framework that does most if not all the functions React does but the developer's knowledge in React makes the project more feasible. There were several CMSs (Content Management System) considered like Payload and Directus, the developer made a sample project in each CMS and found that for the project Strapi was the best option. Another cloud-based platform that offers a lot of the same services as Azure is Amazon Web Services (AWS). AWS is a great platform that the developer is familiar with, but the Azure student tier offered more credits which is ideal for the project as it allows for more development. Azure is currently used by IADT so this application will follow the current environment.

4.2.2 Structure of React/Strapi/Algolia

The front end is built using React, many front ends are built using frameworks but React is a library that allows developers to create user interfaces using JavaScript. React applications are single-page applications (SPA), React creates a virtual Document Object Model (DOM) which is a copy of the real DOM this is where you will see the application, any time there is a change made like a component changing or if a new page is requested, React will create a new virtual Dom and once the changes are identified the real Dom is changed. A component changing refers to its state which is the component lifecycle, React uses hooks, which allow function components to have access to state and other React features making the code cleaner as there is now no need for classes. Common hooks that are used in React are useState and useEffect, useState is a hook used to manage state in functional components, while useEffect is a hook used to manage side effects (like fetching data, setting up event listeners, or updating the DOM) in functional components. One of the problems with a SPA is that there is no built-in routing it needs to be handled by a third party like React router. React Router lets the developer manage the URL and the components that should be displayed with the current URL.

Strapi will be used as the CMS in this application, a CMS will make it easier to manage the data that will be used in the application, Strapi is unique because it is built using Node.js, Node.js is used to create server-side applications using JavaScript making the integration with the frontend more streamlined as they are both JavaScript. The CMS being built on Node.js means it can also be used as the backend for the application. Strapi is a headless CMS which means that the content is returned using an Application Programmable Interface (API) an API is a way to communicate with the database the interact with the data, by getting the data through the API the developer has full control of how the data is displayed on the frontend. This allows the application to be more flexible and scalable as it is not restricted to CMS rules. The users uploading content will be part of the student services group and they will need to have access to the CMS, Strapi provides users with an inviting interface that can be customised by the developer which makes uploading content a less stressful experience. Since there are multiple people in some of the groups in the student services the manager or head person in each group will need to be able to restrict users in the group as they will not have access to all the features, this can be achieved in Strapi by using the roles feature. Data storage is an integral part of this project and Strapi creates the database itself which ensures all the relationships are secure, the database can also be altered to meet the project requirements.

The major focus of this project is search, Algolia is a framework that implements fast and reliable search functionality that would take a lot of development to create from scratch. Algolia has a search engine that searches through datasets and finds results, there is a multitude of features and plugins that can be used in Algolia one that will be used for the project is instant search, this feature will show results based on what is typed in the search box as the users types the results are updated accordingly. Algolia has a dashboard where the developer can see the data that is available to be searched, this data is coming from Strapi so when content is uploaded the content is put in the Algolia database and each piece of content is indexed so it can be searched. Algolia also offers analytics that provide feedback on how the content is being searched.

4.2.3 Application architecture

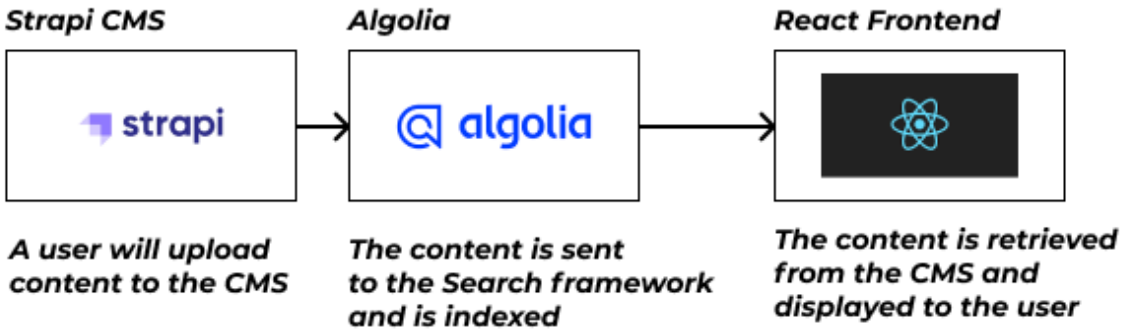


Figure 4.1 Simple application architecture

The architecture of this application is simple as the technologies that are used do a lot of the heavy lifting, as mentioned above the CMS can also be used as a backend for the application, although it will not be necessary for this project the backend is still present in Strapi it just will not be used. Algolia does several different comprehensive actions, for example, the framework implements Natural Language Processing (NLP) which would take an extensive portion of the development time, through using the framework more time can be allocated to other aspects of the project like front-end development and testing. React will be used to create the front-end and it will be styled using Tailwind in conjunction with CUBE principles, in the front end there will be a search box that has the Algolia functionality integrated with it there will be continuous development and testing for this feature.

4.2.4 Database design

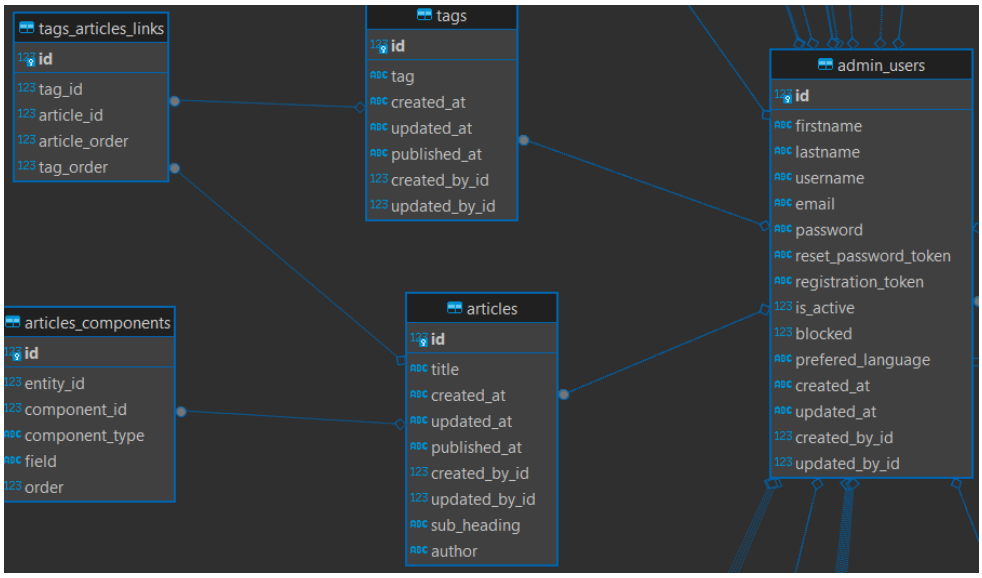


Figure 4.2 Database design

This is a snippet of the tables in the application. When creating the project Strapi will create the database automatically, this is particularly useful but there are a lot of tables that are not relevant as they are never used. The tables shown above are important as it shows the connection between a user and the content, a user can be related to many documents. Strapi lets developers name the collection type the collection type is the tables shown above. The article table is going to be the main content shown in the application, the ERD (Entity Relationship Diagram) shows that an article can be related to many tags. At this stage of the project there is few tables but as the project develops there will be more tables and more relationships.

4.2.5 Process design.

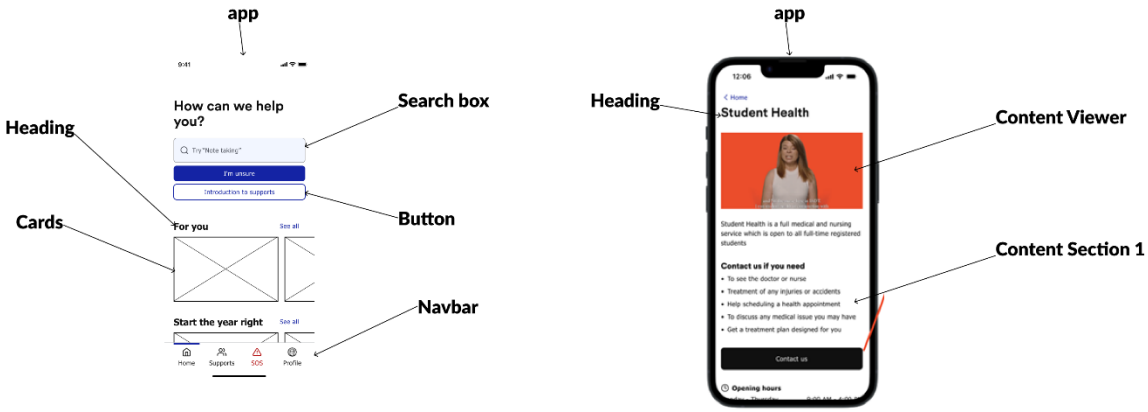


Figure 4.3 Annotated prototypes highlighting components

The image above is a high-level annotated prototype, these annotations give the developer a better idea of how the overall application code will be structured. The application will be built using React's best practices, annotating the prototypes lets the developer decide what elements of the application will be reusable components that will be in a separate file to the page file.

From the annotated prototypes there is a clear sign that there will be many components that will be re-used throughout the project like the cards and the buttons.

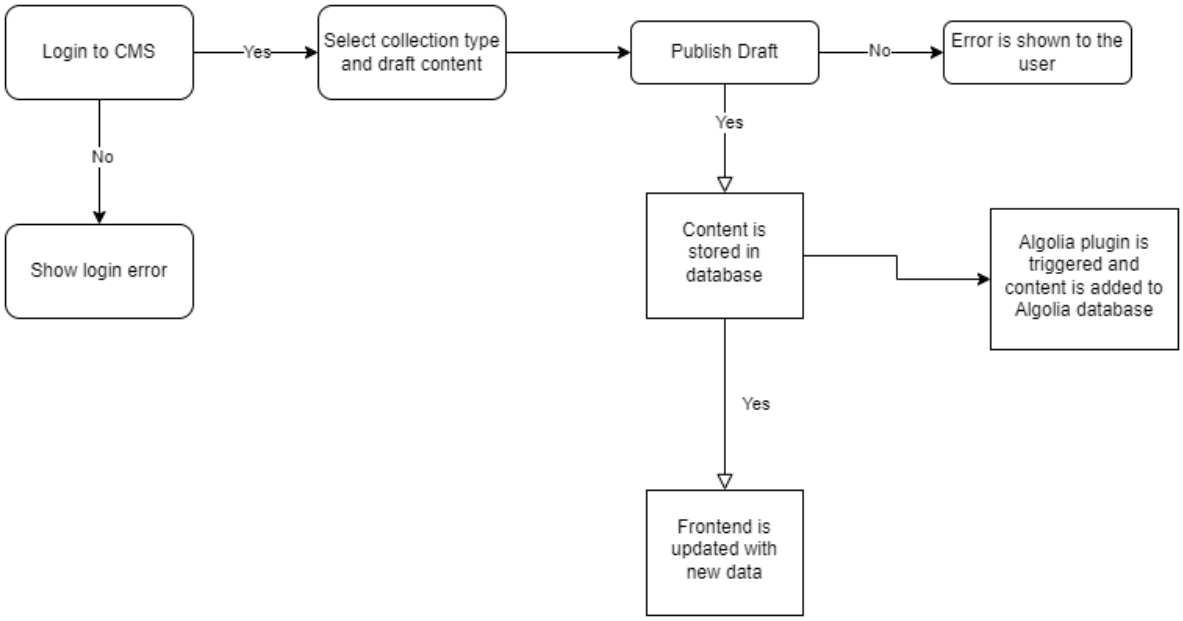


Figure 4.4 System diagram for content added by admins

The diagram above demonstrates how the system will work behind the scenes, A user will access the CMS with their personal details, if they reach an error like entering the wrong password it will be displayed as they fill in the login form. When a draft is created in Strapi it will be saved in the database and the user will need to publish the content for it to be accessible in the API and be sent to the Algolia database, if there is an error saving or publishing the error it will be shown for example if the user forgets to fill in a field there will be an error that prevents the content from being saved and display the error. If the content is saved or published it will be stored in both the Strapi and Algolia databases, the frontend will be updated with the new data.

4.3 User interface design

This section describes how the interface is designed. The section will differ depending on whether an app or a game is being developed.

The application will be used by an array of users from students in and outside the college to students who are neurodivergent, this means that the user interface needs to be inviting and accessible to all users. This will be done by not overcomplicating the UI with overstimulating transitions and components, below is an example of how the first prototypes.

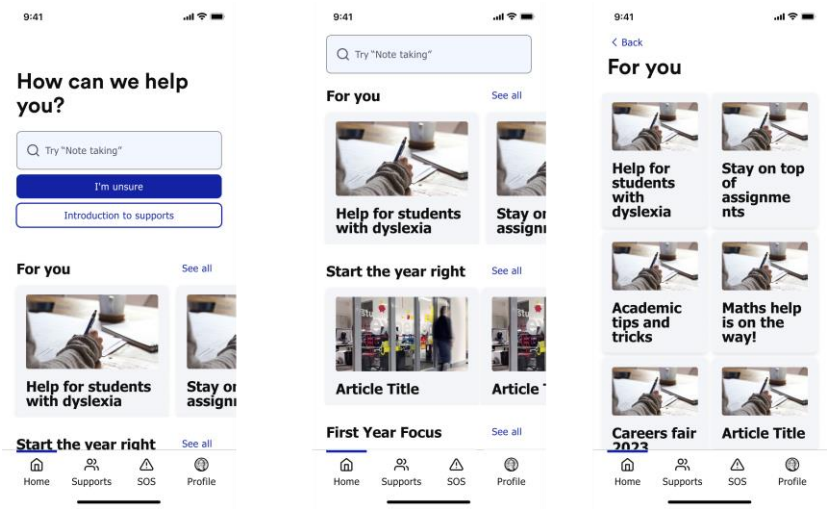


Figure 4.5 Initial wireframes

The design above is basic the layout is not going to change too much but the colours and position of components will change, it is currently too muted and looks very grey. The focus in the design phase is the search. The search box needs to be in a place that is easily accessible and relevant. In the prototype when the search box was selected a user could just type but this caused users to become frustrated as they could not take the focus off the component. Below is an updated prototype that addresses these issues.

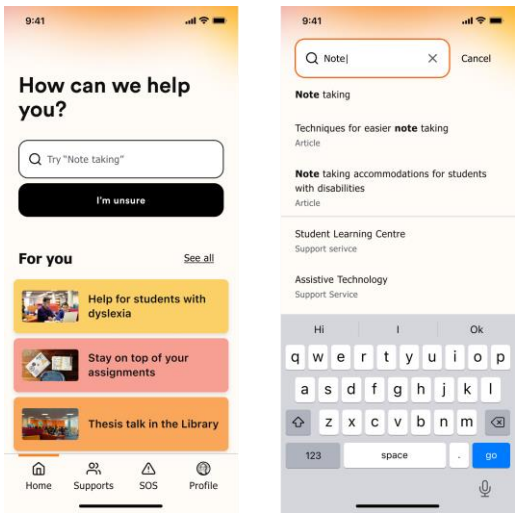


Figure 4.6 Wireframes with more colour

Now the application is more inviting there is a colour gradient which makes for a nicer user experience, now when the search box is selected it opens and takes up the whole page which solves the problem of a user not being able to take the focus off the component. This will be a mobile-first application meaning that it is intended to be used on mobile but the application will also be fully responsive so it will look presentable on all screen sizes. This will be a working prototype that will be

used in the development process and once created the prototype will be tested and refined till it reaches a standard that reaches all the project requirements and satisfies the developer.

4.3.1 Wireframe

Below is a basic wireframe that was made in the preliminary stages of this project and was the first piece coded in the application as a proof of concept.

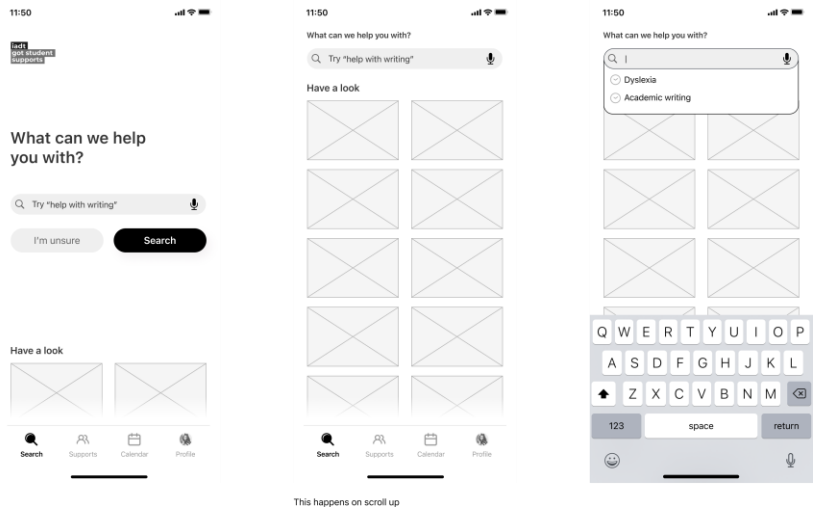


Figure 4.7 Wireframe recreated in React for proof of concept

4.3.2 User Flow Diagram

This shows the intended flow for the user in the application, the diagrams will focus on two users one will be a student accessing the application and looking for information and the other will be a member of the student services adding content through the CMS.

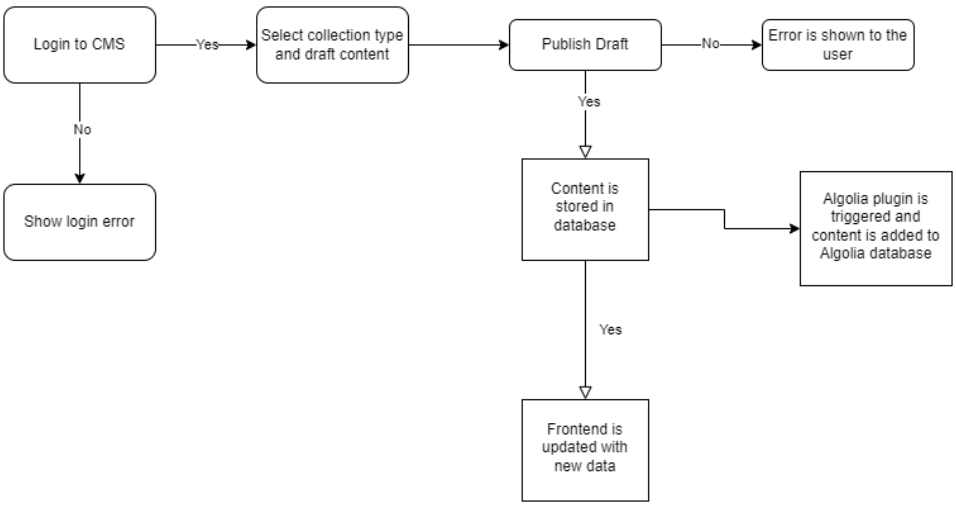


Figure 4.8 System diagram for content added by admins

As shown in the diagram below there is an expectation that users will not be able to find what they are looking for in the event of this happening the user will use the I am unsure function that will point the user toward the relevant services.

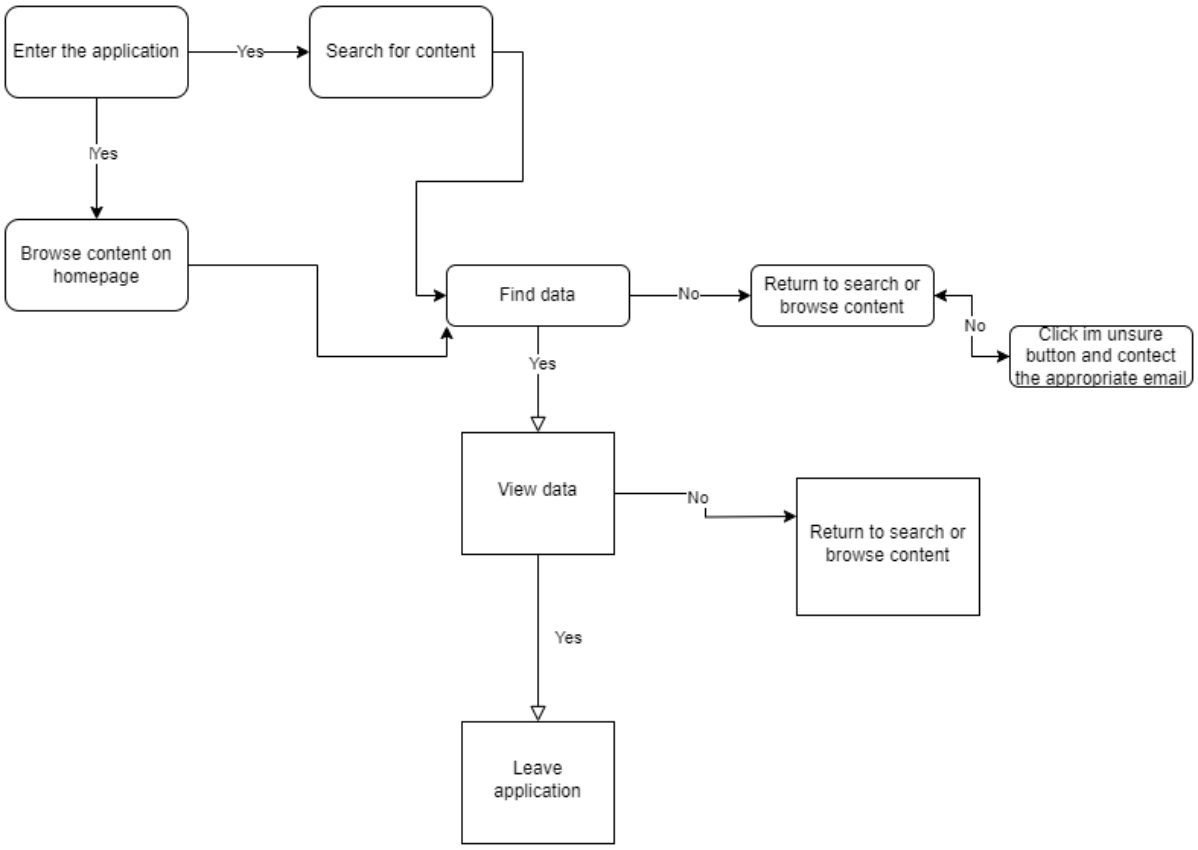


Figure 4.9 User flow for users showing multiple scenarios

4.3.3 Style guide

As mentioned above the application needs to be accessible a major facet of accessibility is typography the typography will be clear and concise making sure the content in the application is easy to read. All headings will be aligned to the left so there will be consistency throughout the application the content will also be aligned to the left except for a few exceptions in particular cases like cards for example.

The colours used in the application will be light colours like yellow and red in a neutral tone, so the colour does not become too distracting and distract users. The UI will have a gradient in the background on some of the pages and the colours will be used sparingly on components like buttons.

Icons will be used in the navbar component on top of text to help users find what they are looking for they will not be used anywhere else in the platform as some users may not be familiar with them.

4.4 Conclusion

This chapter has provided the developer with more clarity as the way this application is now outlined in detail with the previous chapters considered, the technologies that were chosen are also discussed in this chapter and a prominent level of understanding has been established.

5 Implementation

5.1 Introduction

This chapter will uncover the development process for this project, there will be a clear procedure in place so when developing new features there will be a structure to how it is created. The project will follow Agile methodologies, this project management approach makes the development process easier, a customary practice in this methodology is the SCRUM approach this approach focuses on dividing the project into small milestones called sprints. This section will discuss how the project will implement these methods.

5.2 Implementation Roles

The development will be done by the developer, the design of the front end will be done by external designers, and the designs will be created using Figma and shared with the developer.

5.3 Scrum Methodology

As mentioned above SCRUM is an agile framework that is used to keep the project on track by letting the developer self-manage the development and constantly evolve the development this approach will save time and keep the developer on schedule as there will be bi-weekly sprints. A sprint is a period where the developer will complete tasks and clear the backlog of features. A backlog of features will be tracked using a Kanban in a Miro board, following the to-do, doing, complete style, the backlog of features will be all the features in the to-do section of the Kanban, everything in the doing section is what is currently being developed and the completed section is all the features that are completed. At the start of each sprint the developer will select several features to be completed by the time the sprint is finished, each sprint will have a different number of features as they vary in complexity. At the end of each sprint, the developer will review how it went and show the project supervisor the progress that has been made.

There will be 7 sprints in total, each sprint will be split into two sections development and documentation. The developer will be making components and developing the application using code each sprint and document every action they take in the development this will show how the features that are coded have been made and how they will be used in the project.

Figure 5.1 Scrum methodology visual representation

Above is an image of how the process will work. The plan period will be where the developer picks what features to develop, the design and build phase will be the development step where the developer creates the components. In the test phase, the developer will ensure the features created are ready to be put to production and at the end of the sprint it will be reviewed, and the developer will make the necessary changes to the next sprint. There may be a carryover of features if the developer cannot complete all development in the sprint.

5.4 Development environment

The integrated development environment (IDE) that the developer will use is Visual Studio Code, this IDE is created by Microsoft and allows developers to create applications by writing code in the code editor. Visual Studio Code has an array of features that can be utilized in projects like a built-in terminal which will allow the developer to run the project and develop locally. The code editor also implements plugins which can be used to simplify development an example of a plugin used for this project is GitHub Co-pilot which gives artificially intelligent suggestions based off the code that is written. This is an immensely popular IDE that the developer is familiar with which will make the development easier.

GitHub is a popular platform used for managing code, it uses Git software which provides version control for code. The developer will use this platform to store the code for the project in a repository and track the progress with commit messages. The GitHub repository will be connected to Azure so the resources will be automatically redeployed if there are any changes.

5.5 Sprint 1

5.5.1 Goal

The first sprint in this project was to complete the research and requirements sections, the topic was new to the developer so completing research in the area allowed for a greater understanding from the beginning of the project. The requirements section allowed the developer to gather what was needed to create this application, there was also research into similar applications now the developer can select the features that work and do not work to yield the best application.

5.5.2 Item 1

The research chapter which was a literature review of search algorithms was the beginning of this sprint, in this review the developer looked at the several types of search algorithms and where they are used. There was also a considerable amount of research into search frameworks, this application will use the Algolia framework, but the developer had to compare multiple frameworks to find the right one to fit the project requirements. Algolia was picked because it is opensource which leads to a lot of documentation from other users as the service is free, there is also a powerful search engine that will allow users to find content easier.

5.5.3 Item 2

Completing the requirements section was the next step in this sprint, in this section there was a deep dive into what is needed for the project. First the developer looked at other student services applications to see how other applications utilise components to improve their application. Once the developer established an understanding of how other applications worked, the next step was to create personas to depict an example of how distinct types of users will use the application. Now there is a clear depiction of the requirements they can be drafted into a table on level of importance and split into functional and non-functional requirements.

5.6 Sprint 2

5.6.1 Goal

The goal for this sprint was to research and implement technologies that would be used in the project. The developer made starter applications with three different CMSs to find which would be the best for the project. The developer also implemented a search framework into a simple React application to understand how it works.

5.6.2 Item 1

The first task was finding a suitable CMS, the developer came across an article comparing three headless CMSs (Strapi, Directus, Payload). The article concluded that Directus was the best CMS, but the developer saw Strapi as a competent competitor, this led to the developer creating a starter application for each CMS. Each CMS has its documentation on how to create the application and once they were created the developer would pick the one, they liked most.

First was Directus, Directus has a cloud option for creating projects, once the cloud account is created users can create new projects. The developer took this option appose to the docker installation option as it would be less time consuming. Once the project was created the developer could now create collections, collections are like tables in a database each collection will store information that can be accessed by a user.

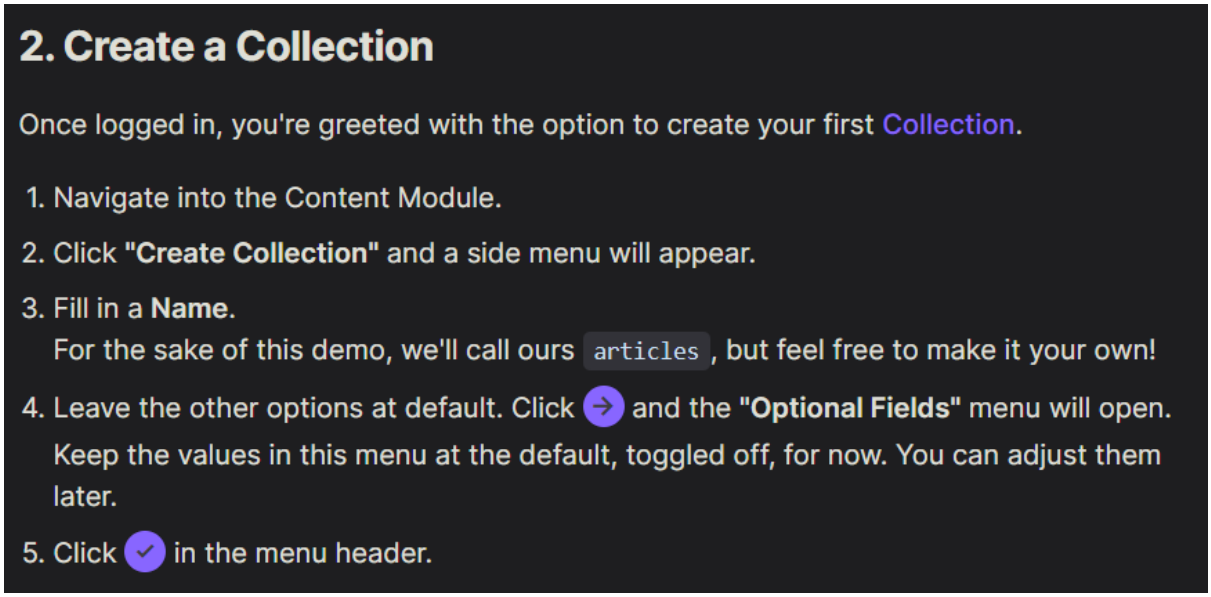


Figure 5.2 Directus documentation for creating collections

Each collection will have multiple fields for example a restaurant collection can have name, location, opening times and capacity fields. These fields can be set in the CMS the developer can customise parameters like input type for example to restrict the name to be a string.

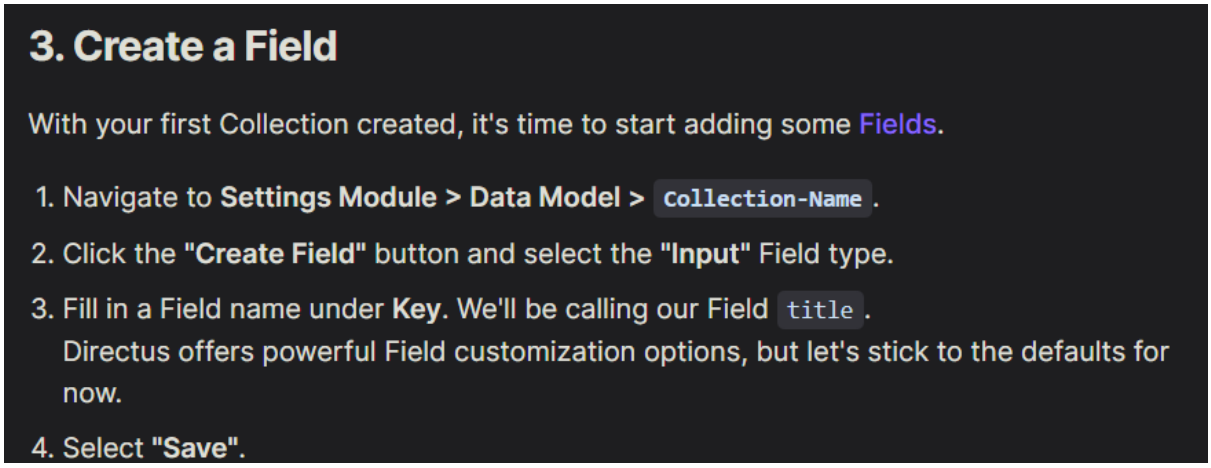


Figure 5.3 Directus documentation for creating fields

Once the collection and fields have been created the developer can now start adding content through the CMS by creating an item once the fields have been filled the item is then saved.

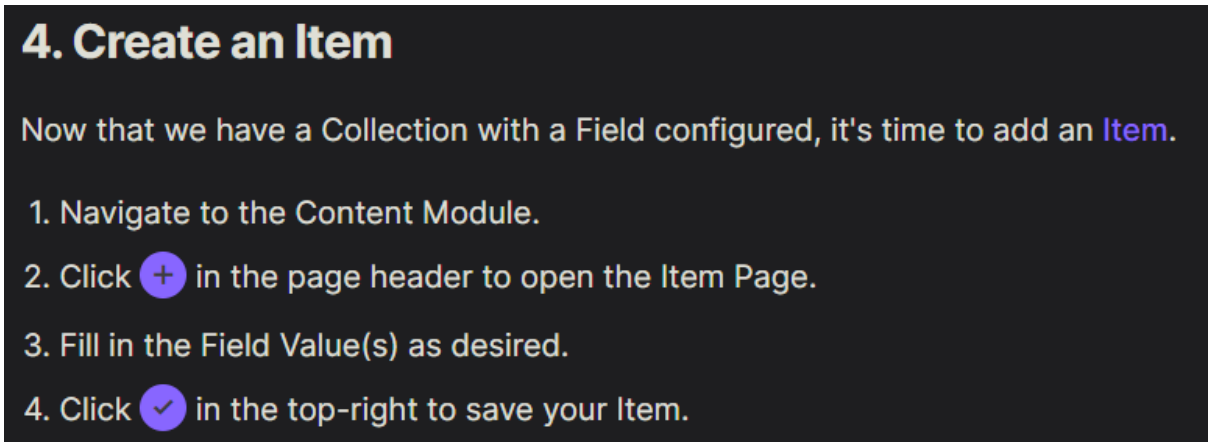


Figure 5.4 Directus documentation for creating item

To access the information the collection needs to be made public so it is accessible through the API, if the collection is not public, it can only be viewed in the database.

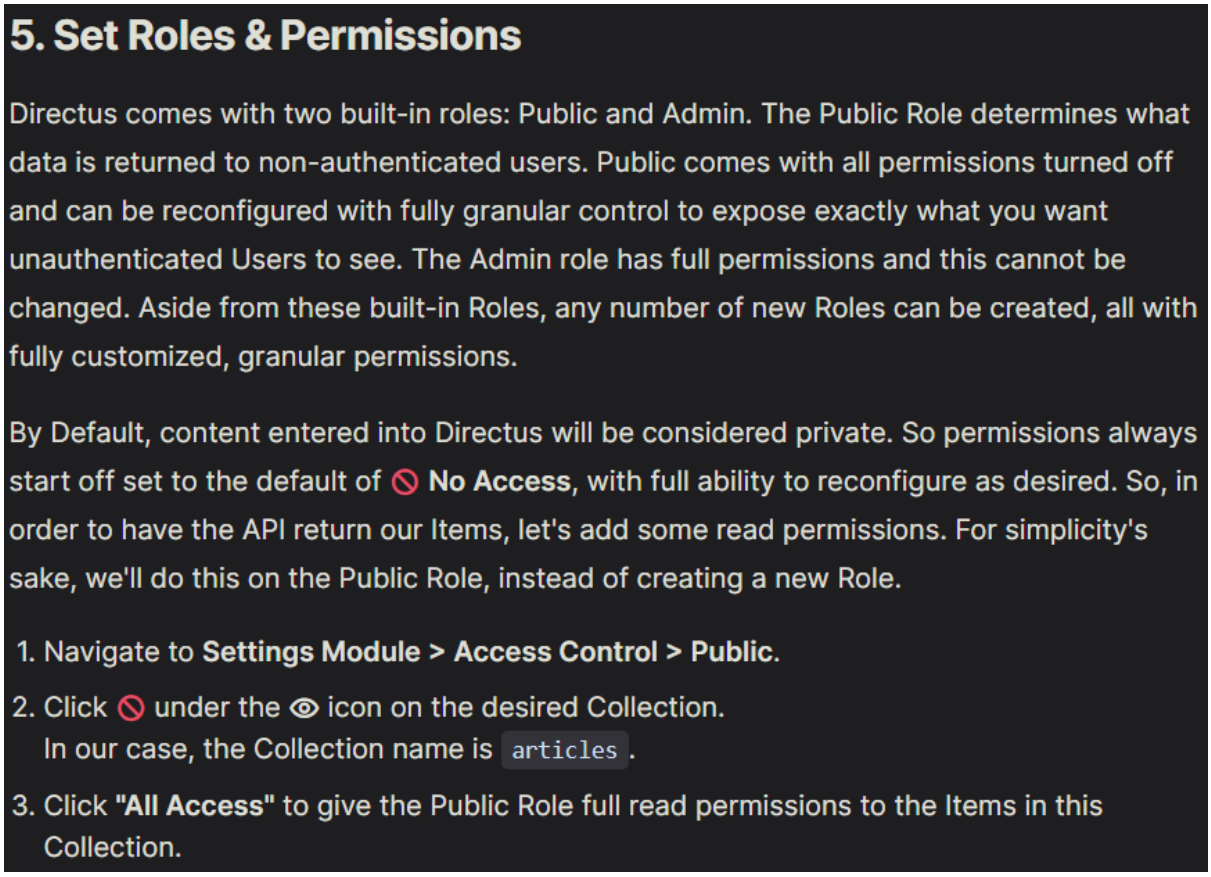


Figure 5.5 Directus documentation for setting roles and permissions

The developer used insomnia to test out the API to see the data that would be returned from the CMS.

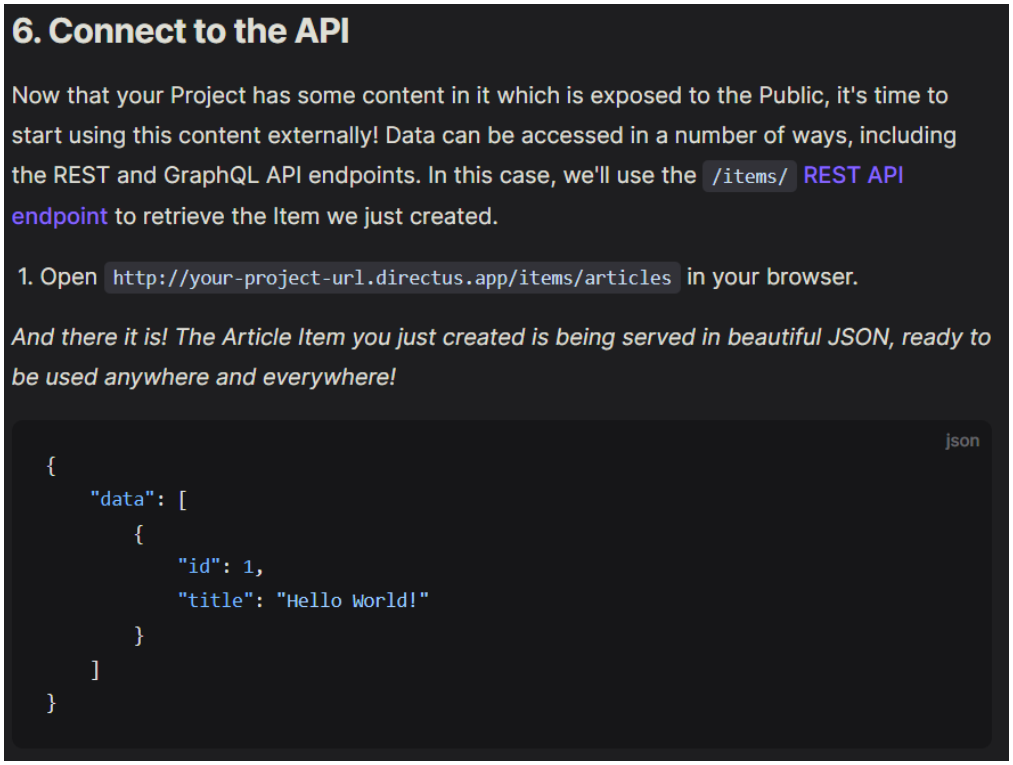


Figure 5.6 Directus documentation for connecting to the API

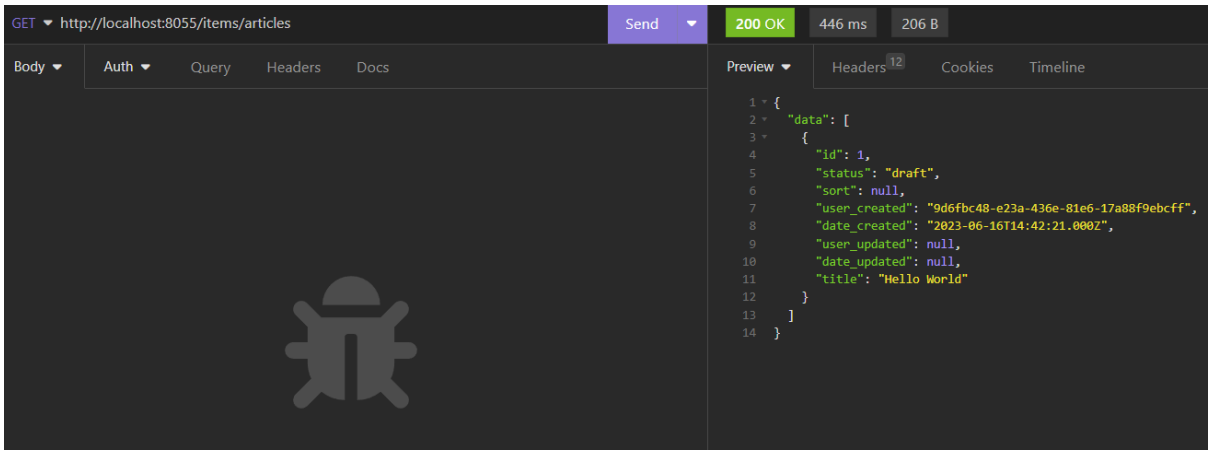


Figure 5.7 Testing the API in Insomnia

Now the first application is complete the developer made similar applications with both Strapi and Payload, the setup was the same as Directus there were no major differences, Strapi and Directus were the main contenders although Payload was good the documentation was not as strong. The developer did notice a useful plugin feature in the Strapi dashboard. The plugin feature is useful because the developer intends to use a search framework in the application that interacts with the CMS, this led to the developer picking Strapi as the CMS for this project. Below is a table of the main findings.

Strapi vs Directus

Pros	Cons
Easy installation and setup	Interface is a bit basic
User friendly interface	Uses GraphQl but is the same without
Compatible with MongoDB	
Works well with React	
Powerful plugin feature	

Pros	Cons
User friendly interface	Interface is very basic
Works well with React	Not compatible with MongoDB
Compatible with SQL databases	
Uses GraphQl for more efficient queries	

Figure 5.8 Strapi and Directus comparison

This exercise was especially useful to the developer as it gave them a familiarity with the CMS that will be used in the project. The next step was to configure sample collections in Strapi so it could mimic what the real application would look like. The developer followed the documentation and created these collections quickly as they were not complex because they were like the one created in the sample application.

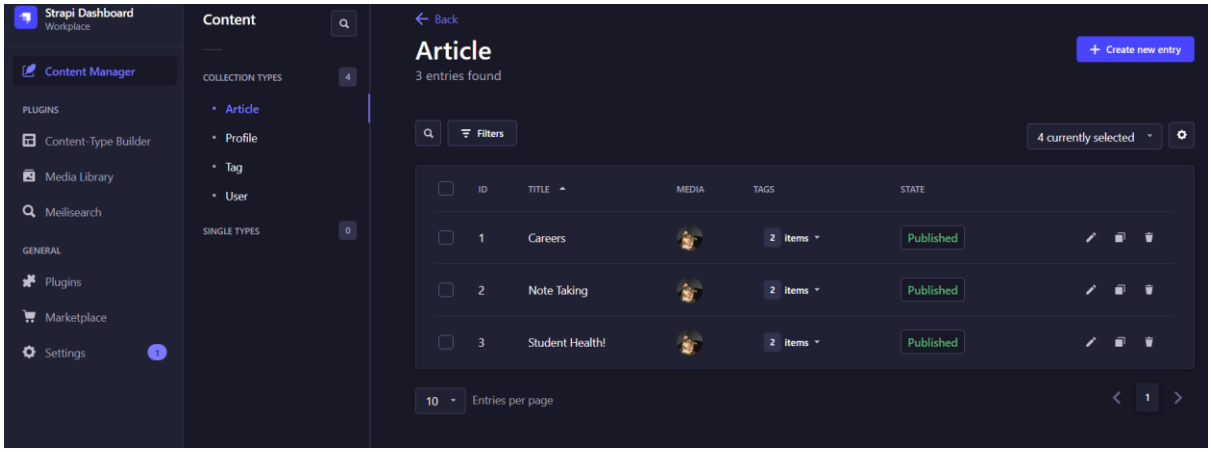


Figure 5.9 Strapi dashboard

The collections are set in such a way as to let the users upload articles that will be seen by students accessing the application.

5.6.3 Item 2

The next task was to implement a search framework that would let users search for content uploaded through the CMS. The developer researched multiple frameworks and found an open-source framework called Algolia. The developer followed articles and YouTube

tutorials to create a simple React application that implements the framework. The developer used the create react app command to create the template for the application. Once the application was ready the developer created a simple page where the user can search.

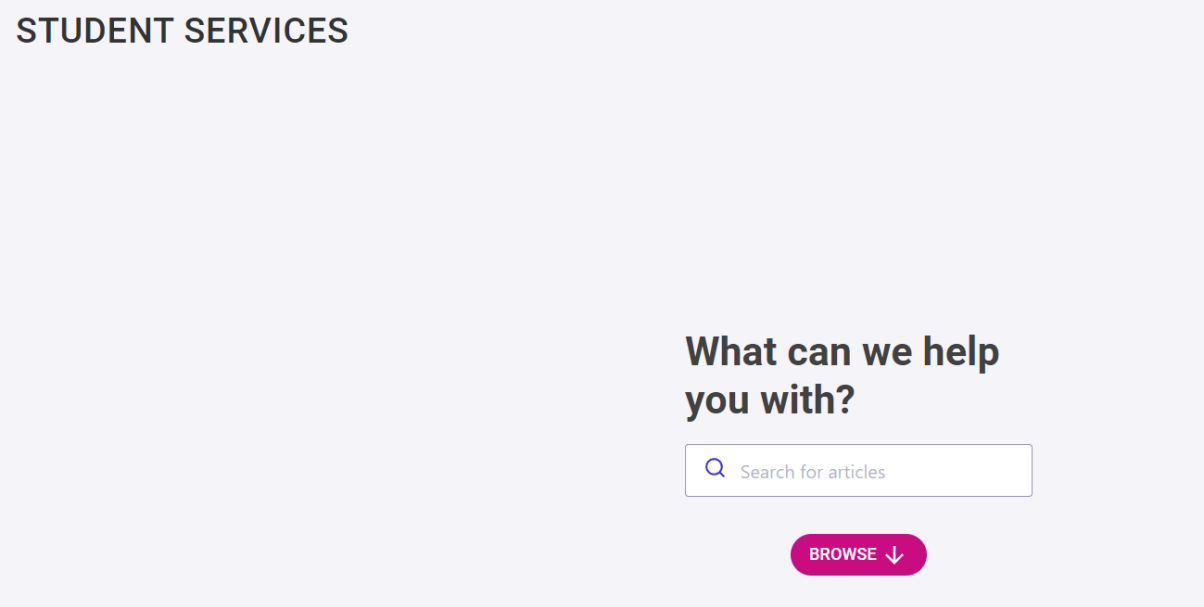


Figure 5.10 Simple React application to integrate the component

The image above shows the application that would be used to implement the framework. To use the framework there are multiple packages required. Once the packages have been installed the developer can now use the component provided by Algolia in the application.

```
"dependencies": {
  "@algolia/autocomplete-js": "^1.11.1",
  "@algolia/autocomplete-plugin-query-suggestions": "^1.11.1",
  "@algolia/autocomplete-plugin-recent-searches": "^1.11.1",
  "@algolia/autocomplete-theme-classic": "^1.11.1",
  "@babel/plugin-transform-numeric-separator": "^7.23.4",
  "@jridgewell/sourcemap-codec": "^1.4.15",
  "@rollup/plugin-terser": "^0.4.4",
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "algoliasearch": "^4.20.0",
```

Figure 5.11 Dependencies for the component

Algolia provides a component with multiple customizable properties that allow users to search, for this project the developer wants to use the recent searches and autocomplete features the component accepts a plugin property where plugins to be used are specified.

```
<Autocomplete
  openOnFocus={true}
  detachedMediaQuery="none"
  placeholder="Search for articles"
  plugins={[recentSearchesPlugin, querySuggestionsPlugin]}
  getSources={({ query }) => [
    {
      sourceId: "articles",
      getItemUrl({ item }) {
        return `/view`;
      },
      getItems() {
        return getAlgoliaResults({
          searchClient,
          queries: [
            {
              indexName: "development_api::article.article",
              query,
            },
          ],
        });
      },
      templates: {
        item({ item, components }) {
          return <SearchItem hit={item} components={components} />;
        },
      },
    },
  ]}
/>
```

Figure 5.12 Autocomplete component

As this component is being tested and implemented the developer set up the Algolia dashboard where the information to be searched will be held. Now when a user uses the search component in the application the information in the Algolia index will be retrieved and showed to the user.

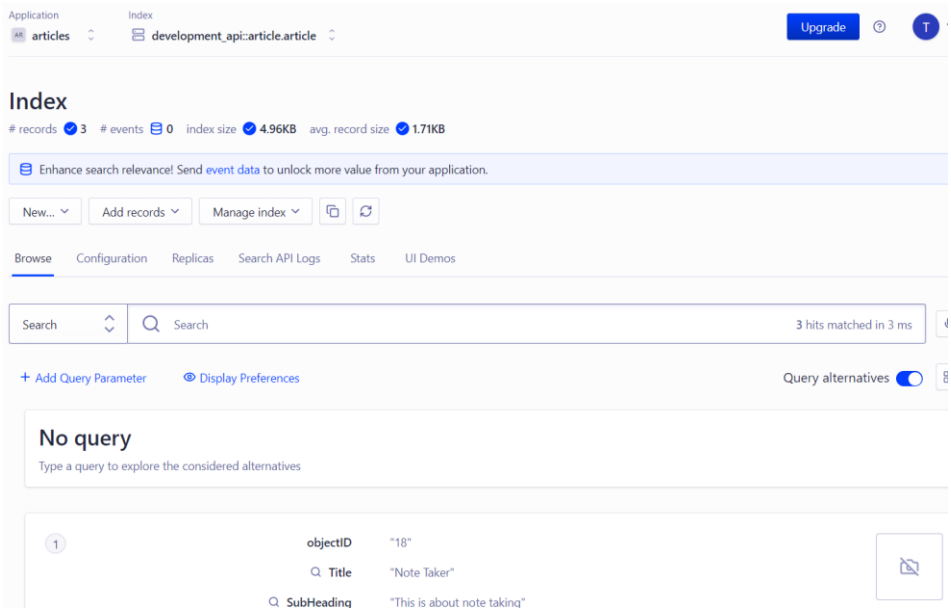


Figure 5.13 Algolia dashboard

Now the basic structure of the project is set up and the developer is confident the technologies used are suitable for the project.

5.7 Sprint 3

5.7.1 Goal

The focus in this sprint was to host all relevant applications and to set up a way for the content uploaded in Strapi to be automatically sent to Algolia so it can be searched.

5.7.2 Item 1

The developer researched how to get the data in Strapi to be sent to Algolia and found that the best way to do this would be through plugins. As mentioned above Strapi has a powerful plugin feature in the application and the developer can use this to overcome this challenge. The developer manually created a plugin through code that is triggered when content is uploaded, now the data is saved in the database and in the Algolia index.

```
module.exports = ({ env }) => ({
  // ...
  search: {
    enabled: true,
    config: {
      provider: "algolia",
      providerOptions: {
        apiKey: process.env.ALGOLIA_PROVIDER_ADMIN_API_KEY,
        applicationId: process.env.ALGOLIA_PROVIDER_APPLICATION_ID,
      },
      contentTypes: [{ name: "api::article.article" }],
    },
  },
});
```

Figure 5.14 Algolia code to trigger the plugin

5.7.3 Item 2

The developer chose Azure to host the application there are two parts to the hosting both the frontend and the CMS. The developer set up an account and logged in to the Azure directory then navigated to the create a resource tab. Using the search, the developer found the static web app service, this service allows developers to create the resource for where the frontend will be hosted. There are multiple fields required for the resource the developer needed to provide a GitHub account and repository that point to where the code for the frontend to be held. Once the resource is created it took about an hour for Azure to configure the

deployment settings and then the application was fully accessible. Now when the developer makes a change locally and pushes it to GitHub the application is automatically redeployed.

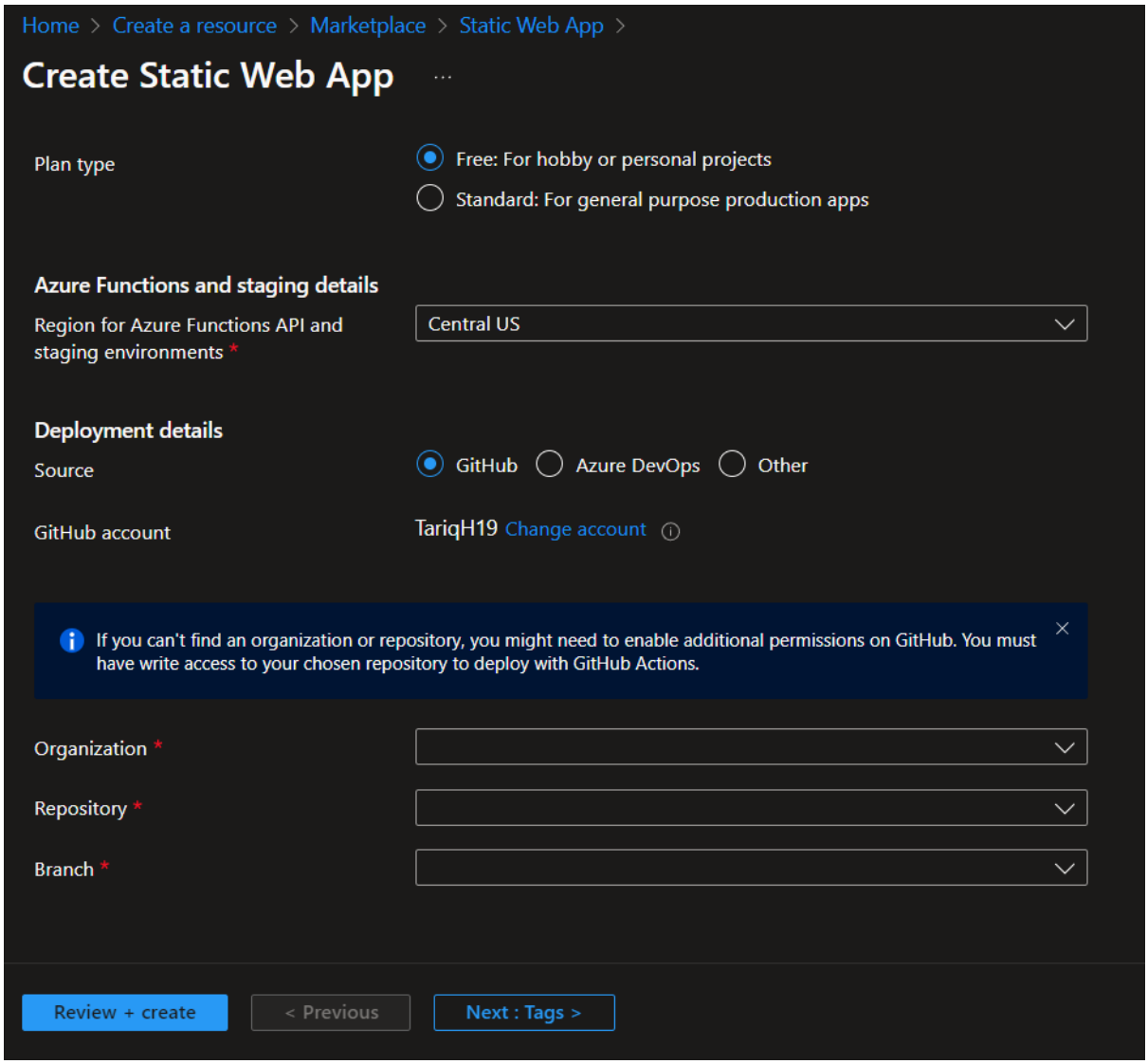


Figure 5.15 Azure form to create a static web app

The only problem that arose was the routing, the developer needs to point the routes back to the index.html file as it is a single page application there was a simple couple lines of code that fixed this problem as there is an abundance of documentation in relation to this situation.

```
<?xml version="1.0"?>
<configuration>
<system.webServer>
<rewrite>
<rules>
<rule name="React Routes" stopProcessing="true">
<match url=".*" />
<conditions logicalGrouping="MatchAll">
<add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
<add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
<add input="{REQUEST_URI}" pattern="^/(api)" negate="true" />
</conditions>
<action type="Rewrite" url="/" />
</rule>
</rules>
</rewrite>
</system.webServer>
</configuration>
```

Figure 5.16 Code to route the application to the root file

Now the frontend is hosted the developer switched focus to the CMS. In Azure there is a resource that automatically sets up your CMS configurations, similar to the frontend Azure requires a GitHub repository and some other data like the database type in this case it was SQLite, once the data is provided Azure creates the resource for the developer, there is a database and a frontend for the Strapi dashboard this process took approximately 2 hours to complete and once completed the developer could now access the CMS through a hosted link provided by Azure.

Now the structure of the project is completely set up the developer can start coding the frontend by creating components that will be used by an array of users.

5.8 Sprint 4

5.8.1 Goal

The goal of this sprint was to develop pages that will be used by users and necessary for testing other features like search.

5.8.2 Item 1

The developer created multiple files which would become pages that the user will see in the application. The first page was the supports page, this page consists of all the services

available in the application, if a user does not want to use the search functionality in the app they will ideally navigate to this page and browse through the services to find what they are looking for. The first step was routing the developer created a new route in the app file so now it can be accessed. Now in the services file the services are fetched from the backend, an Axios call is made to the endpoint and the services are retrieved.

```
useEffect(() => {
  axios
    .get("https://strapiv6336web.azurewebsites.net/api/services?populate=*")
    .then(({ data }) => {
      setSupports(data.data);
      setLoading(false);
    })
    .catch((error) => {
      console.log(error);
      setLoading(false);
    });
}, []);

console.log(supports);
```

Figure 5.17 Axios call to request the services from Strapi

Now to display the services the developer will map through the supports array and display a card for each service in the array, the information displayed will just be the title and a summary of what the service has to offer. This component is shown when the page has loaded, the first visual on the page is a loading spinner which lets the user know the content is being fetched and then the cards are shown. There are multiple classes added to the div that wraps around the card these classes make the page responsive so when the screen is resized the overall structure of the page is still visually pleasing for the user. If the card is clicked it will navigate to the support view page which will show all the information related to that service.

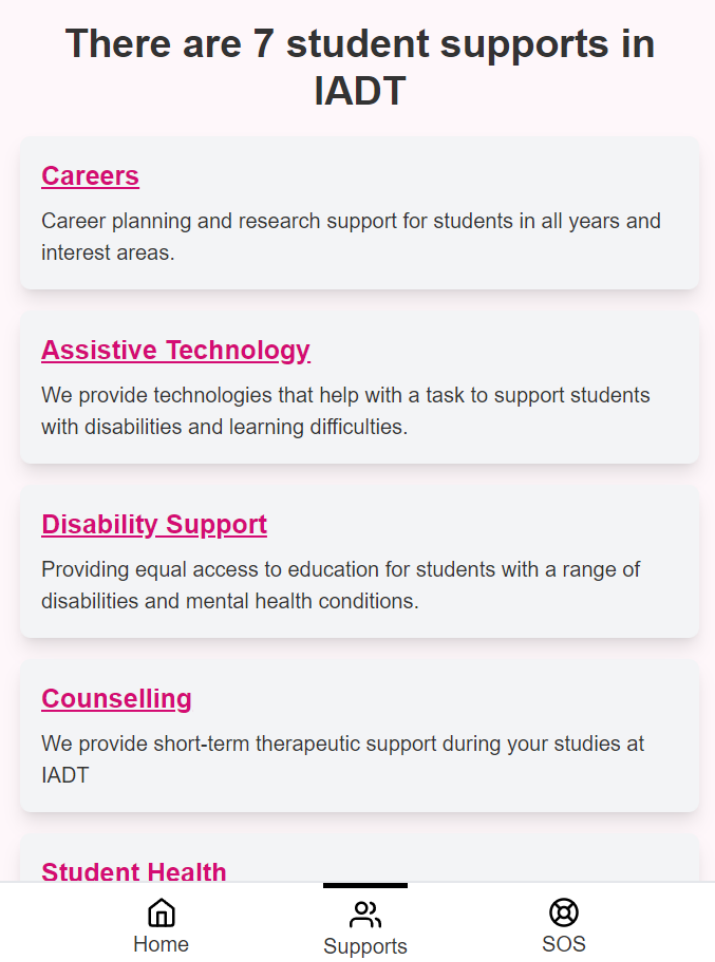


Figure 5.18 Supports page to show all the supports available. Mobile view

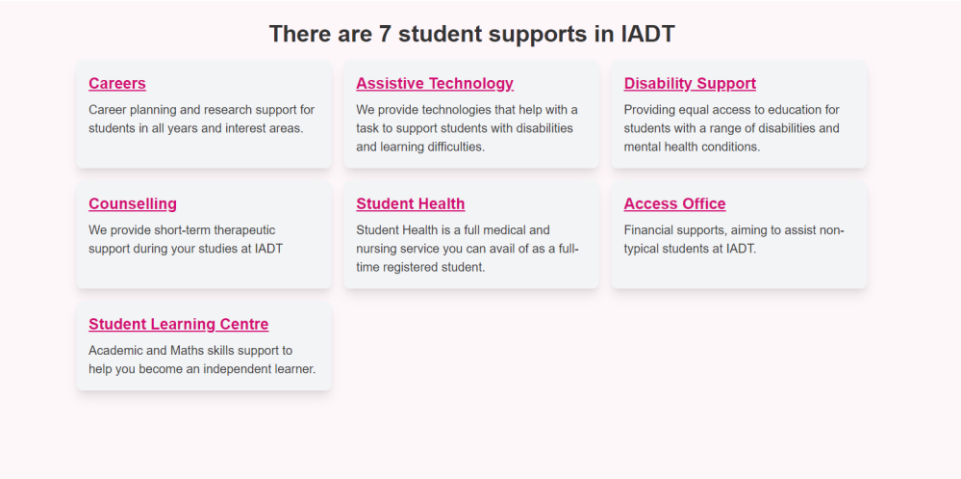


Figure 5.19 Supports page to show all the supports available. Desktop view

Now the services page has been created and is responsive on all screens the developer will now create the support view page.

Figure 5.20 Code for the supports page

To create the support, view the developer repeated the same process completed above for the services page, one slight difference is that the route path will contain a slug the slug will be coming from the backend, and it will essentially be the name of the service, now when a user views the service the slug will appear in the URL. Similarly, to the services page the page has multiple classes to make the page responsive. The support view page is more complex as there is a side menu on the left of the screen so the user can navigate to different sections of the service content and beside this component is the main content but when the screen is in a mobile view the menu on the left disappears and there is just a main body component with the data from the menu now shown with the service information. To achieve this there is a call made to the backend but instead of retrieving all the services the data retrieved is just the service that matches the slug in the request.

Figure 5.21 Axios call to get a specific service

Now the data collected has the service the developer focuses on the layout of the page, there is a parent div which holds all the content on the page this allows for consistent spacing, inside this div is two more children div components one for the side menu on the left of the page and the other will be dedicated to the main content on the page.

The data in the menu will contain the heading of each section in the main body so when a user clicks on the heading the page will jump to the content. There will also be information like opening times and location for the service. Since this content is less important it takes up less space the developer makes the div take up just 25 percent of the screen.

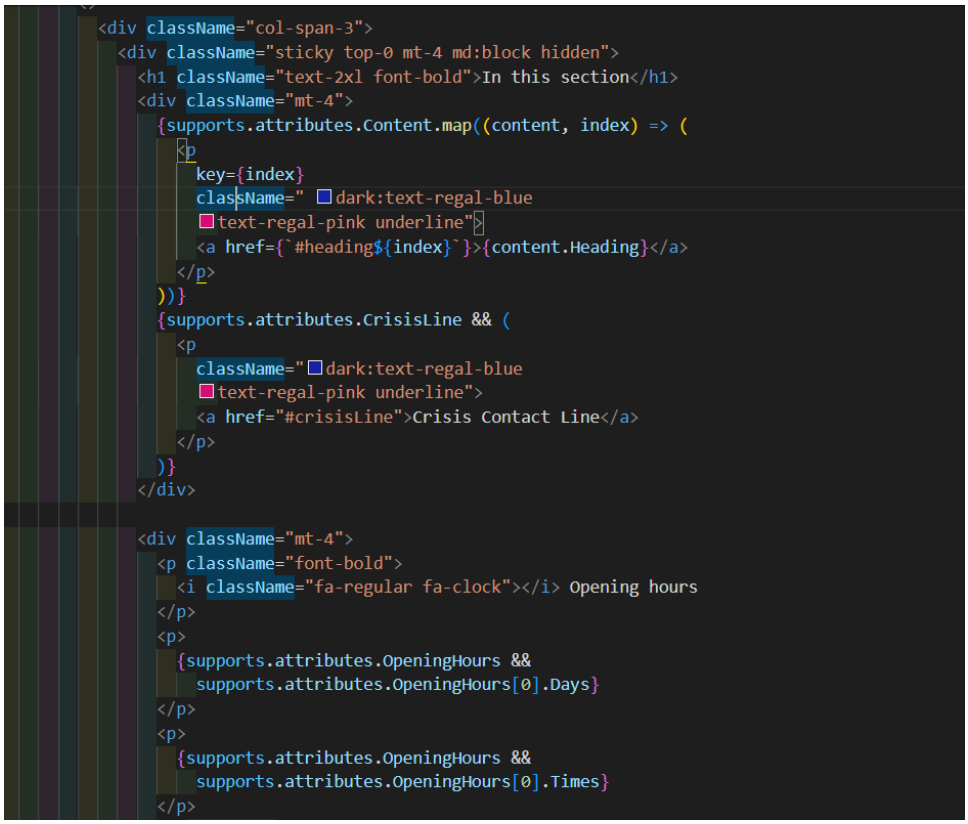


Figure 5.22 Code to create a side menu

The menu has been created now when the heading is clicked it will jump to the content that matches the id, and if the data is present in the array it is displayed for example if there are multiple opening times for the service it will be shown if there is just one opening time that's all that is shown. The crisis line heading is also only shown if it is in the array this can be set in Strapi which will be explained later.

Now the main body component the remaining 75 percent of the page will be used, each service has a YouTube video that will be displayed through an iframe, under this video there will be data about the service, things like what the specialise in and what they can help with.

To display the data there is a map function that goes through the content array and displays each heading and bullet point.

```
<div className="col-span-9">
  <div>
    <Link
      to="/supports"
      className="text-regal-pink dark:text-regal-blue">
      <i className="fa-solid fa-chevron-left fa-xs"></i> All Supports
    </Link>{" "}
  </div>

  <h2 className="text-2xl font-bold my-2">
    {supports.attributes.Title && supports.attributes.Title}
  </h2>
  {supports.attributes.VideoLink ? (
    <iframe
      width="100%"
      height="453"
      src={supports.attributes.VideoLink}
      title="YouTube video player"
      allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"
      allowFullScreen></iframe>
  ) : (
    supports.attributes.Picture && (
      <img
        src={`https://strapiv6336web.azurewebsites.net/${supports.attributes.Picture.data.attributes.url}`}
        alt={supports.attributes.Title}
        className="w-full h-48 object-cover mb-4"
      />
    )
  )}

  <h1 className="mt-4">
    {supports.attributes.VideoCaption &&
      supports.attributes.VideoCaption}
  </h1>
```

Figure 5.23 Code for the main content section

```
{supports.attributes.Content.slice(1).map((content, index) => (
  <div className="my-6" id={`heading${index + 1}`} key={index + 1}>
    <strong className="text-l font-bold my-2">
      {content.Heading}
    </strong>
    <div>
      {content.BulletPoint.split("\n").map(
        (bullet, bulletIndex) => (
          <ul className="ps-4" key={bulletIndex}>
            <li className="list-disc">
              <ReactMarkdown className="list-disc">
                {bullet}
              </ReactMarkdown>
            </li>
          </ul>
        )
      )}
    </div>
  </div>
))}
```

Figure 5.24 Code to map through the content related to each service

5.8.3 Item 2

In conjunction with these frontend developments, the developer continuously made changes to the backend. The first being the structure of the service collection in the backend, the developer added fields to allow user to share data that will be shared on the front end of the application, the most important field is the content component this is a dynamic component

meaning it can be repeated this will allow the user to add multiple sections of content. This creates a quick and user-friendly way of adding content.

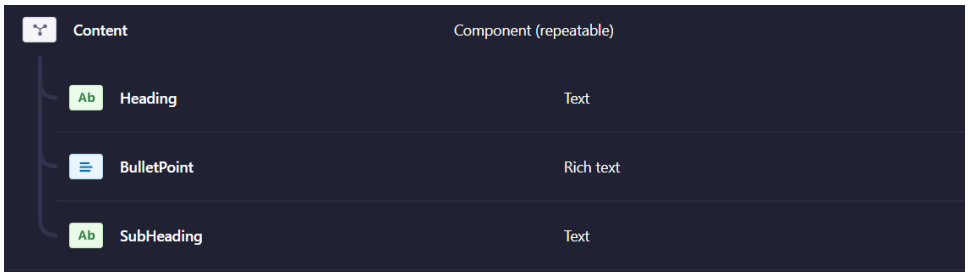


Figure 5.25 Content component in Strapi

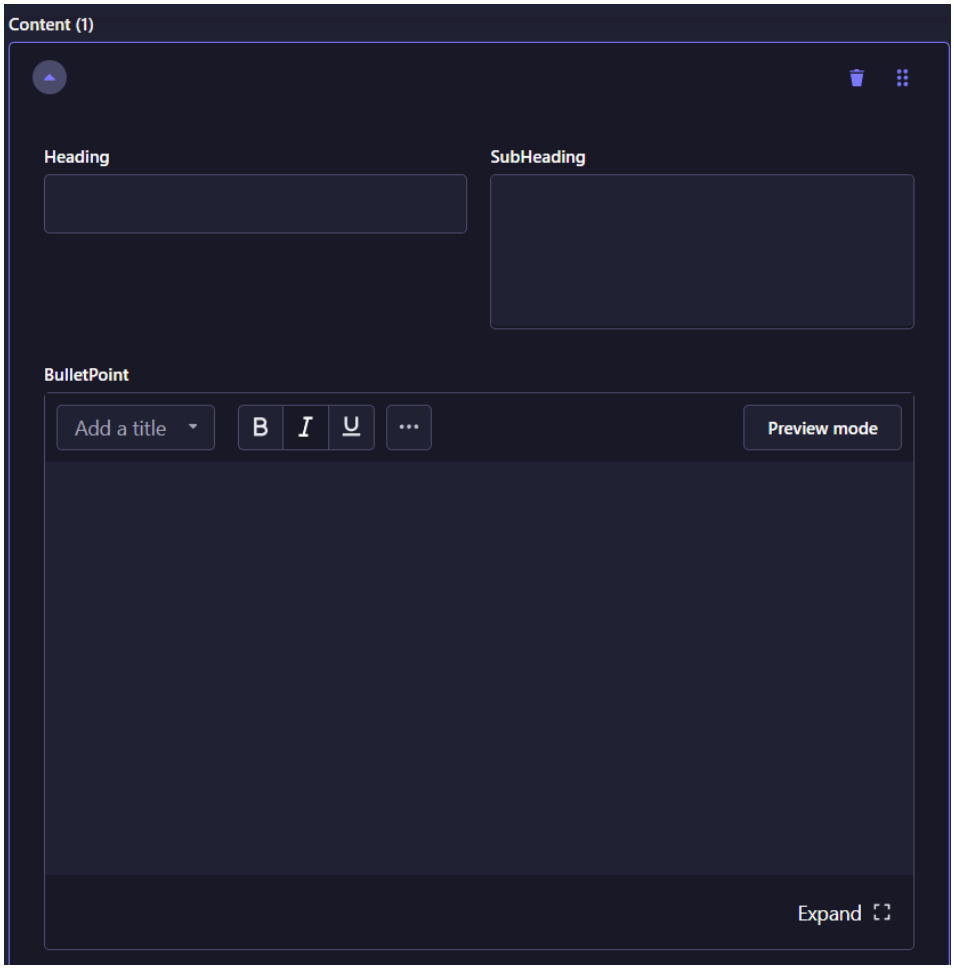


Figure 5.26 Form to add content to Strapi

Next was the crisis line field as mentioned above the crisis line component is only shown on the front end if it is enabled in the backend the developer created a Boolean field so now if the service does not need the component to be displayed, they set the value to false.

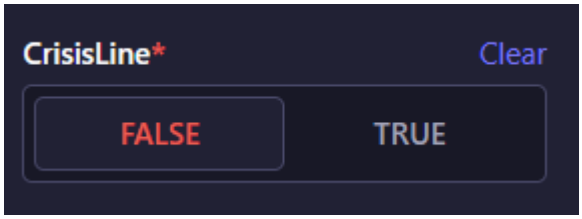


Figure 5.27 Boolean value for Crisis line

Now some of the front end has been set up and the backend has been updated the focus is now on the search functionality in the application.

5.9 Sprint 5

5.9.1 Item 1

The main goal of this sprint is to get the search functionality in the app to work as intended the main issue is that when a user clicks on one of the search results they are not navigated to the right page. To fix this problem the developer set up a new field in the back end, in Strapi there is a slug field that dynamically creates a slug based on the title of the service.

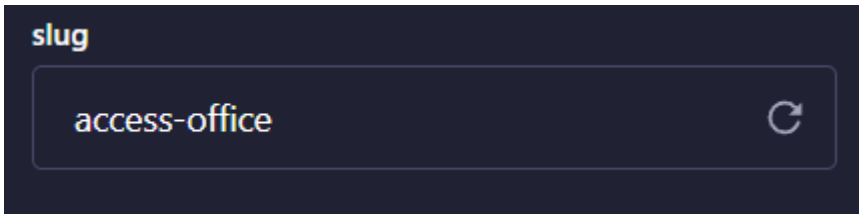


Figure 5.28 Slug value used for search dynamically matching the title

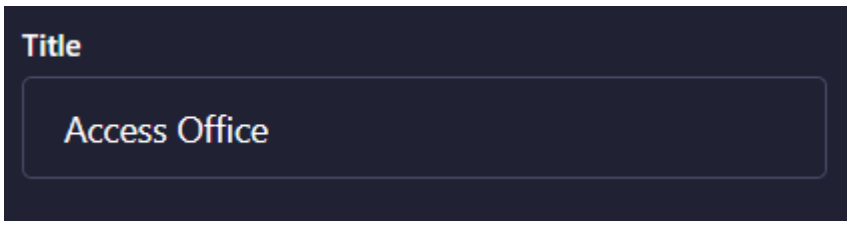


Figure 5.29 Title field used to name the service and create the slug

Now on the front end when a user clicks on one of the search items they should be navigating to the results page, the developer created a function that takes the slug of the item that was clicked and adds it to the URL so now the results page will be sent the slug of the service.

```
const SearchItem = ({ hit, components }) => {
  const handleClick = () => {
    const newUrl = "/";
    window.history.replaceState(null, "null", newUrl);
    // window.location.href = `${slugType}/${type}/${hit.slug}`;
    window.location.href = `/results/${hit.slug}`;
  };

  return (
    <div className="aa-ItemContent" style={{ cursor: "pointer" }}>
      <div onClick={handleClick} className="flex items-center justify-between">
        <div className="pb-6">
          <components.Highlight hit={hit} attribute="Title" />
        </div>
      </div>
    </div>
  );
};
```

Figure 5.30 Component to show results in the search component

Now that it has been set up behind the scenes, the developer creates the page to show the results for what has been clicked. Again, a route is created, and a page file is made so now the results of the search can be shown. The developer is still working on how the data will be filtered to show the results so for now the page just shows hello world, but now when a user clicks the search item the user is shown the results page, and the slug is shown in the URL.

5.9.2 Item 2

The Algolia autocomplete component comes with multiple functionality and design features that do not fit the application first being the appearance of the component there are some minor changes that need to be made the developer changed the components colours to match the wireframes in the design section. To make these visual changes the developer created a useEffect that would find the elements in the component by class name and then add classes that were created in the CSS file.

```
useEffect(() => {
  const icon = containerRef.current.querySelector(
    ".aa-DetachedSearchButtonIcon"
  );

  const submitIcon = containerRef.current.querySelector(".aa-Label svg");

  if (icon) {
    icon.classList.add("search-icon");
  }

  if (submitIcon) {
    submitIcon.classList.add("search-icon");
  }

  const focus = containerRef.current.querySelector(".aa-Form");
  if (focus) {
    focus.classList.add("search-focus");
  }
}, []);
```

Figure 5.31 Adding custom CSS classes to the search component

The next change to the component was to add keyboard functionality, the component closes when enter is pressed on the keyboard, but the developer needs the search to be triggered meaning the words typed into the search box and show the results for the search term. The developer created another function that would check for the enter key event and complete the search functionality like the clicked search item. Now when a user adds data into the search box and presses enter, they will be shown the results page.

```
const handleKeyDown = (event) => {
  if (event.key === "ENTER" || event.key === "Enter") {
    // Get the entered query from the Autocomplete input field
    const newUrl = "/";
    window.history.replaceState(null, "null", newUrl);
    const query = event.target.value.trim();

    // Redirect the user to the search results page with the entered query
    window.location.href = `/results/${encodeURIComponent(query)}`;
  }
};

return (
  <>
    <div className="mx-4">
      {" "}
      <Autocomplete
        onKeyDown={handleKeyDown}
```

Figure 5.32 Function to navigate users to the results page

Now the search functionality works in the component the developer plans to meet with their supervisors to discuss potential enhancements.

5.10 Sprint 6

5.10.1 Goal

The goal of this sprint was to get feedback from the supervisors and see what changes can be considered while also developing the front end.

5.10.2 Item 1

The developer met with the supervisors and showed them the application, how it works behind the scenes and what other changes will be made in the future. The major takes away from the discussion was that the search should be the focus meaning features like filtering and suggestions should be considered and lastly a field in the back end for the users to add search terms so on the front-end content is found that is not just related to the title of the service. The application also had an abundance of sample data and looked unfinished in certain places making it look underwhelming, to take the project look more professional these issues would have to be addressed by the developer.

First was to look at the search and the developer found that due to a lack of data projected to be in the application that filtering would be redundant and that the search should be used to filter content as well as finding relevant results, the next researched item was the suggestions piece, while being a good idea the search is intended to be used by users that know roughly what they want to find and if they are unsure they will browse through the services. Although the suggestions made by the supervisors will not be in the definitive version it was useful feedback for the developer, and it gave them the opportunity to develop the results page.

The developer cleaned up the application by adding content to the backend, the content was taken from the IADT website and the content that was already in the services pages was condensed to fit the application, once the content was added the application started to take shape and the developer focused on colours and the small details like spacing between elements on throughout the application.

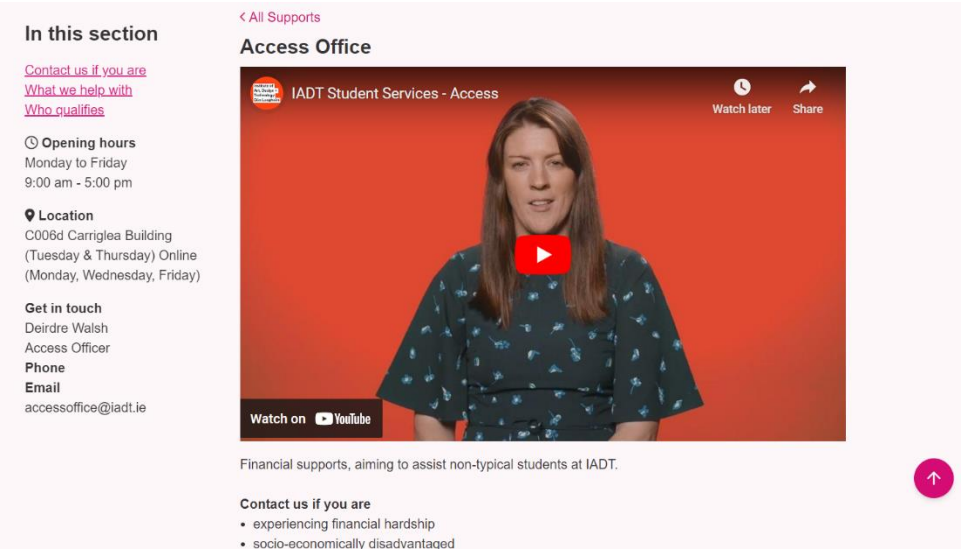


Figure 5.33 Support view page

5.10.3 Item 2

Now the attention is back on the search functionality, the developer creates a new text field in Strapi called search terms this will be used to have a more conversational approach to searching, the intention being if a user searches, “I need help” they will see the relevant content even though it is not the name of a service. For this to work a slight change needs to be made in Algolia to allow the search terms to be searched by setting the field as a searchable attribute it will now be picked up when a user types it in the search box.

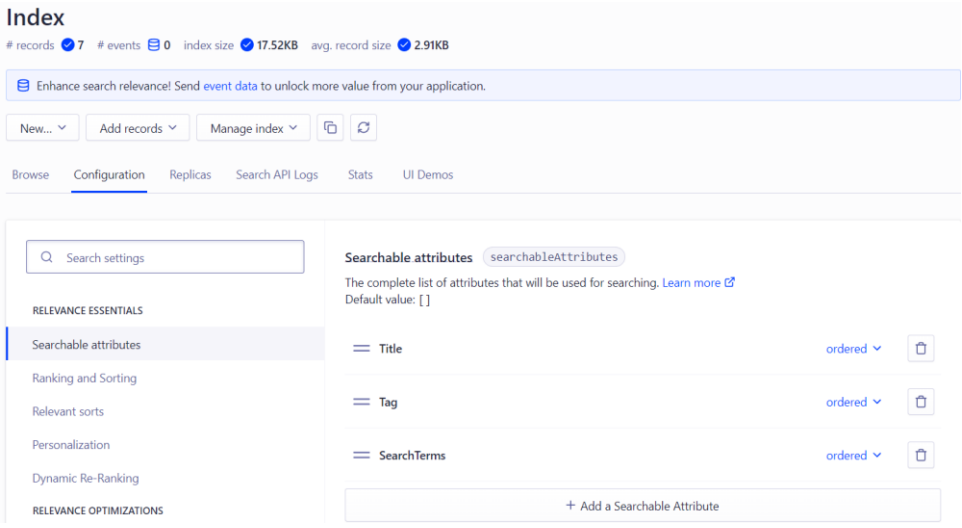


Figure 5.34 Setting the searchable attributes in Algolia

The results page is going to have many functions behind the scenes that will intelligently find content relevant to what has been searched and relevant content that may help the user. Like the other pages the data is retrieved from the back end.

Next the developer focused on the functions needed to get the relevant results. If a user is on the homepage and searches for health for example there is a function that takes that search query and filters the services based on the query, if that query matches the title of a service or the tags related to the service or is in the search terms it will be shown first, now to show related results any services that share the query in either the tags or the search terms will be shown after. This implements filtering functionality it just is not done by the user, lastly if the query is in the main content of the service, it will be highlighted in the card that displays the results otherwise the first three bullet points in the service will be shown.

```
const searchedServices = services.filter((service) => {
  const title = service.attributes.Title
    ? service.attributes.Title.toLowerCase()
    : "";
  const searchTerms = service.attributes.SearchTerms
    ? service.attributes.SearchTerms.toLowerCase()
    : "";
  // return title.includes(searchQuery) || searchTerms.includes(searchQuery) ;
  const content = service.attributes.Content.map(
    (section) => section.BulletPoint
  )
    .join(" ")
    .toLowerCase();
  return (
    title.includes(searchQuery) ||
    searchTerms.includes(searchQuery) ||
    content.includes(searchQuery)
  );
});
```

Figure 5.35 Function to retrieve services based on the Title, search terms and content

```
let filteredServices = [];
if (searchedServices.length > 0) {
  filteredServices = [...searchedServices];
  searchedServices.forEach((searchedService) => {
    const searchedTags = searchedService.attributes.tags.data.map((tag) =>
      tag.attributes.Tag.toLowerCase()
    );
    services.forEach((service) => {
      if (
        service.id !== searchedService.id &&
        service.attributes.tags.data.some((tag) =>
          searchedTags.includes(tag.attributes.Tag.toLowerCase())
        ) &&
        !filteredServices.find((a) => a.id === service.id)
      ) {
        filteredServices.push(service);
      }
    });
  });
}
```

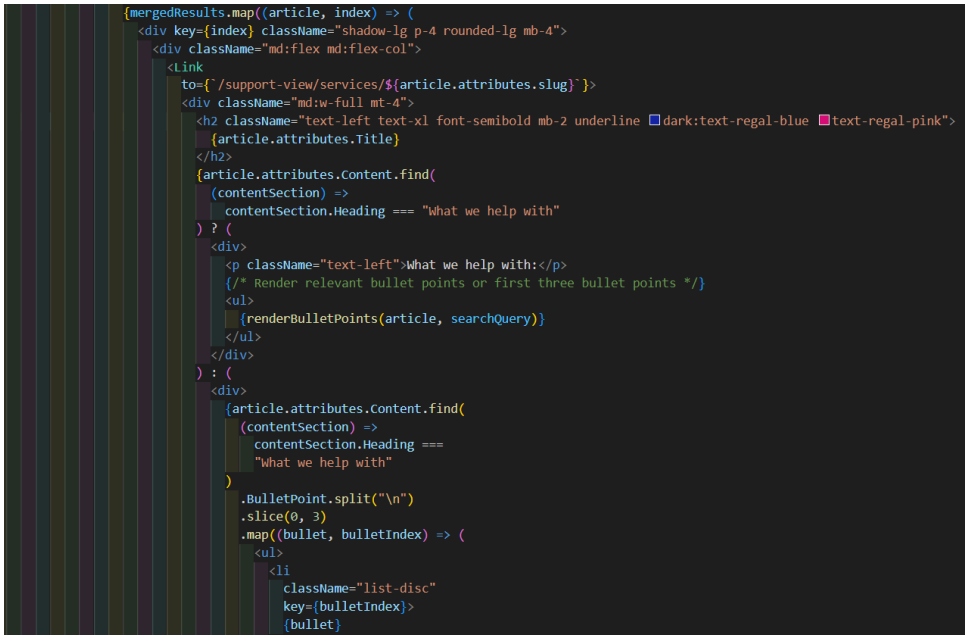
Figure 5.36 Function to get services with the same tags

```
const renderBulletPoints = (service, searchQuery) => {
  const helpWithSection = service.attributes.Content.find(
    (section) => section.Heading === "What we help with"
  );

  if (helpWithSection) {
    const bulletPointsArray = helpWithSection.BulletPoint.split("\n");
    const relevantBulletPoints = bulletPointsArray.filter((bullet) =>
      bullet.toLowerCase().includes(searchQuery)
    );

    if (relevantBulletPoints.length > 0) {
      return relevantBulletPoints.map((bullet, bulletIndex) => (
        <ul key={bulletIndex} className="pl-4">
          {" "}
          <li className="list-disc" key={bulletIndex}>
            {bullet
              .split(new RegExp(`(${searchQuery})`, "gi"))
              .map((part, index) =>
                part.toLowerCase() === searchQuery.toLowerCase() ? (
                  <strong key={index}>{part}</strong>
                ) : (
                  part
                )
              )}
          </li>
        </ul>
      ));
    } else {
      return bulletPointsArray.slice(0, 3).map((bullet, bulletIndex) => (
        <ul key={bulletIndex} className="pl-4">
          <li className="list-disc " key={bulletIndex}>
            {bullet}
          </li>
        </ul>
      ));
    }
  }
};
```

Figure 5.37 Function to show relevant information if the search query is in the service



```
{mergedResults.map((article, index) => (
  <div key={index} className="shadow-lg p-4 rounded-lg mb-4">
    <div className="md:flex md:flex-col">
      <Link
        to={`/support-view/services/${article.attributes.slug}`}>
        <div className="md:w-full mt-4">
          <h2 className="text-left text-xl font-semibold mb-2 underline dark:text-regal-blue text-regal-pink">
            {article.attributes.Title}
          </h2>
          {article.attributes.Content.find(
            (contentSection) =>
              contentSection.Heading === "What we help with"
          ) ? (
            <div>
              <p className="text-left">What we help with:</p>
              {/* Render relevant bullet points or first three bullet points */}
              <ul>
                {renderBulletPoints(article, searchQuery)}
              </ul>
            </div>
          ) : (
            <div>
              {article.attributes.Content.find(
                (contentSection) =>
                  contentSection.Heading ===
                  "What we help with"
              )
                .BulletPoint.split("\n")
                .slice(0, 3)
                .map((bullet, bulletIndex) => (
                  <ul>
                    <li
                      className="list-disc"
                      key={bulletIndex}>
                      {bullet}
```

Figure 5.38 Code to show the results based on what is searched

The developer made major progress in relation to the search and the results in this sprint now the emphasis will be on making the app more accessible for users by making user interface changes, the developer plans to consult with Kacper the designer to discuss changes and final designs.

5.11 Sprint 7

5.11.1 Goal

The goal of this sprint is to workshop with Kacper and come to an agreement of how the application should look and feel. This meeting will define how the application should look and the content that will be in the application.

5.11.2 Item 1

The meeting provided the developer with an array of changes, as there is not sufficient data to have articles in the application that part of the application will be taken out, the functionality remains the same as the developer has already set up the application to accept articles but due to the cease in articles the application will now just be services information. This change means the developer will have to remove the articles from being seen by the user. The last major change was the addition of a theme switcher, the application has an accessibility requirement and as the colours used might not suit some users, they will need an option to change the colours through a theme switcher, there will also be two different navigation bars

one for mobile and another for screens bigger than mobile. Other than that, there were just small corrections like spacing issues and certain icons were the wrong colour and size.

5.11.3 Item 2

To fix the issues that were related to design the developer took to the built-in inspector tool in chrome that allowed them to pick the specific component and observe the styles applied that need to be changed accordingly.

For the different navbars the developer thought it would be best to separate the components to a top navbar, a bottom navbar and a general navbar component where both components will be imported, this is so when the screen is resized the applicable component will be shown.

```
<div
  className={`mt-auto bottom-0 fixed flex border-t-2 w-full justify-evenly items-center text-center bg-white `}
  style={{ zIndex: 100 }}>
  <Link
    className={`cursor-pointer flex flex-col items-center justify-center border-regal-black ${
      location.pathname === "/" ? "border-t-4 border-black" : ""
    }`}
    to="/">
    <img src={Home} alt="home" className="w-6 h-6 mt-2" />
    <span>Home</span>
  </Link>

  <Link
    className={`cursor-pointer flex flex-col items-center justify-center ${
      location.pathname === "/supports" ? "border-black border-t-4  " : ""
    }`}
    to="/supports">
    <img src={Support} alt="support" className="w-6 h-6 mt-2" />
    <span>Supports</span>
  </Link>

  <Link
    className={`cursor-pointer flex flex-col items-center justify-center ${
      location.pathname === "/sos" ? "border-t-4 border-black" : ""
    }`}
    to="/sos">
    <img src={buoy} alt="buoy" className="w-6 h-6 mt-2" />
    <span>SOS</span>
  </Link>
</div>
```

Figure 5.39 Navbar component for mobile

In the top navbar the user will be able to open a modal that will allow them to choose the colour theme of the application.

```
<div className="mb-4 lg:px-32 bg-regal-pink text-white">
  <div className="py-4 mx-4 flex justify-between  ">
    <a href="/" className=" text-2xl font-bold">
      Student Services
    </a>

    <div className="flex space-x-4 ml-4">
      <a href="/" className=" hover:text-gray-300">
        Home
      </a>
      <a href="/supports" className=" hover:text-gray-300">
        Supports
      </a>
      <a href="/sos" className=" hover:text-gray-300">
        Emergency
      </a>
      <p>|</p>
      <div className="flex cursor-pointer" onClick={openModal}>
        <img src={SettingsIcon} alt="settings" className="w-6 h-6 mr-2" />
        <p className="text-white">Settings</p>
      </div>
      {showModal && <SettingsModal onClose={closeModal} />}
    </div>
  </div>
</div>
```

Figure 5.40 Navbar component for desktop

```
<div>
  <div className="hidden md:block">
    <TopNavbar />
  </div>
  <div className="md:hidden">
    <BottomNavbar />
  </div>
</div>
```

Figure 5.41 Component to show the relevant navbar based on screen size

The last feature to be implemented in this sprint is the theme switcher, the developer started by creating a context that would wrap around the app so when the user changes the theme the colours all throughout the app change this change will be tracked through the user clicking a settings button.

```
<div
  className="inline-block align-bottom bg-white rounded-lg text-left overflow-hidden shadow-xl transform transition-all sm:my-8 sm:align-middle
  role="dialog"
  aria-modal="true"
  aria-labelledby="modal-headline">
  <div className="bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb-4">
    <h3
      className="text-xl mb-2 leading-6 font-bold text-gray-900"
      id="modal-headline">
      Settings
    </h3>
    <p className="mb-4 text-black">
      Choose the appearance of IADT supports to best meet your needs
    </p>
    <ThemeSwitcher />
  </div>

  <div className="bg-gray-50 px-4 py-3 sm:px-6 sm:flex sm:flex-row-reverse">
    <button
      onClick={onClose}
      type="button"
      className="  dark:bg-regal-blue bg-regal-pink w-full inline-flex justify-center rounded-md border border-transparent shadow-sm px-4 py-2
      Done
    </button>
  </div>
</div>
```

Figure 5.42 Theme switcher modal

The theme switcher sets the contrast mode to true in the local storage now the developer can check if this variable is true and change the colours in the app.

```
useEffect(() => {
  const root = document.documentElement;
  if (contrast) {
    root.classList.add("contrast-mode");
  } else {
    root.classList.remove("contrast-mode");
  }
  document.documentElement.classList.toggle("contrast", contrast);
  localStorage.setItem("contrast", JSON.stringify(contrast)); // Stringify boolean value
}, [contrast]);

const toggleContrast = () => {
  setContrast((prevMode) => !prevMode);
};

return (
  <div className="flex items-center">
    <p className="mr-auto text-black">High Contrast Mode</p>
    <label className="inline-flex items-center cursor-pointer">
      <input
        type="checkbox"
        checked={contrast}
        onChange={toggleContrast}
        className="sr-only peer"
      />
      <div className="relative w-11 h-6 bg-gray-300 rounded-full peer peer-checked:after:translate-x-full rtl:peer-checked:after:-translate-x-full
    </label>
  </div>
);
```

Figure 5.43 Theme Switcher component

This feature works but as the developer was implementing and testing the colours on the app would change but could not be changed back, the developer decided to fix this feature in the next sprint.

5.12 Sprint 8

5.12.1 Goal

The goal of this sprint was to implement a testing framework and fix any bugs that the developer could not fix.

5.12.2 Item 1

The testing framework that the developer chose was SonarQube, this testing framework checks the quality of static code. The developer will test the front end and back end through

user testing and the static code will be checked by SonarQube. The developer decided to download the framework and run it locally as hosting the framework would be redundant as the developer wants to only test the code when there are substantial changes only and not for small design changes. The developer runs the framework on port 9000 and this is where the dashboard is accessed, to add a project the developer generates a token that will be used in the command in the project directory. The framework runs tests that test the code on principles like security, reliability, and maintainability. Once the code has been checked the developer can see the results in the dashboard, there is an overall score, and each principle is given a grade the developer can now review the issues raised by the framework and can either change them in the dashboard by marking them as false positives or change the code itself.

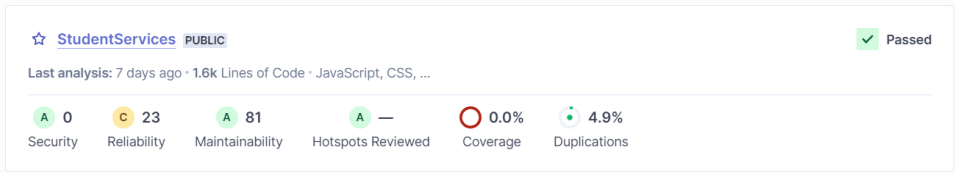


Figure 5.44 SonarQube test results

The results were positive overall the only issue was there was sensitive data related to the Algolia index that were exposed the developer moved the data to an .env file.

The first error the developer fixed was in the autocomplete component when the component is mounted a new root is created but due to the developer using the latest version of React the current code does not match the new syntax.

```
const search = autocomplete({
  container: containerRef.current,
  renderer: { createElement, Fragment, createRoot },
  render({ children }, root) {
    createRoot(root).render(children);
  },
  ...props,
});
```

Figure 5.45 Code that creates errors in the console as it creates a new root

The first approach involves creating a new root element using createRoot for each rendering operation, while the second approach checks if a root element already exists and updates it, if necessary, before rendering the children. The key distinction is that the first approach creates a new root instance for every rendering, potentially leading to inefficiency and errors,

whereas the second approach ensures that only one root is used, thereby promoting consistency, and avoiding potential issues associated with multiple root creation.

```
const search = autocomplete({
  container: containerRef.current,
  renderer: { createElement, Fragment, render: () => {} },
  render({ children }, root) {
    if (!panelRootRef.current || rootRef.current !== root) {
      rootRef.current = root;

      panelRootRef.current?.unmount();
      panelRootRef.current = createRoot(root);
    }

    panelRootRef.current.render(children);
  },
  ...props,
});

return () => {
  search.destroy();
};
}, [props]);
```

Figure 5.46 Updated code that creates one root for the autocomplete component

5.12.3 Item 2

The next error is in the app index again this is because of the latest version of React being used.

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter as Router } from "react-router-dom";
import "./index.css";
import App from "./App";

ReactDOM.render(
  <React.StrictMode>
    <Router>
      <App />
    </Router>
  </React.StrictMode>,
  document.getElementById("root")
);
```

Figure 5.47 Deprecated index file for React

In the first approach shown above, the app is rendered using the conventional ReactDOM.render method, while in the second approach shown below, createRoot from react-dom/client is utilized to create a root instance for rendering the app. The second approach provides more control over the rendering process.

```
import React from "react";
import { createRoot } from "react-dom/client";
import "./index.css";
import App from "./App";
import { BrowserRouter } from "react-router-dom";

const root = createRoot(document.getElementById("root"));

root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

Figure 5.48 Code to fix deprecated React index file

The last error to fix was the high contrast functionality, to fix the error the app component needs to be wrapped in a theme provider this will make the colours within the app change globally as the new theme provider will update the children components.

```
import React, { createContext, useState, useEffect } from "react";

export const HighContrastContext = createContext();

export const HighContrastProvider = ({ children }) => {
  const [highContrast, setHighContrast] = useState(() => {
    const storedHighContrast = localStorage.getItem("highContrast");
    return storedHighContrast ? JSON.parse(storedHighContrast) : false;
  });

  useEffect(() => {
    localStorage.setItem("highContrast", JSON.stringify(highContrast));
  }, [highContrast]);

  const toggleHighContrast = () => {
    const newHighContrastState = !highContrast;
    setHighContrast(newHighContrastState);
    if (!newHighContrastState) {
      localStorage.removeItem("highContrast");
    }
  };

  return (
    <HighContrastContext.Provider value={{ highContrast, toggleHighContrast }}>
      {children}
    </HighContrastContext.Provider>
  );
};
```

Figure 5.49 Theme switcher code that addresses the previous bugs

The final issue with this feature is that the colours would not revert to the original state, to fix this the developer removed to high contrast item from local storage meaning that the colours will update as the variable is in the useEffect.

5.13 Sprint 9

5.13.1 Goal

The objective of this sprint was twofold: first, to conduct user tests to assess the stability and functionality of the application, and second, to integrate an AI chatbot into the frontend to enhance user interaction and help.

5.13.2 Item 1

After consulting with the supervisor, the developer received valuable advice on enhancing the project further. One of the ideas that emerged from this discussion was the integration of a chatbot feature to assist students. Acting upon this suggestion, the developer embarked on the process of implementing a chatbot into the project.

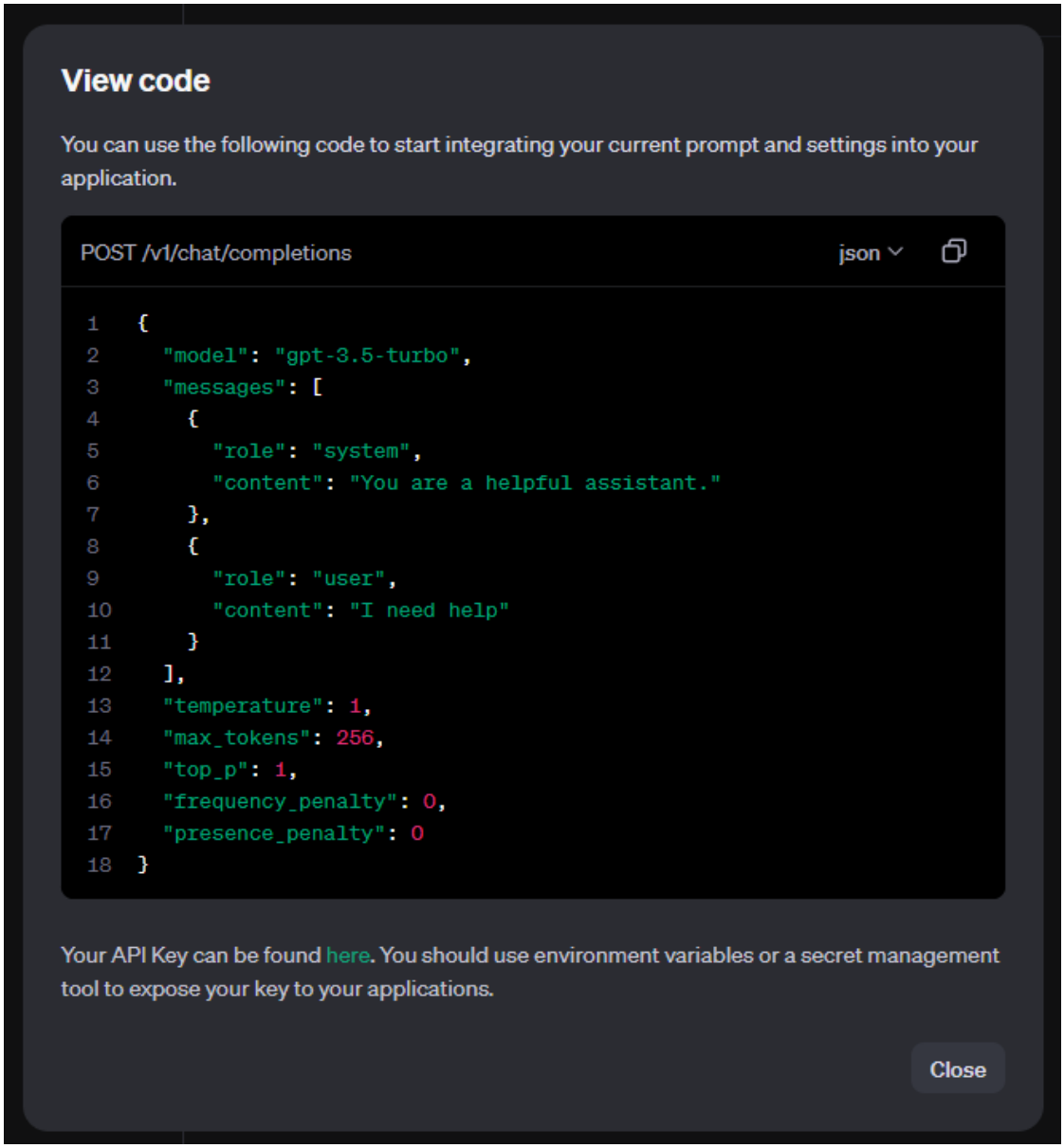


Figure 5.50 ChatGpt Json documentation

To begin, the developer accessed the OpenAI dashboard to gain insights into the necessary code and API calls required to communicate with the chatbot API. Familiarizing themselves with the endpoint and syntax of the API, the developer initiated simple API requests to establish a foundational understanding. From there, the developer progressively advanced, gradually building upon these initial interactions to create a fully functional chat component within the application. This iterative approach allowed the developer to methodically develop the chatbot feature, starting with basic functionality and gradually incorporating more advanced capabilities. By enabling users to input messages and receive responses from the API, the chatbot adds an interactive and dynamic element to the application, enhancing its overall utility and user experience. Through dedicated effort and strategic implementation,

the developer successfully integrated the chatbot feature into the project, aligning with the supervisor's recommendation to enrich the application with additional functionality.

```
export const Chatbot = () => {
  const { highContrast } = useContext(HighContrastContext);
  const [chatHistory, setChatHistory] = useState(() => {
    const storedChatHistory = localStorage.getItem("chatHistory");
    return storedChatHistory ? JSON.parse(storedChatHistory) : [];
  });
  const [inputText, setInputText] = useState("");
  const [loading, setLoading] = useState(false);
  const chatHistoryRef = useRef(null);

  const apiKey = "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";

  const saveChatHistoryToLocalStorage = (history) => {
    localStorage.setItem("chatHistory", JSON.stringify(history));
  };

  const addToChatHistory = (message) => {
    setChatHistory((prevHistory) => [...prevHistory, message]);
    saveChatHistoryToLocalStorage([...chatHistory, message]);
  };

  const sendMessage = async () => {
    setLoading(true);
    const data = {
      model: "gpt-3.5-turbo",
      messages: [
        { role: "system", content: "You are a helpful assistant." },
        ...chatHistory,
        { role: "user", content: inputText },
      ],
      temperature: 1,
      max_tokens: 256,
      top_p: 1,
      frequency_penalty: 0,
      presence_penalty: 0,
    };
```

Figure 5.51 Chatbot component using ChatGpt

```
return (
  <div
    className={`mx-4 mt-10 border-4 p-4 rounded-lg ${
      highContrast ? "dark" : ""
    }`}>
    <h2 className="text-3xl font-bold mb-4">CHATBOT</h2>
    <div
      className="chat-history mb-4"
      ref={chatHistoryRef}
      style={{ maxHeight: "300px", overflowY: "auto" }}>
      {chatHistory.map((message, index) => (
        <div
          key={index}
          className={message.role === "user" ? "text-right mb-2" : "mb-2"}>
          <span
            className={`inline-block p-2 rounded-lg ${
              message.role === "user"
                ? " bg-regal-pink  dark:bg-regal-blue  text-white"
                : " bg-gray-300  text-gray-700"
            }`}>
            {message.content}
          </span>
        </div>
      ))}
    </div>
    <div className="flex items-center">
      <input
        type="text"
        placeholder="Ask me anything"
        value={inputText}
        onChange={handleChange}
        onKeyDown={handleKeyDown}
        className="flex-grow mr-2 p-2 rounded-lg border  border-gray-400 focus:outline-none"
      />
      <button
        onClick={sendMessage}
        disabled={!inputText || loading}
        className={`p-2 rounded-lg ${
          !inputText || loading
            ? " bg-gray-400 cursor-not-allowed"
            : " bg-regal-pink  hover:bg-regal-pink  dark:bg-regal-blue  text-white"
        }`}>
        {loading ? "Sending..." : "Send"}
      </button>
    </div>
  </div>
);
```

Figure 5.52 User interface for the chatbot component

To enhance the user experience even further, the developer implemented a feature to save chat history locally. This functionality allows users to retain a record of their conversations within the application, facilitating continuity and convenience. By leveraging local storage capabilities, the developer ensured that chat transcripts are preserved locally, enabling users to reference past interactions effortlessly. Whether revisiting previous conversations for reference or resuming discussions where they left off, this feature empowers users with greater control and flexibility over their chat experience. Moreover, saving chat history locally contributes to a more seamless and personalized user experience. Users can track their interactions with the chatbot over time, enabling them to monitor progress, review information, and reflect on past discussions. This not only fosters a sense of continuity but

also promotes engagement and retention within the application. Overall, the implementation of local storage for chat history represents a thoughtful enhancement aimed at improving usability and enhancing the overall user experience. By prioritizing user convenience and accessibility, the developer has further solidified the application's value proposition and utility for its users.

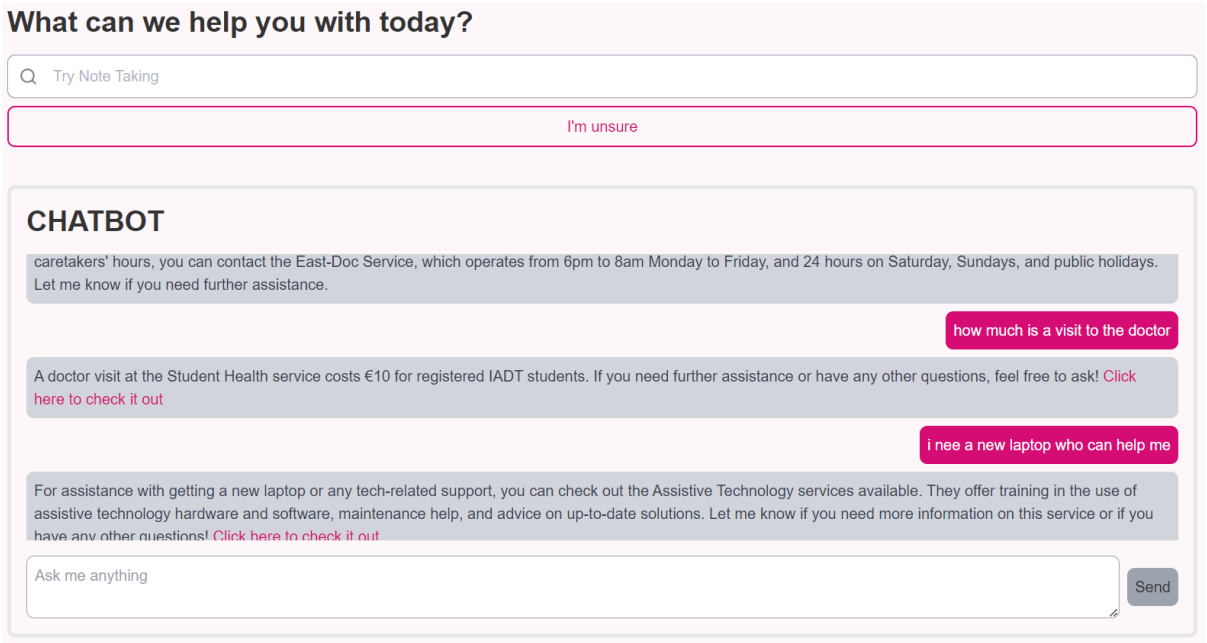


Figure 5.53 Visual display of the chatbot and search on the homepage

The developer found time to mock up content for articles to be added to the platform, the articles content is the same as the services, so it was just a case of copying the same files and adjusting the variables accordingly. The only major change was the search logic now the search will work for both services and articles this was achieved by calling the articles from the API and storing them in state variables so they can be manipulated on the front end like the services, this is shown in the image below.

```
const [searchedArticles, setSearchedArticles] = useState([]);
const { highContrast } = useContext(HighContrastContext);

useEffect(() => {
  setLoading(true);
  const articlePromise = axios.get(
    "https://strapiv6336web.azurewebsites.net/api/articles?populate=*"
  );
  const servicePromise = axios.get(
    "https://strapiv6336web.azurewebsites.net/api/services?populate=*"
  );

  Promise.all([articlePromise, servicePromise])
    .then(([articlesResponse, servicesResponse]) => {
      setArticles(articlesResponse.data.data);
      setServices(servicesResponse.data.data);
      // console.log(articlesResponse.data.data);
      // console.log(servicesResponse.data.data);
    })
    .catch((error) => {
      console.error("Error fetching data:", error);
    })
    .finally(() => {
      setLoading(false);
    });
}, []);
```

Figure 5.54 Axios call to get services and articles information

The articles are displayed if the array is greater than 0 meaning if there are no articles that match the search nothing will be displayed.

```
{mergedArticles.length > 0 ? (
  <>
    <p className=" font-semibold text-gray-400 ">
      Articles for you
    </p>
    {mergedArticles.map((article, index) => (
      <div
        key={index}
        className="shadow-lg p-4 rounded-lg mb-4"
      >
        <div className="md:flex md:flex-col">
          <Link
            to={`/article-view/articles/${article.attributes.slug}`}
          >
            <div className="md:w-full mt-4">
              <h2 className="text-left text-xl font-semibold mb-2 underline dark:text-regal-blue text-regal-pink">
                {article.attributes.Title}
              </h2>
              {article.attributes.Content &&
                article.attributes.Content.length > 0 && (
                  <p>
                    {article.attributes.Content[0].BulletPoint.split(
                      " "
                    )
                      .slice(0, 50)
                      .join(" ")}
                  </p>
                )}
            </div>
          </Link>
        </div>
      </div>
))}
```

Figure 5.55 Code to show articles

Now on the front end the users have a variety of content they can view.

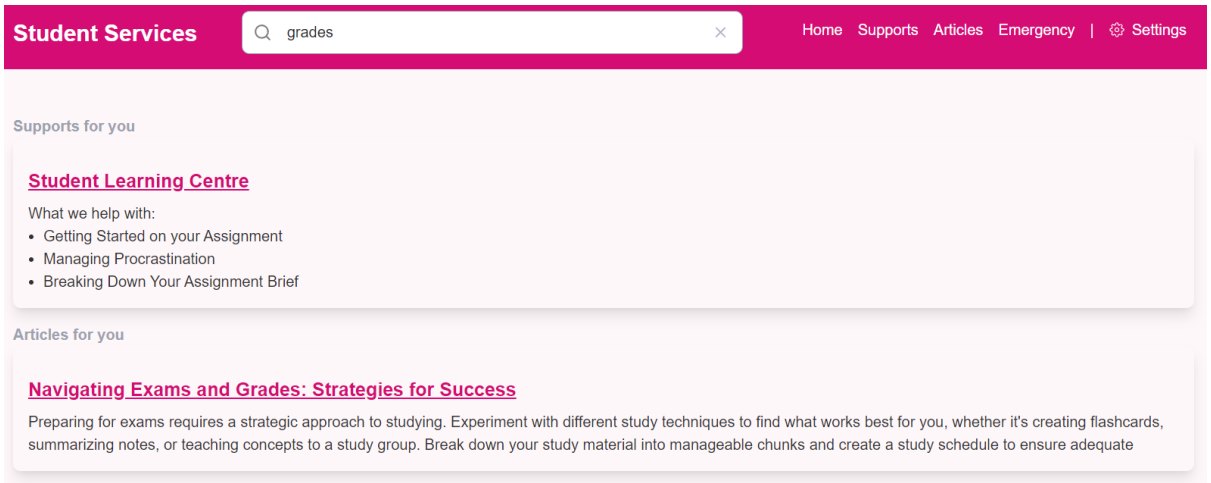


Figure 5.56 Results page showing both services and articles relevant to the search query

If the user clicks on the article, they will see the article view page which holds the information on the respective articles.

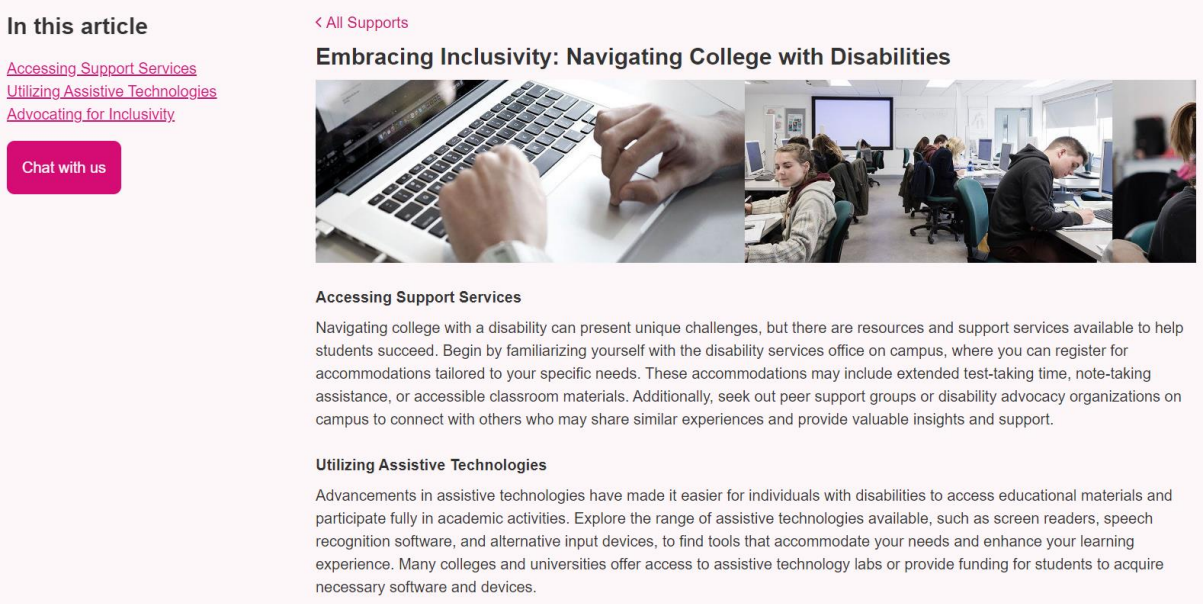


Figure 5.57 Article view page

5.13.3 Item 2

The developer conducted app testing by gathering data from a diverse range of users, comprising individuals with varying levels of familiarity with the app and technology in general. Some participants possessed prior experience with the application or technology, while others were inexperienced. Each user was assigned two tasks: first, to locate information about grades using the search functionality, and second, to find such information without using the search feature. These tasks were timed individually, and users completed them without assistance. Users were encouraged to vocalize their thought processes during the tasks, providing reasoning behind their actions. The developer recorded users' verbal feedback and noted the time taken to complete each task. Additionally, participants were provided with a Google form to share their experiences and offer overall feedback on the application. The findings from this testing are detailed in the testing section.

The developer observed that users were notably quicker in locating information when utilizing the search functionality. A comparison of average completion times revealed that users who utilized the search feature took half the time compared to those who did not.

A crucial factor in all applications developed is the impact it has on the environment and how sustainable it is. The developer took the time to understand how the application built preforms in the aspect of sustainability the results are shown below.

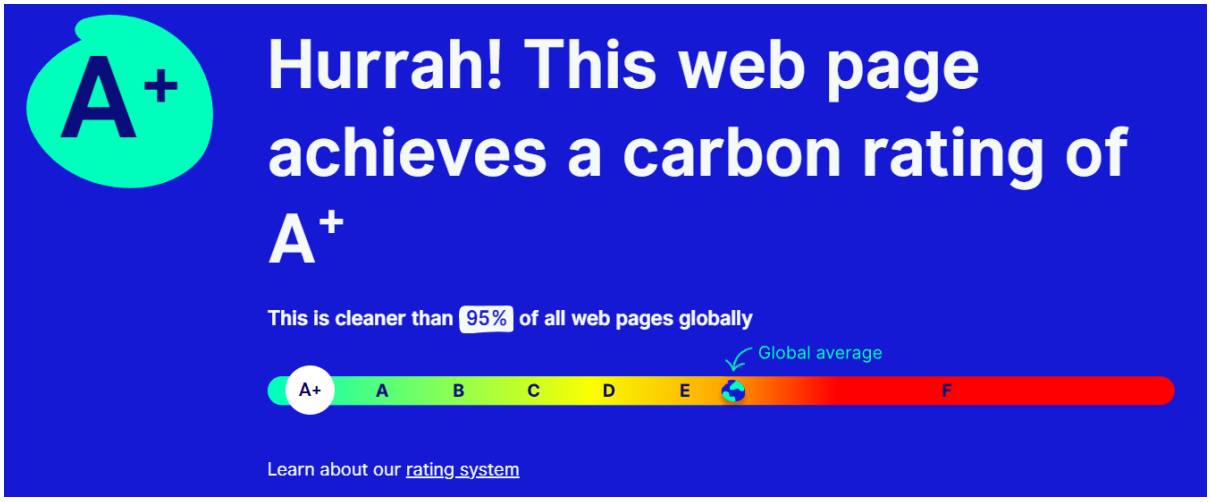


Figure 5.58 Carbon rating for application

The application has received an A+ carbon rating meaning that it is sustainable and follows the best practices in the world of web development, which is imperative in this project as it is being developed for a major institution being IADT.

5.14 Conclusion

This chapter holds significant importance within the project as it serves as a documentation of the app development process. Through this documentation, the developer gained valuable insights into problem-solving techniques and task management strategies. Specifically, the developer learned how to effectively break down complex problems into smaller, more manageable tasks, enabling smoother progress and more efficient problem resolution.

Furthermore, the chapter highlights the iterative nature of the project, wherein the app underwent several changes and adaptations to accommodate new requirements and features. This flexibility and adaptability were crucial in responding to evolving needs and ensuring that the final product aligned with stakeholder expectations.

Overall, this chapter not only chronicles the development journey of the app but also underscores the developer's growth and learning throughout the process. By documenting challenges faced, solutions implemented, and lessons learned, the developer has created a valuable resource for future reference and continuous improvement.

6 Testing

6.1 Introduction

This chapter describes the testing that has been undertaken for the application. This chapter is presented in two sections:

1. Functional Testing
2. User Testing

Functional testing is a type of software testing whereby the system is tested against the functional requirements. The app is tested by looking to see if the actual output for a given input corresponds with the expected output. The tests should be based on the requirements for the app. The results of functional testing can indicate if a piece of software is functional and working, but not if the software is easy to use.

This phase of testing is dedicated to evaluating the ease with which users can interact with the application interface and navigate through its functionalities. By soliciting feedback from representative users and observing their interactions with the software, user testing provides invaluable insights into the user experience (UX) aspects of the application.

The developer decided to do functional tests on the technical functional requirements and run user tests on the non-functional requirements in the application.

6.2 Functional Testing

The functional requirements listed below represent the cornerstone of the application's foundation, serving as the primary features intended for user interaction. These requirements underwent rigorous testing by the developer to ensure their reliability and functionality.

- Search
- Bookings
- CRUD

6.2.1 Search

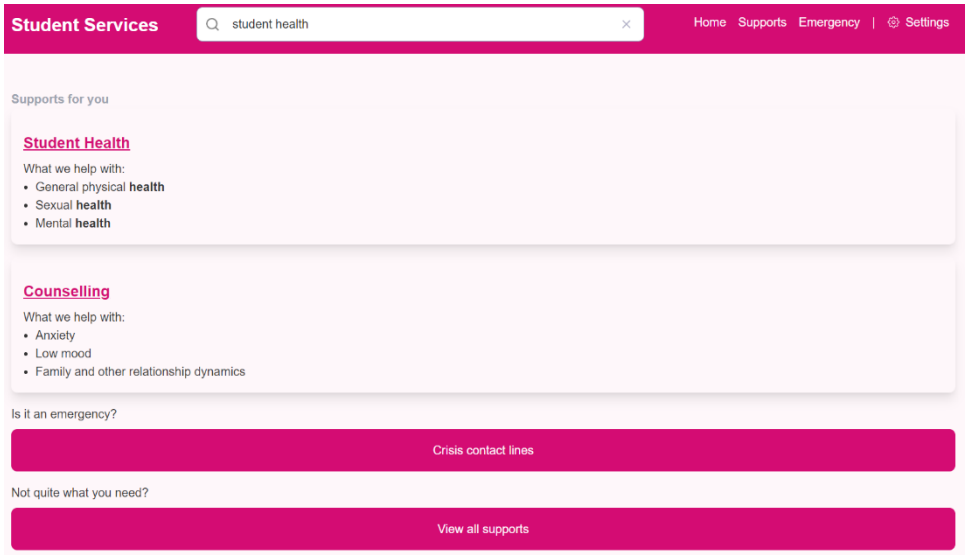


Figure 6.1 Results page

Test No	Description of test case	Input	Expected Output	Actual Output	Result
1	Search for content known to be in the backend by name	Searched for content in the backend	The user should see the results page and see relevant results	The user should see the results page and see relevant results.	Passed
2	Search for content known to be in the backend by name. Search for content known to be in the backend through a term	Searched a general term in the backend	The user should see the results page and shown the service relevant to the term	The user should see the results page and shown the service relevant to the term.	Passed
3	Search for a non-related term	Searched a random term not related to any of the services	The user will be shown the results page and a no results component	The user will be shown the results page and a no results component.	Passed

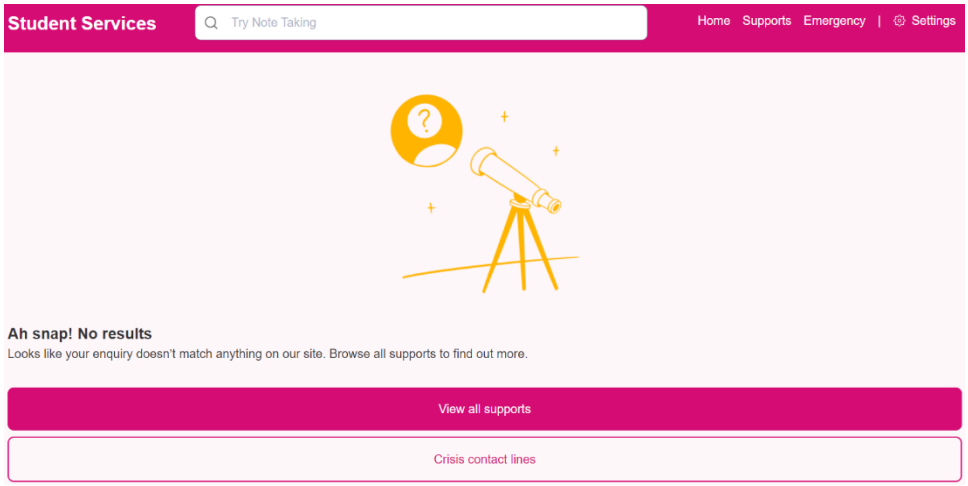


Figure 6.2 Results page showing no results component

6.2.2 CRUD

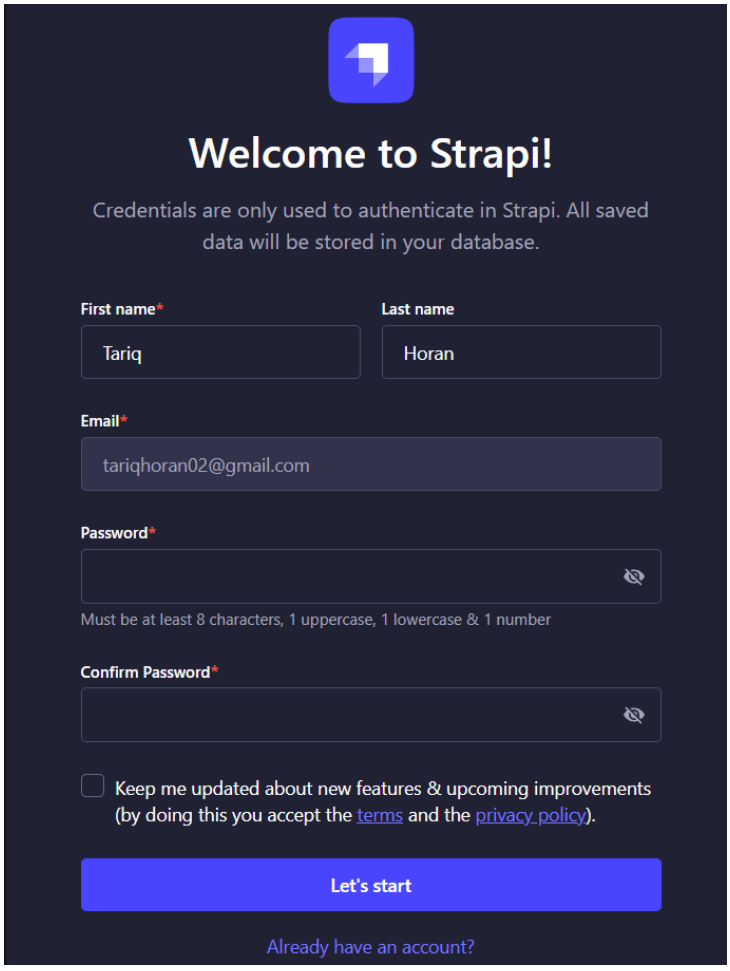


Figure 6.3 Register page for new users in Strapi

Test No	Description of test case	Input	Expected Output	Actual Output	Result
1	Admin logs in to CMS	Admin attempts to access the CMS through a link	Admin logs in or registers to the CMS	Admin logs in or registers to the CMS	passed
2	Admin CRUD	Admin access the Strapi dashboard then creates, views, updates, and deletes content	Content in the backend is manipulated by the admin	Content in the backend is manipulated by the admin.	passed

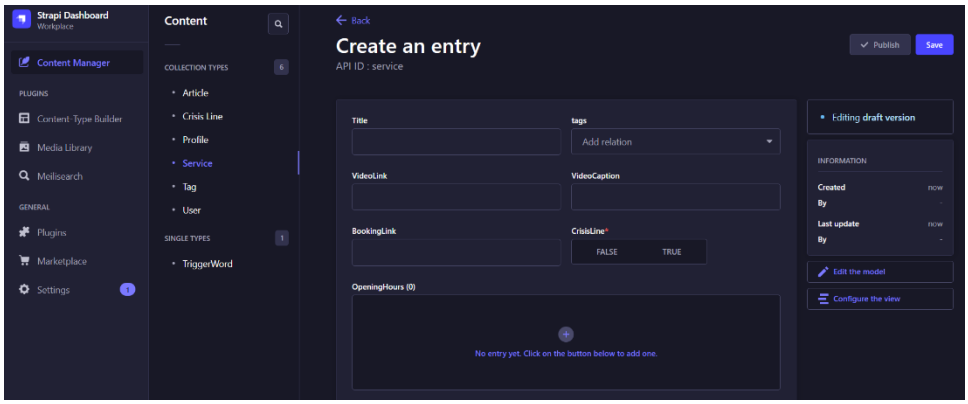


Figure 6.4 Creating a new entry in Strapi

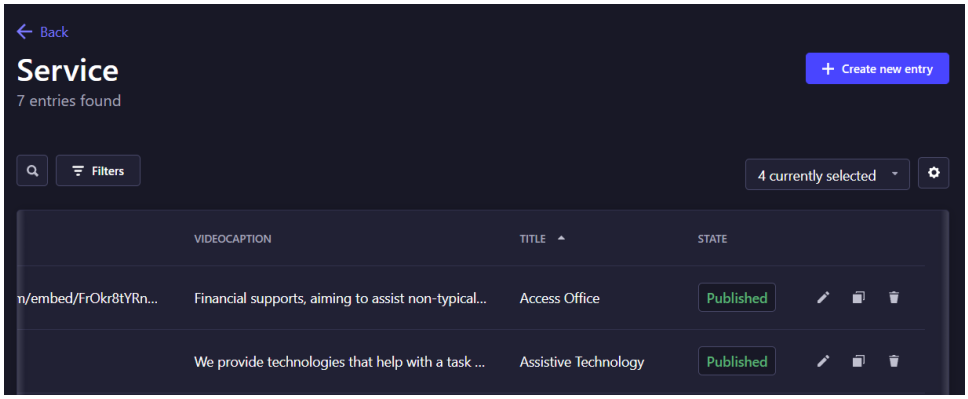


Figure 6.5 Services published in the backend

Test No	Description of test case	Input	Expected Output	Actual Output	Result
1	Install website as an app	User clicks install button in the URL	App downloads to users' desktop	App downloads to users' desktop	passed

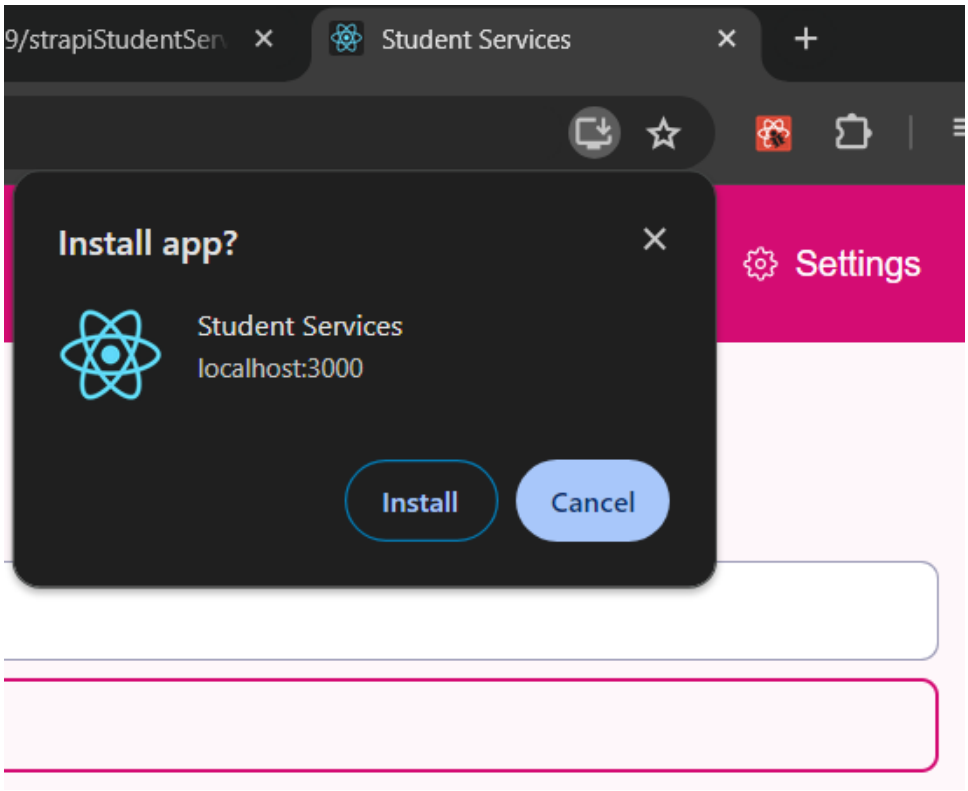


Figure 6.6 Installing the application in the browser

6.2.3 Discussion of Functional Testing Results

The functional testing results demonstrate the thorough examination and validation of key features within the application.

Search Functionality:

Test 1: This test successfully verifies that the search function retrieves relevant results when querying content known to be in the backend by name. The expected outcome aligns with the actual output, indicating that the search feature functions as intended.

Test 2: Similarly, the search functionality performs well when using a general term to retrieve relevant services. The expected outcome matches the actual output, indicating successful functionality.

Test 3: The search feature correctly handles non-related terms by displaying a no-results component. This ensures that users receive appropriate feedback when searching for content that does not exist.

CRUD Operations:

Test 1: The test confirms that administrators can successfully log in to the CMS, ensuring secure access to content management features.

Test 2: Admin CRUD operations are validated, showing that administrators can create, view, update, and delete content within the backend. This ensures the effective management of content by authorized personnel.

Installation:

Test 1: The installation process of the website as an app is seamless, with the application successfully downloading to the user's desktop upon clicking the install button in the URL.

6.3 User Testing

As the project progresses, it has reached a crucial phase where user testing becomes paramount. The application is now ready to undergo comprehensive testing by end-users to identify any potential bugs, usability issues, or areas for improvement. User testing serves as a vital feedback mechanism, enabling the developer to gain valuable insights into how users interact with the application and what adjustments may be required to enhance its functionality and user experience.

The developer conducted app testing by gathering data from a diverse range of users, comprising individuals with varying levels of familiarity with the app and technology in general. Some participants possessed prior experience with the application or technology, while others were inexperienced. Each user was assigned two tasks first, to locate information about grades using the search functionality, and second, to find such information without using the search feature. These tasks were timed individually, and users completed them without assistance. Users were encouraged to vocalize their thought processes during the tasks, providing reasoning behind their actions. The developer recorded users' verbal feedback and noted the time taken to complete each task. The developer observed that users were notably quicker in locating information when utilizing the search functionality. A comparison of average completion times revealed that users who utilized the search feature took half the time compared to those who did not.

The table below shows the data collected from the users.

User Id	Search task time	Navigation task time	Search task thoughts	Navigation task thoughts
1	5 seconds	20 seconds	“I like how quick it is to find the information it took me straight to it”	“I rather the search but I got there in the end”
2	8 seconds	15 seconds	“The autocomplete part is cool I like the suggestions as I type”	“The search is better in my opinion, but it is a good alternative”
3	8 seconds	18 seconds	“Quick and easy found what I was looking for efficiently”	“If I didn’t find it through search I probably would have given up”
4	10 seconds	21 seconds	“Basic but it works I found what I wanted”	“I liked this way more I like to browse”
5	9 seconds	12 seconds	“I found it was easy to understand and to use”	“I prefer the search, but I found the data quickly”

After the users completed the tasks, they were permitted to use the app fully and provide any feedback that they felt necessary. Additionally, participants were provided with a Google

form to share their experiences and offer overall feedback on the application. The following section outlines the findings from the user testing.

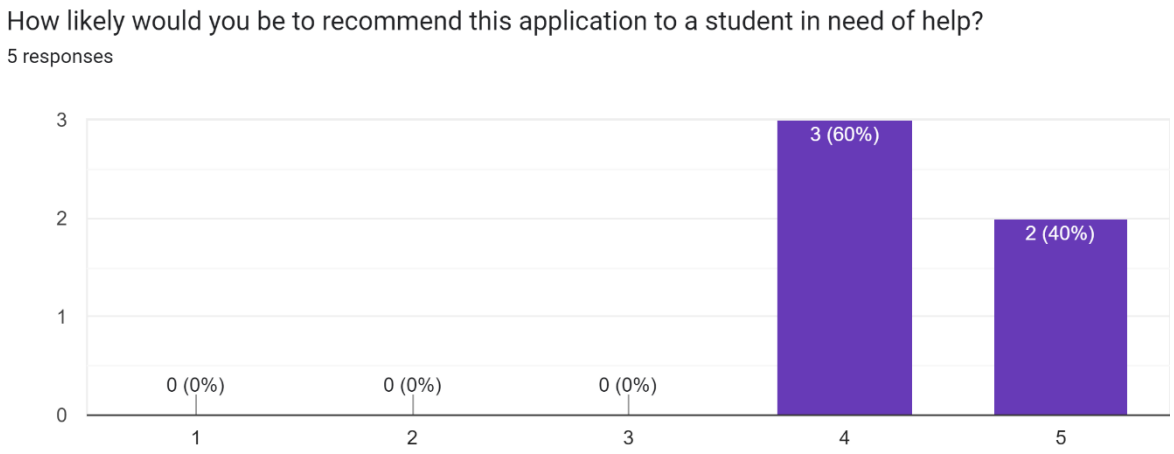


Figure 6.7 Recommendation question. Google form

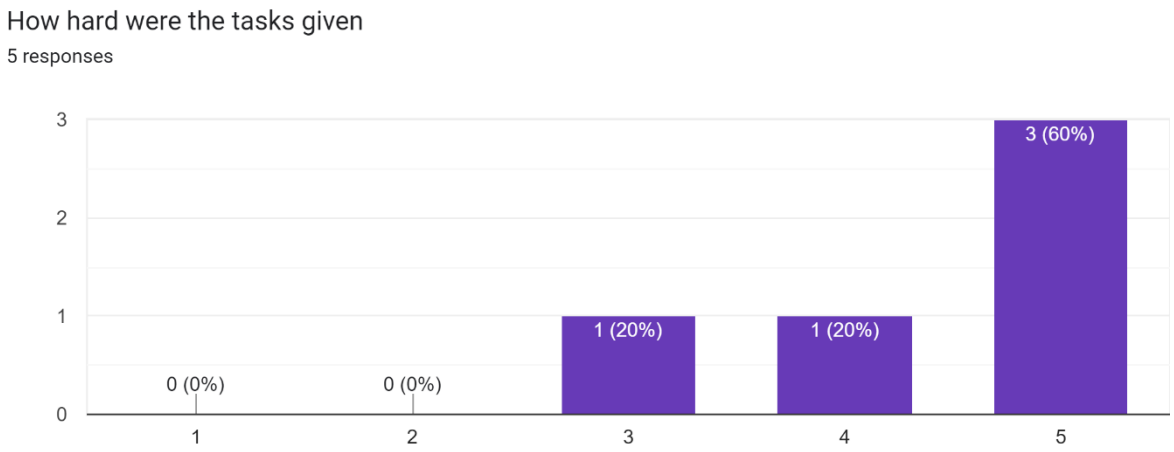


Figure 6.8 Difficulty question. Google form

The feedback from users who tested the app indicates a positive experience overall. Users expressed satisfaction with various aspects of the application, highlighting its usability, functionality, and effectiveness in meeting their needs. This positive reception is a testament to the developer's efforts in designing and implementing a user-friendly and intuitive application.

Additionally, users provided constructive feedback and suggestions for improvement, demonstrating their engagement and willingness to contribute to the enhancement of the app.

This valuable input will be instrumental in further refining the application and ensuring that it continues to deliver a positive user experience.

Overall, the positive responses from users affirm the success of the app in fulfilling its intended purpose and underscore the developer's commitment to creating a high-quality product that meets the needs of its users. Moving forward, the developer will continue to incorporate user feedback and iterate on the application to further enhance its usability and functionality.

6.4 Conclusion

Overall, the functional testing results indicate that the critical features of the application, including search functionality, content management, and installation, perform as expected without any significant issues. These tests validate the reliability and functionality of key features, ensuring a positive user experience and the smooth operation of the application.

7 Project Management

7.1 Introduction

In managing the intricate facets of the software project, the developer adopted a two-pronged approach, leveraging both a GitHub repository and a Miro board Kanban system. This strategic combination allowed for efficient tracking of features, seamless problem-solving, and agile bug fixes. By utilizing GitHub's robust version control capabilities, the developer could collaborate effectively on code development, ensuring a unified and organized codebase. Concurrently, the Miro board Kanban provided a visual representation of tasks, enabling the developer to prioritize and manage workflow with ease. Together, these tools facilitated effective project management, enabling the developer to navigate the project's complexity while maintaining focus on delivering high-quality results.

7.2 Project Phases

The project progressed through distinct stages, each carefully aligned with a predefined timeline, ensuring that deadlines for each section were consistently met. This structured approach enabled the developer to manage the project's complexity effectively and maintain momentum throughout its duration. By breaking the project down into manageable stages, the developer could focus on specific objectives, allocate resources efficiently, and monitor progress closely. Additionally, adhering to a set timeline helped instil a sense of accountability and urgency, ensuring that tasks were completed promptly, and milestones were achieved on schedule. Overall, this systematic and disciplined approach to project management played a pivotal role in the successful completion of each stage within the designated period.

7.2.1 Proposal

Initially uncertain about the project's focus, the developer contemplated research on project and development lifecycles. However, a shift occurred towards a more compelling interest in search algorithms and their potential benefits for applications. This pivot led to a project direction centred on exploring the workings and applications of search algorithms, aiming to optimize functionality within the project and beyond.

7.2.2 Requirements

This section provided the developer with an unclouded vision of the completed project, offering a tangible understanding of its eventual form. While not all initially proposed

features were implemented, the process of ideation and conceptual development played a crucial role in bringing the project to fruition. Through brainstorming and refining ideas, the developer gained valuable insights into the project's scope and direction, laying the groundwork for its successful realization. Despite the evolution of the project's features and functionalities during development, the initial ideation phase served as a catalyst for progress, shaping the project's trajectory and ensuring its alignment with overarching objectives.

7.2.3 Design

The design chapter paralleled the requirements phase by providing the project with structure and delineating its various components. During this stage, the project transitioned from conceptual ideas to tangible manifestations, with each component taking on a defined form. By outlining the project's design, including its architecture, user interface, and interaction flows, the developer brought the project to life, imbuing it with substance and functionality. Just as the requirements phase established the project's objectives and scope, the design phase served as a blueprint for its implementation, guiding the development process and ensuring coherence across all elements. Overall, the design chapter marked a crucial milestone in the project's evolution, transforming abstract concepts into concrete realities and setting the stage for its successful execution.

7.2.4 Implementation

This pivotal phase marked the project's transformation from concept to reality, as the developer dedicated critical time and effort to its construction. Like any software development endeavour, meticulous attention was paid to debugging code and ensuring the project remained on course. The primary focus of this chapter cantered on building new components while simultaneously maintaining existing ones. This balancing act required adept management of resources and priorities to ensure the project's overall integrity and functionality. As the developer navigated the intricacies of coding and problem-solving, each line of code contributed to the project's evolution, bringing it closer to its envisioned state. Despite the inevitable challenges encountered along the way, the commitment to building and maintaining project components remained unwavering, underscoring the dedication and perseverance driving its progress. This chapter epitomized the culmination of effort and dedication, propelling the project forward towards its ultimate realization.

7.2.5 Testing

This section brought to light several challenges within the project, prompting the developer to reflect on its progress up to that point. Confronted with these issues, the developer was compelled to act swiftly to address and mitigate the problems that had arisen. This necessitated a thorough examination of the project's current state, identifying areas of weakness and devising effective solutions to rectify them. With a sense of urgency, the developer implemented strategic measures to nullify the problems encountered, ensuring that the project remained on track and aligned with its objectives. This phase of reflection and proactive problem-solving underscored the developer's commitment to delivering a high-quality product, demonstrating resilience and adaptability in the face of adversity. Through decisive action and careful consideration, the developer navigated through challenges, paving the way for the project to continue its onward trajectory towards successful completion.

7.3 Teamwork

Most of the project's development was undertaken by the developer, who contributed significantly to its creation. Throughout the process, support and guidance were provided by project supervisors, whose expertise and oversight proved invaluable in navigating challenges and ensuring alignment with project objectives. Additionally, the involvement of designer Kacper played a crucial role in shaping the project's visual identity and user experience. Together, this collaborative effort between the developer, project supervisors, and designer facilitated the project's progress and contributed to its overall success. Each member brought unique skills and perspectives to the table, complementing one another's contributions, and fostering a dynamic and productive working environment.

7.3.1 Communication

The developer maintained an open line of communication with all stakeholders involved in the project. This was achieved through weekly meetings with the project supervisor, during which any issues or challenges encountered were promptly discussed and addressed. Additionally, these meetings served as an opportunity to provide updates on the project's progress and advancements, ensuring transparency and alignment with project objectives.

Similarly, regular communication was maintained with the designer to stay informed about design developments and ensure consistency between the project's technical aspects and its visual elements. By keeping abreast of design updates and incorporating them into the development process, the developer ensured that the project's overall aesthetic and user experience remained cohesive and aligned with the intended vision.

7.3.2 Difficulties

The developer approached software development with a passion for coding and a willingness to tackle challenges head-on. Encountering issues along the way, the developer embraced them with a positive attitude, viewing them as opportunities for growth and learning. Any mistakes made were promptly acknowledged and rectified, reflecting a commitment to delivering high-quality results. Despite encountering various difficulties throughout the project, the developer's main challenge lay in documenting the process in a formal manner. While proficient in coding and technical aspects, the developer found it challenging to articulate the project's progress and methodologies in a structured and formalized format. Nonetheless, recognizing the importance of documentation in software development, the developer made concerted efforts to improve in this area, seeking guidance and support as needed.

7.3.3 Resolving Difficulties

The developer demonstrated a strong commitment to personal and professional growth by actively seeking opportunities to push beyond their comfort zone and focus on areas where they were less confident. Recognizing the importance of addressing these weaker aspects, the developer placed a significant emphasis on improving and expanding their skills. A key strategy employed by the developer was to seek guidance and support from the project supervisor when faced with challenges or uncertainties. By actively listening to suggestions and advice, the developer showed a willingness to learn from others' expertise and incorporate valuable insights into their approach. This proactive approach to seeking mentorship and guidance not only facilitated the developer's personal development but also contributed to the overall success of the project. Through collaboration and a willingness to step outside their comfort zone, the developer was able to strengthen their skills, broaden their expertise, and deliver a high-quality product.

7.4 SCRUM Methodology

The project employed the Scrum methodology, a well-established approach in project management, which proved to be highly effective. Given the customary practice of utilizing Scrum in software development projects, the developer found it to be a practical and familiar framework to follow. Drawing from previous experience with Scrum, the developer seamlessly integrated its principles and practices into the project workflow. The use of sprints within the Scrum framework provided clear, achievable goals for the developer to focus on

each week. By breaking down the project into manageable increments, the sprints ensured that progress remained on track and that the project advanced steadily towards its objectives. This iterative approach allowed the developer to adapt to changing requirements and priorities, while maintaining a consistent pace of development. Overall, the combination of Scrum methodology and sprint-based planning proved to be instrumental in driving the project's success. It provided structure, clarity, and accountability, empowering the developer to efficiently manage tasks and deliver results within a dynamic and collaborative environment.

7.5 Project Management Tools

7.5.1 Miro

Utilizing Miro as a Kanban board tool, the developer efficiently managed the project's feature backlog. At the project's inception, all planned features were added to the backlog column, providing a comprehensive overview of the project's scope and requirements. As development progressed, features in active development were transitioned to the "in progress" column, while completed features were moved to the "done" column. This Kanban workflow within Miro provided the developer with a clear visualization of how the feature backlog was being addressed and cleared over time. It facilitated efficient prioritization of tasks and enabled the developer to focus on completing features in a systematic manner. By tracking progress through each stage of development, Miro served as a valuable tool for monitoring the project's advancement and ensuring that key milestones were achieved.

7.5.2 GitHub

GitHub served as a pivotal platform for storing the project's codebase, with two distinct repositories dedicated to frontend and backend development. Each repository was seamlessly integrated with Azure resources, enabling automated redeployment whenever changes were pushed to the repository. This streamlined workflow ensured that updates made in the repositories were promptly reflected in the deployed application. The utilization of GitHub provided several benefits, notably in facilitating efficient code management and deployment processes. The platform enabled the developer to track code changes made in both repositories, offering a comprehensive history of modifications. Furthermore, GitHub's robust error reporting mechanisms proved invaluable, providing insightful error messages in the event of deployment failures. This functionality allowed for quick identification and resolution of issues, ensuring smooth and reliable deployment processes. In summary,

GitHub played a crucial role in the project's development lifecycle, offering a centralized and organized platform for code management and deployment. Its integration with Azure resources streamlined the deployment process, while its error reporting capabilities proved instrumental in troubleshooting and resolving issues efficiently.

7.6 Reflection

7.6.1 Your views on the project

In summary, the project's outcome is deemed a major success by the developer. Despite encountering challenges and numerous changes throughout each stage of development, the project achieved its objectives. Adapting to shifting priorities and switching focus between unrelated tasks posed difficulties at times, yet the developer's skillset, dedication, and hard work ensured the project's success. While the project may appear simple on the surface, the developer emphasizes the complexity of the work accomplished behind the scenes. The high degree of development undertaken contributed to a seamless user interface, masking the intricacies and complexities involved. The project's success is attributed to the developer's proficiency, willingness to code, and relentless effort, resulting in a product that meets or exceeds expectations.

7.6.2 Working with a supervisor

Working with a supervisor like Mohammed proved to be immensely valuable for the developer. The guidance and support provided by Mohammed were instrumental in navigating the challenges and complexities of the project. With Mohammed's expertise and availability, the developer found an invaluable resource for both technical assistance and strategic direction. Mohammed's supportive presence extended beyond mere code review, encompassing a broader perspective on the project's overarching goals and vision. This comprehensive approach ensured that the project not only met technical requirements but also aligned with broader objectives and considerations. Overall, the collaboration between the developer and Mohammed facilitated the project's progression from a mere idea to a high-level application. Mohammed's unwavering support, guidance, and availability played a pivotal role in overcoming obstacles, refining ideas, and realizing the project's full potential.

7.6.3 Technical skills

Throughout the project, the developer underwent significant learning and growth experiences. In frontend development, they honed their skills in writing clean, efficient code that enhances functionality and user experience. This emphasis on code quality not only ensures a smoother

development process but also results in a more maintainable and scalable application. Venturing into backend development with the utilization of a CMS was a new experience for the developer. Despite encountering challenges and making mistakes along the way, this experience proved invaluable in expanding their knowledge base and skill set. Through trial and error, they gained insights into backend development best practices and learned valuable lessons that will inform their future projects. Technically, the developer's most significant learnings revolved around bug fixing and customization of components to meet project requirements. Addressing bugs and errors requires a meticulous approach to troubleshooting and problem-solving, further refining their analytical and debugging skills. Additionally, customizing components to align with project specifications highlights their adaptability and resourcefulness in tailoring solutions to meet specific needs. Overall, the project served as a rich learning experience, equipping the developer with a diverse set of skills and insights that will prove invaluable in future endeavours. Through perseverance, experimentation, and a willingness to learn from mistakes, they have emerged as a more capable and well-rounded developer.

7.7 Conclusion

In conclusion, the project management chapter highlights the meticulous planning, execution, and reflection of the software project. Leveraging a combination of GitHub and Miro, ensuring efficient tracking of features, seamless problem-solving, and agile bug fixes. The structured approach adopted throughout the project's phases, from the initial project proposal to implementation, facilitated effective project management, enabling the developer to navigate challenges and meet deadlines consistently. Notably, the collaboration with supervisors and the utilization of Scrum methodology further enhanced project efficiency and effectiveness. The chapter highlighted various aspects of project management, including proposal ideation, requirements gathering, design implementation, and testing. Throughout the project's lifecycle, the developer demonstrated resilience, adaptability, and a commitment to delivering high-quality results. Overall, the project management chapter encapsulates the meaning of software project management.

8 Business Opportunities

The application, initially developed for a business, represents the first iteration of what promises to be a series of enhancements and expansions. There are several avenues for further development that could significantly enhance the system's effectiveness and user experience. One potential enhancement involves the addition of a repository of useful articles tailored for students. By providing informative and educational content, the application can serve as a valuable resource for students seeking supplementary materials related to their studies. This addition not only enriches the user experience but also adds substantial value to the application, positioning it as a comprehensive educational platform. Furthermore, integrating a chatbot feature into the application presents another opportunity for improvement. A chatbot tailored to address students' needs and provide support during moments of crisis can enhance the system's functionality. Whether offering guidance on academic matters, providing resources for mental health support, or connecting users with relevant services, the chatbot adds an interactive and responsive element to the application, fostering a more supportive and user-centric environment. Incorporating these enhancements aligns with the overarching goal of continuously improving the application to better meet the needs of its users. By leveraging technology and innovation, the application can evolve into a dynamic and indispensable tool for students, offering not only academic support but also valuable resources for personal development and well-being. As the application continues to grow and adapt to user feedback and changing needs, it has the potential to become a cornerstone resource for students seeking support and guidance throughout their academic journey.

9 Conclusion

The primary aim of the application was to enhance the accessibility of information for students, surpassing the existing offerings provided by IADT. Through the development of a user-friendly interface, the application achieved this objective by providing students with a seamless and intuitive platform for accessing information. By prioritizing usability and accessibility in its design, the application ensures that students can navigate through its features effortlessly, finding the information they need with ease. By employing scalable and modular design principles, the application lays the groundwork for future enhancements and updates. In summary, the application's focus on user-friendliness, scalability, and accessibility underscores its commitment to empowering students with easy access to information. By creating a platform that is not only intuitive and adaptable but also inclusive and accessible, the application sets a new standard for information accessibility within the academic community.

10 References

Mahnke, M. S. (2014). Algorithming the algorithm. In Society of the query reader: Reflections on web search.

Boyer, R. S., & Moore, J. S. (1977). A fast string searching algorithm. Communications of the ACM, 20(10), 762-772.

M ahesh, B. (2020). Machine learning algorithms-a review.

International Journal of Science and Research (IJSR). [Internet], 9(1), 381-386.

GeeksforGeeks. (2023, September 26). Searching algorithms. GeeksforGeeks.

https://www.geeksforgeeks.org/searching-algorithms/

Sultana, N., Paira, S., Chandra, S., & Alam, S. S. (2017, February). A brief study and analysis of different searching algorithms. In 2017 second international conference on electrical, computer and communication technologies (ICECCT) (pp. 1-4). IEEE.

Ingram, G., & Zhang, T. (2009). Overview of applications and developments in the harmony search algorithm. Music-Inspired Harmony Search Algorithm: Theory and Applications, 15-37.

What is Information Retrieval? | A Comprehensive Information Retrieval (IR) Guide. (2023). Elastic.co; Elastic. https://www.elastic.co/what-is/information-retrieval

Roshdi, A., & Roohparvar, A. (2015). Information retrieval techniques and applications. International Journal of Computer Networks and Communications Security, 3(9), 373-377.

Liu, T. Y. (2009). Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval, 3(3), 225-331.

Carpineto, C., & Romano, G. (2012). A survey of automatic query expansion in information retrieval. Acm Computing Surveys (CSUR), 44(1), 1-50.

Engelbrecht, A. P. (2007). Appendix A: Optimization Theory. Computational Intelligence, 551–579. https://doi.org/10.1002/9780470512517.app1

Randall, M. (2020, April). Heuristic Search Part 1: Introduction and Basic Search Methods – Matt Randall. Lancaster.ac.uk. https://www.lancaster.ac.uk/stor-i-student-

sites/matthew-randall/2020/04/01/heuristic-search-part-1-introduction-and-basic-search-methods/

What are some techniques to prune the search space when using backtracking algorithms? (2023). Linkedin.com. https://www.linkedin.com/advice/0/what-some-techniques-prune-search-space-when

Parallel Algorithm - Introduction. (2023). Tutorialspoint.com. https://www.tutorialspoint.com/parallel_algorithm/parallel_algorithm_introduction.htm

Search Algorithms in AI - Javatpoint. (2022). Www.javatpoint.com. https://www.javatpoint.com/search-algorithms-in-ai#:~:text=In%20Artificial%20Intelligence%2C%20Search%20techniques,agents%20and%20use%20atomic%20representation.

Kaput, M. (2022, March 7). AI in Search Engines: Everything You Need to Know. Marketingaiinstitute.com; Marketing AI Institute. https://www.marketingaiinstitute.com/blog/how-search-engines-use-artificial-intelligence

Aaron (Ari) Bornstein. (2019, May 7). AI Search Algorithms Every Data Scientist Should Know. Medium; Towards Data Science. https://towardsdatascience.com/ai-search-algorithms-every-data-scientist-should-know-ed0968a43a7a

Neural Search 101. (2023, May 11). Algolia. https://www.algolia.com/blog/ai/what-is-neural-search-and-how-does-it-work/

Rouse, M. (2017, January 4). Search Algorithm. Techopedia. https://www.techopedia.com/definition/21975/search-algorithm

Pavan Vadapalli. (2023, March 21). Top 12 Commerce Project Topics & Ideas in 2023 [For Freshers]. UpGrad Blog; upGrad Education. https://www.upgrad.com/blog/all-about-informed-search-in-artificial-intelligence/

CodeCoda LTD. (2021, October 22). Search algorithms and their implementations. CodeCoda.com. https://codecoda.com/en/blog/entry/search-algorithms-and-their-implementations

Singhal, A. (2001). Modern information retrieval: A brief overview. IEEE Data Eng. Bull., 24(4), 35-43.

AI. (2023). AI | Search Algorithms | Codecademy. Codecademy. https://www.codecademy.com/resources/docs/ai/search-algorithms

Chapter, & Blair, D. (n.d.). Language and Representation Information Retrieval and the Philosophy of language. Retrieved December 28, 2023, from https://deepblue.lib.umich.edu/bitstream/handle/2027.42/34570/1440370102_ftp.pdf?sequence=1

Shah, E. (2023, April 18). Search Algorithms in Artificial Intelligence - Scaler Topics. Scaler Topics; Scaler Topics. https://www.scaler.com/topics/artificial-intelligence-tutorial/search-algorithms-in-artificial-intelligence/

What are Neural Networks? | IBM. (2023). Ibm.com. https://www.ibm.com/topics/neural-networks#:~:text=Neural%20networks%2C%20also%20known%20as,neurons%20signal%20to%20one%20another.

Iyer, S. (2022, July 22). Neural Search: An In-Depth Guide. Aisera: Best Generative AI Platform for Enterprise. https://aisera.com/blog/neural-search-enterprise/#how-enterprise-search-work

11 Appendix

Hosted frontend link

https://white-moss-0f08d4f03.4.azurestaticapps.net/

Video Demo

https://youtu.be/I2vFYwXRLSM

Frontend GitHub repository link

https://github.com/TariqH19/studentServices

Backend GitHub repository link

https://github.com/TariqH19/strapiStudentServices

Miro board project management link

https://miro.com/app/board/uXjVPsmNOkI=/