



Pre – Purchase Car Inspection Mobile Application

Amy Rintoul

N00192537

Supervisor: Mohammed Cherbatji

Second Reader: Cyril

Year 4 2022-23

DL836 BSc (Hons) in Creative Computing

Abstract

The creation of a mobile application for pre-purchase automobile inspections was the goal of this project. A business that inspects second-hand vehicles before a consumer turns over their valuable money. Enabling the consumer to leave with confidence that they have chosen the ideal vehicle. With a thorough 100+ element check, this service allays any anxieties the consumer may have about the intimidating process of buying a vehicle.

Each examination looks at over 100 elements, including the engine, gearbox and chassis, as well as minor issues. One of the company's highly skilled technicians will provide the customer with a comprehensive report.

The developer conducted an interview with the business owner in order to create a fully functional mobile application that meets the demands of the companies target audience. The mobile application was developed for the users to interact with the mechanic with ease. Allowing the users to book appointments through the app, choosing a specific mechanic and a specific service and viewing any past or present reports that the user may have. The user may also see any upcoming appointment they may have.

The mechanic may use the application to confirm bookings, show their availability, create reports for a specific customer and see upcoming appointments.

Both the user and the mechanic can interact with each other through a messenger on the mobile application.

The developer carried out testing throughout and after the implementation of the application. This testing conducted of CRUD and navigation testing, User testing and testing the backend though Insomnia. The outcomes demonstrate that the programme navigates to satisfy the user's needs. Each test demonstrated that the user successfully travelled to each screen to which they wished to browse.

The application isn't fully functional, several of the functional requirements have not been met. With more time and knowledge of React Native the developer would have gotten a more successful finish to the application although the developer has created an extremely strong backend and frontend with few unfinished elements.

Acknowledgements

Firstly, I would like to thank my supervisor Mohammed Cherbatji for his ongoing mentorship throughout this whole process which wouldn't have been possible without his continues support and intelligence. His approach to frontend development and hard work is an inspiration, which I aspire to emulate throughout my career. My heartfelt gratitude also goes to Mohammed for his patience and understanding during the last 4 years of attending IADT, He is a fantastic tutor from whom I have learned so much.

I am also extremely grateful to my brother, Danny Rintoul who continued to give valuable and helpful recommendations during the planning stages of my thesis. Danny is a highly qualified mechanic, bouncing ideas off each other and constantly receiving his feedback became extremely beneficial and I am forever grateful to him.

Lastly, I would like to thank my family, friends and boyfriend for getting me through the last 4 years. Even though it seemed never ending, they persisted in pushing me all the way to the finish line.

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 leave copies of your own files on a hard disk where they can be accessed by others. 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

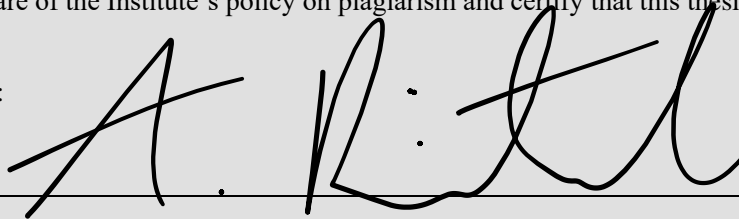
Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.

DECLARATION:

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

Student :

Signed

A handwritten signature in black ink, appearing to read 'A. R. L.', is written over a horizontal line. The signature is stylized with large, sweeping loops.

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

Table of Contents

1	Introduction.....	1
2	Research	2
2.1	Introduction.....	2
2.2	Mobile Application	2
2.3	Software Development	4
2.3.2	API Development:	4
2.3.3	Back-End Development:.....	4
2.3.4	Front-End Development	5
2.4	Software Methodologies (SDM's)	5
2.4.1	Agile Development Methodology.....	5
2.4.2	Waterfall	6
2.5	Software Development Approach.....	7
2.5.1	Native Development	7
2.5.2	Hybrid Development.....	8
2.6	Application Framework	10
2.6.1	React Native	10
2.6.2	Flutter.....	11
2.7	Conclusion	12
3	Requirements	13
3.1	Introduction.....	13
3.2	Requirements gathering.....	13
3.2.1	Similar applications	13
3.2.2	Interviews.....	15
3.3	Requirements modelling	15
3.3.1	Personas.....	15
3.3.2	Functional requirements	18
3.3.3	Non-functional requirements	18
3.3.4	Use Case Diagrams.....	19
3.4	Feasibility.....	21
3.5	Conclusion	21
4	Design	22

4.1	Introduction.....	22
4.2	Program Design	22
4.2.1	Technologies	22
4.2.2	Design Patterns	23
4.2.3	Application architecture	24
4.2.4	Database design.....	25
4.3	User interface design	26
4.3.1	Wireframe	26
4.3.2	User Flow Diagram.....	28
4.3.3	Style guide.....	29
4.4	Conclusion	34
5	Implementation	35
5.1	Introduction.....	35
5.2	Scrum Methodology.....	35
5.3	Development environment	36
5.4	Backend & Data Management	37
5.5	Setting up the frontend environment.....	49
5.6	Conclusion	53
6	Testing	54
6.1	Introduction.....	54
6.2	API Testing.....	54
6.2.1	GET request for all reports.....	55
6.2.2	GET request for reports associated to a specific user.	57
6.2.3	GET request for a report with a specific id.	58
6.2.4	POST request which creates a report.	60
6.2.5	PUT request which updates a specific report.	61
6.2.6	DELETE request made for a specific report	62
6.2.7	Bad request, errors	62
6.3	Functional Testing	64
6.3.1	Navigation	64
6.3.2	CRUD	67
6.3.3	Functional Testing Results	69
6.4	Conclusion	69

7	Project Management.....	70
7.1	Introduction.....	70
7.1.1	Testing.....	70
7.2	SCRUM Methodology.....	70
7.3	Project Management Tools.....	70
7.3.1	Microsoft Planner	70
7.3.2	GitHub	71
7.4	Conclusion	72
8	Conclusion	73
8.1.1	Your views on the project.....	73
8.1.2	Working with a supervisor	74
8.1.3	Technical skills.....	74
8.1.4	Further competencies and skills	75
	References	76

1 Introduction

Pre-purchase car inspections are conducted independently of car dealerships, with the primary purpose of ensuring that the client's future vehicle is safe, roadworthy, and not a waste of money. The overall aim for this project is to build a successful mobile application using MERN Stack. The client can create an account and login. The application is designed for clients to interact with mechanics through a messenger. Each client can choose which service they want and which mechanic they would like to view their vehicle.

This chapter concludes that the developer will develop a mobile application using the MERN Stack. The developer will use the Agile methodology and Native development for the application.

The features, functions, and activities that must be fulfilled for the application to be considered successful are covered in the requirements chapter. They provide the developer with a precise set of guidelines to operate within and help them identify the many objectives that need to be achieved.

The design chapter starts by providing a description of the project's application architecture and design, covering database and process design. The developer will provide information regarding the technologies that will be used to create the application and why they are appropriate for the process. This chapter starts to demonstrate how the requirements and the research come together before implementation.

The implementation chapter is a detailed description of the implementation through each sprint, from start to finish of the backend api and the frontend interface. Each sprint highlights the goal, learning outcome and any issues that occurred during this time.

The testing chapter is the process of evaluating the applications components with the intentions of finding whether it satisfies the specified requirements or not.

2 Research

2.1 Introduction

The aim of this literature review is to explore and determine what is software development, which is the best software development approach and what is the most suitable framework for building a mobile application. This review opens with a discussion of mobile applications and where we are now in terms of the rapid growth of this industry. Additionally, the developer will investigate many forms of software development and software methodologies, providing definitions for each along with its advantages and disadvantages discovered via extensive study.

This review will cover briefly about mobile applications before pointing out some significant statistics. We have truly grown reliant on our smartphones, and this development will only increase rapidly. In every aspect of our everyday lives, we utilise mobile apps. Doing any kind of fitness, we track our distance, heart rate, pace, mileage and the most important for many people our calories. I need to remember something, but there is an app for that. If you cannot find your TV remote, there is an app for that.

Choosing a software development approach can be the biggest dilemma when building an app. There are three commonly known development approaches Native, Hybrid, Web. Throughout this literature review, I will investigate Native and Hybrid approach and compile a report on which is the stronger path to take for developing mobile apps. This review aims to put forward an argument for using native rather than hybrid development.

What software development methodology (SDM) you are going to consider is one of the key factors to consider before creating a mobile app. A multitude of issues arise when a SDM is not implemented. It is an extremely crucial part of project management as it is a framework for structuring, planning, and controlling the development of an information system. This will decide whether a project succeeds or fails.

2.2 Mobile Application

A software programme created solely for use with a mobile device, such as a smartphone or tablet, is known as a mobile application. Applications for mobile devices are classified as to whether they are web-based, native, or hybrid. Throughout the review, these categories will be discussed in depth. Each programme offers specialised functionality such as a game, calculator, or mobile web browsing. (Mobile Application, 2020)

We have reached a point where mobile apps have come to make our lives effortless. Everyone who owns a phone or tablet is familiar with using mobile apps whether for playing a game, staying connected with family or friend or even tracking their fitness levels. According to Buildfire “88% of mobile time is spent on apps.”

The first smartphone was invented in 1992 and released by IBM in 1994, which came with built-in apps such as a phonebook, calendar, world clock, and calculator. (IBM, n.d)

Types of mobile operating systems which run specifically for mobile include Android, Apple IOS, Blackberry OS and Windows. Each operating system has unique App stores. Android –

Google Play Store, Apple – Apple app store, Windows – Microsoft Store, Blackberry – App World.

The apple app store opened on July 10, 2008, via an update to iTunes. Launched with second-generation iPhone 3G which supported mobile apps. According to Buildfire “In 2022 2.87 million apps are available for download on the Google Play Store and 1.96 million apps are available for download on the Apple App Store.” (Mobile App Download Statistics & Usage Statistics (2023) - BuildFire, 2021)

In terms of application sectors and features, there are various distinct types of mobile applications that a user may seek from the marketplace.

Lifestyle, Entertainment/social media, and Gaming

These applications cover a wide range of personal lifestyle and social activities, including dating, Fitness, Food and Music. Examples of these applications are Spotify, Tinder, Just Eat, and Nike Training Club. Every day, users use each of these programmes; some do so simultaneously, such as when using Spotify to listen to music and the Nike Training App while out on a run or in the gym.

The entertainment industry is vast and highly competitive. It has become near impossible for users to become bored as there are thousands of entertainments apps to choose from. Disney+, YouTube, Netflix, PrimeVideo

Social Media apps are the most popular worldwide. Through virtual networks, social media facilitates the exchange of ideas and information. There are 4.76 billion social media users globally, which is equivalent to 60% of the population worldwide. According to Statista, Facebook being the most used, as of January 2023 has 2.958 billion monthly active users. (Statista) Instagram, Twitter, TikTok, Snapchat are all examples of social media applications, and there are countless more.

In the app marketplaces for online gaming, as of 2023, there will be approximately 700,000 games, with 500,000 on Google Play and 200,000 on the Apple App Store. (Mobile Game Market Trends for 2023 You Need to Know - Udonis, 2023) According to Statistics users are anticipated to reach 2.32 billion in the mobile games industry by 2027. Among the most popular games are Candy Crush Saga, Wordscapes, 8 Ball Pool, and many others.

Utility

Most utility applications are pre-installed once a user purchases a mobile device or tablet. Utility applications include the weather, calculator, Flashlight and Reminders. Although they are not always as popular, without these applications we would be at a huge loss.

Productivity

Applications for productivity are primarily concerned with staying organised, monitoring job progress, composing documents, using your phone to make payments, sending emails,

booking hotels, and other activities. Productivity applications include Trello, Google Drive/Calendar, Google Docs, Word, Microsoft To Do, Google/Apple Pay.

2.3 Software Development

IBM have defined Software Development as “a set of computer science activities dedicated to the process of creating, designing, deploying, and supporting software.” (What Is Software Development? | IBM, n.d.). So all instructions needed by a computer to function correctly is known as software.

Software developers create software for end users all over the world by working across multiple platforms and using development languages. Their responsibilities include software research, design, implementation, and testing. Developers can create code, a database or frontend servers.

The following is a list of various software types.

2.3.1.1 Application software

Computer programming, which is the process of creating and maintaining source code, is referred to as application development. It includes any other processes that led to the finished application, such as research, new development, improvements, reuse, re-engineering and maintenance. (What Is Software Development? | IBM, n.d.)

2.3.1.2 Programming software

Assists the programmer in creating additional software. Software for programming is frequently referred to as a programming tool or a software development tool. Programming software includes compilers, assemblers, interpreters etc.

2.3.1.3 System software

A program created to control the hardware, software, and resources of a computer, including memory, CPU's and other hardware. There are multiple types of system software, such as Windows, Linux, MacOS X, etc. Each system software has distinct characteristics. (What Is System Software? | Simplilearn, 2022)

2.3.2 API Development:

Software can communicate with one another using an API (Application Programming Interface). A crucial element of software development is API Integration. They are the cause of the most recent clickbait being shared on social media or the instant opening of Google Maps after a search. (Brewster, n.d.). For anything and everything you can think of, there is an API. To make things simple, a mobile application that connects to the internet, sends data to the server and the server responds with the information requested.

2.3.3 Back-End Development:

The backend is the server side which is what goes on in the background of the application. Backend developers are mostly concerned with how a website functions. The backend is

required for the frontend to function. The frontend talks to the backend to retrieve the information shown to the user. It consists of an application, API, server, and a database. The application content is kept in databases, which are structured in a way that makes it simple to access, manage, modify, and save data.

Backend Languages consist of Python, JavaScript, PHP, SQL, Ruby, Java and many more.

2.3.4 Front-End Development

Front-end is based on what the user see's and how they interact with the finished product. The front-end consists of everything from a simple heading, paragraph, button, link and even a logo. An individual who specialises in designing and developing the user interface (UI) and user experience (UX) of a website, web application is referred to as a front-end developer. Front end developers are also responsible for ensuring that the website appears well on all platforms. For example, mobile devices, tablets, and computer displays.

Front-end development consists of HTML, CSS, and JavaScript. For a more sophisticated and responsive design, there are many alternative CSS Frameworks and libraries available. Utilising these frameworks enables quicker and more convenient web development because they come with ready-to-use stylesheets. These libraries include, among others, Tailwind CSS, Bulma, and Bootstrap. (Top 5 CSS Frameworks for Developers and Designers | BrowserStack, 2022)

JavaScript frameworks are quite crucial to programming since they provide your code structure. These frameworks serve as models and the basis for software programmes. It gathers and distributes shared resources for developers to utilise, including libraries, reference materials, photos, and more. (Toal, 2014). JavaScript libraries and frameworks Node. Js, Angular Js, React Js.

2.4 Software Methodologies (SDM's)

Each application developed requires meticulous project management. There is a vast majority of steps to approach in software development methodology. Planning, structure, and performance monitoring fall under this category. The goal is to simplify the process making SDM extremely crucial in this situation. SDM's concentrate more on the internal operations rather than the technical features. Numerous businesses now use SDM to enhance the quality of their product which obtains the greatest outcomes.

Frequently used methods of software development include Agile, DevOps, Waterfall and Rapid (RAD). Each SDM has unique characteristics, it is up to developers and their team to determine the most appropriate approach for their project.

2.4.1 Agile Development Methodology

When adding new functionality, teams use the agile development approach to reduce risk which include glitches, funding issues and changing requirements. Agile methodologies include the Crystal method, Scrum, Dynamics System Development among several others. In recent years, agile methodology has become one of the most popular software

development approaches. Agile places a high focus on communication, especially amongst engineers, clients, and users.

Scrum

One of the numerous varieties of agile technique is scrum, which is recognised for segmenting projects into sizeable units called "sprints." Each phase is comprised of two to four-week sprints, with the purpose of completing the most critical elements first and producing a deliverable product. Following sprints, further features are added to the product and are changed depending on the feedback from, in this case, supervisors and users. (What Is Agile Scrum Methodology? - businessnewsdaily.com, 2023). There are distinct roles within the scrum methodology which consist of Product Owner, Scrum expert and a Scrum Team.

Positives

The most significant advantage of agile scrum approach is its adaptability. The developer may easily and swiftly modify product goals throughout next sprints to deliver more beneficial iterations if there are any issues or adjustments. As the Scrum approach typically requires less record-keeping and control, it can be cost-effective for an organisation.

Everyone should take complete ownership of their job to foster a productive environment that produces products of the highest calibre.

To avoid working in isolation, integrated groups of designers, developers, testers, and business and functional specialists can collaborate as a team when using the scrum development technique. (7 Key Benefits of Using the Agile Scrum Methodology | Jile, 2023)

Negatives

Agile development methodologies focus on real-time communication, thus inexperienced users frequently lack the necessary background information to get started. (Team et al., 2017). They necessitate a considerable time commitment from users and are labour demanding since developers must finalise every functionality for user approval throughout each iteration.

Teams of at least three members, but no more than ten, often perform well when using the Scum approach. Even while this might encourage cooperation and teamwork, some businesses can find it challenging to organise their employees into teams. (Indeed, Editorial Team, June 25. 2022). When big teams are brought in, they frequently try to work in a more regimented way, which can either slow down the workflow or even block it completely.

2.4.2 Waterfall

The waterfall method is frequently regarded as the oldest approach to software development. The waterfall model is linear approach that makes use of a set of phases that do not overlap. Each phase emphasis certain objectives and must be fully completed before the next phase can begin. Developers will not be able to go to previous phases once

finished, thus the name waterfall. These stages are as follows: Requirements, Design, Implementation, Verification and Maintenance. (Team et al., 2017)

Positives:

The waterfall approach adheres to a certain format. It is a continuous framework with many stages, each containing its own collection of objectives and deliverables. Knowledge may be transferred amongst team members effortlessly since the project is easily transitioned from a particular stage to another.

For each stage of the project development process, the Waterfall methodology has clearly defined objectives and deliverables. Since each milestone must be met before moving on to the next, this kind of application, is simple to manage.

Negatives:

Since the Waterfall model involves a linear ordered design procedure, each phase must be finished before proceeding on to the next. The entire process is extremely established and planned, leaving little room for flexibility.

When creating an application, the Waterfall project management methodology fails to provide customer feedback top priority. This method has the drawback that the demands are subject to change, particularly when people begin working with your application and provide insightful feedback. (Singh, 2022)

Testing is not permitted under the Waterfall technique until the entire system testing phase, which is the last stage of the development process. (Singh, 2022)

2.5 Software Development Approach

There are currently three main types of smartphone apps available to us right now: Native, Hybrid and Web. One must decide which of these three main approaches is ideal for them when developing an app for a tablet or smartphone and how to execute it. As there are so many factors to consider, selecting one of these three options can be challenging and exhausting. Therefore, a suitable starting point for choosing between Native, Hybrid and Web is what features and functionality will your app have? What operating system does the target audience use? What industry does your proposal cover? What is the app's development budget? As each feature and functionality will behave differently whether on Android, IOS etc.

2.5.1 Native Development

Native Apps are specifically created for an IOS or Android platform. The platform that is chosen determines the set of tools and languages available for future app development. Swift and Objective -C are programming languages used to create native iOS apps while Kotlin and Java are used to create native Android apps. (What Software Development Approach to Pick: Native, Hybrid or Cross-Platform? - Touchlane, n.d.). Native Mobile App development is desirable when you want to provide the best user experience possible in

terms of how the product appears and feels. They can use fundamental smartphone hardware components like GPS, camera, microphone, proximity sensors and so forth. (Singh, 2022)

There are a high range of native apps that we use daily without even realising. Some being Google Maps, Spotify, Twitter, Pokémon Go, Pinterest and much more. (Singh, 2022)

Positives

There are several significant benefits to developing native apps. The fact that native apps function properly even when there is no internet connectivity is one of the main advantages. It is exceedingly rare you can use an app that functions offline, this is highly beneficial for any user, we cannot always rely on having a strong Internet connection or even any for that matter.

Another strong positive is the development speed. Native mobile apps do not utilize complex code like hybrid and cross-platform, which allows them to be significantly quicker. According to (What Software Development Approach to Pick: Native, Hybrid or Cross-Platform? - Touchlane, n.d.) native apps are a 25-30% faster option, in the grand scheme of things that is an extreme difference. A vast number of the app's components load quickly because they preload before they appear. As we use mobile apps for our everyday lives, we expect them to be rapid and ready when needed.

Because native apps are built for a specific platform, they provide a higher-quality user experience.

Native apps have a higher level of security than other methods because they use the operating systems built-in security features. (What Software Development Approach to Pick: Native, Hybrid or Cross-Platform? - Touchlane, n.d.)

Negatives

In comparison to the number of benefits native apps provide, there are not as many drawbacks as possible.

As I have already mentioned, native apps are exclusively made for one particular operating system; so, if you need to develop for both iOS and Android, you will have to do so individually. As you can expect, this would necessitate a considerable amount of money, and workforce. (Singh, 2023)

2.5.2 Hybrid Development

According to Net Solutions, (Singh, 2023) hybrid development is defined as “ the combination of native and web solutions where developers need to embed the code written with the languages such as CSS, HTML, and JavaScript into a native app with the help of plugins including Ionic’s Capacitor, Apache Cordova, and so on which enables to get the access of native functionalities.”

Uber is used to illustrate this definition. Uber is composed of a single codebase. Therefore, users can download the exact same Uber software from their individual app store regardless of the operating system on their phone.

Programming languages like HTML5, CSS, JavaScript are used to create hybrid apps.

Most consumers are unaware that their most used apps could potentially be hybrid. Here are a few examples; Instagram, Discord, Evernote, Gmail, and Microsoft Teams. These are only a few of the countless programs that have been created as hybrid applications.

Positives:

As mentioned previously native apps are solely based on a specific operating system unlike Hybrid apps which are composed of a single codebase. This time required for development will be lowered in half by eliminating the need to create two apps. Hybrid applications are the most efficient way to create a product and is rapid time to market.

A hybrid application is less expensive to develop because fewer developers are considered necessary.

Hybrid applications are easier to maintain because developers can update the code once and have it pushed to both operating systems.

Negatives:

For access to the app's features, user's must be connected to the internet, with hybrid there is no offline support.

A hybrid app must have approval for more than one platform lengthening the testing process.

There are several operating system-specific features that may not run properly since hybrid apps only use one piece of code. For instance, it is possible that some iOS capabilities will not work perfectly with Android, which will make things difficult for the user. (Singh, 2023)



Figure 1 - Native, Web & Hybrid Applications

2.6 Application Framework

A software library known as an application framework offers a core foundation to assist in the creation of applications for a particular environment. An application framework serves as the basic structure upon which an application is built. Since numerous companies now depend on having a mobile involvement to survive, the demand for these frameworks has increased significantly over the last several decades.

2.6.1 React Native

React.js, which is used to create web applications, is the foundation of React Native, a framework created exclusively for the creation of native mobile software. React was first made available to the public in 2013 and Facebook released React Native in 2015. You can build genuinely native apps using React Native while maintaining the quality of the user experience.

It is unaware of the total number of react engineers in the globe. But it is known that around 1.6 million individuals have the react devTools installed as a Google Chrome extension.

Thousands of applications employ React Native, but chances are you have previously used it in one of the many; Facebook, Outlook, WordPress and also Skype.

React Native allows you to expand and enhance the code base of an existing native application. This calls for the use of a native development IDE, in this context Android Studio.

In contrast to React, React Native features built-in components that render similarly rather than using the existing standard HTML element tags. The “View” component and the <div> tag both simply serve as containers for their content, which makes them quite comparable. Additionally, React Native uses a class named “Stylesheet” rather than separating CSS files to style elements.

When a user launches a programme, the device launches a total of three primary threads and, if necessary, further background threads.

UI Thread

The main native thread on which the programme will execute is called the Main Thread. It oversees interaction between users and displaying UI on a device's screen. Every natively created programme has an identical thread operating. (Kosmal, 2022)

Shadow Thread

This is the background thread. This thread will be launched concurrently with the JavaScript thread. Its goal is to calculate the locations of the views and build a layout structure written in the JS thread. React Native takes advantage of the Yoga layout engine, which translates flexbox-based structure into an architecture of layout that a native host can comprehend. (Kosmal, 2022)

JS Thread

The JavaScript thread is responsible for executing the React (JavaScript code) code.

Positives

The most significant advantage of React Native is the fact that programmers do not need to write distinct code for each platform (Android and iOS).

A web development technique called JavaScript is the foundation of React Native. React Native allows JavaScript programmers to construct mobile applications. By using this, developers may design apps faster and without having to master difficult languages like Java or Objective-C. (7 Advantages of Using React Native for Mobile App Development, n.d.)

React Native components correspond exactly to native development features. The app's native-like appearance is accomplished by merging JavaScript and native user interface elements into one.

The instantaneous reloading method in React Native enables developers to observe and perform modifications in real time. If you make any adjustments to the code, the app will automatically reload. You may also refresh sections of the update to save time while compiling.

Negatives

Because some unique modules are not present in the framework, developers may need to spend more time building and creating their very own additional modules from the ground up.

When faced with challenging UI design considerations, intricate animations, and intensive interactions, the React Native's responsiveness is pitiful.

Utilising a newer version of React Native is advised because each upgrade offers additional improvements. Nevertheless, you are unable to depend solely on the implementation of automatic updates; you might experience some unexpected problems as you gradually adjust to the latest version. (React Native Pros and Cons [2023 Update], n.d.)

2.6.2 Flutter

Altextsoft define flutter as an “open-source technology for creating mobile, desktop, and web apps with a single codebase. Unlike other popular solutions, Flutter is not a framework or library; it is a complete SDK” (The Good and the Bad of Flutter App Development, 2022)

Flutter was released in 2016 by Google. All apps created using Flutter are written in Dart, a programming language supported by Google. Dart can convert into native code along with JavaScript. Flutter did not require much time to become wildly successful after it is alpha debut in May 2017. The official Stable release in December 2018. (Wu, 2018)

Flutter contains a rendering engine, pre-made widgets, testing and integration API's etc as it is an entire SDK. Widgets are a whole other important aspect of Flutter. There is a vast variety of these pre-made widgets available, you may also modify and build your own. (The Good and the Bad of Flutter App Development, 2022)

Positives

The framework provides a selection of unique and attractive widgets. Additionally, they upgrade the libraries regularly with different interface elements. Because of the Flutter functionality, you just need one Flutter development group to create and maintain your apps.

The online community for developing apps with Flutter has been expanding steadily. Professionals and novices alike are eager to contribute their skills and expertise because of the welcoming development community. (Vietnam, 2020)

Negatives

Since they make certain functionalities available to developers, third-party libraries and packages have a considerable influence on software development. Nonetheless, finding such free packages and libraries is difficult because Flutter is a new platform for creating mobile apps.

The enormous file size of Flutter apps is a major flaw that may not be overlooked.

Dart is the programming language used with Flutter. Despite it being a trustworthy programme due to it being rapid, it is still unable to maintain a competitive edge among the other programming languages. (Montaño, 2023)

2.7 Conclusion

The developer discusses two software methodologies, Agile and Waterfall. Although the waterfall methodology brings many positives to project management, it leaves truly little room for flexibility as each phase must be completed before moving onto the next. It also fails to provide feedback from the customers which gives no room for change although the application may not be up to the customers standards.

For these reasons, the developer chose to implement the Agile, Scrum methodology for the development of the mobile application. The main factors that stood out to the developer where how easily and swiftly the products goals could be modified. This is extremely beneficial as it allows less room for issues or later adjustments. Each phase is divided into 2 – 4-week sprints, allowing the developer to keep evaluating their workflow and communication with their supervisor throughout the entire process.

Throughout this review, the developer examined two different software development approaches, Native and Hybrid. There are a vast number of benefits to developing native apps. One major key feature that stood out significantly well, native apps function offline without needing to connect to the Internet. This is highly beneficial to user's all around the

world. Native applications are extremely fast and preload before the information is displayed. Rapid apps on the go, is an immensely powerful tool to have.

React Native is a freely accessible framework with a quick setup that plenty of web developers find extremely useful. It is a powerful framework that has amazing performance results on iOS and Android. Giving the rising need of mobile apps it is crucial to choose the right application framework for your design. Although I do not feel there is a particularly right or wrong answer here, I feel like React Native has stolen the show over Flutter. I do believe it depends on the mobile app you are building and the circumstances, there is a lot to take into consideration as I have already mentioned. React Native enables developers to be more efficient by eliminating the need to create duplicate code for several platforms.

Overall, React Native is a win, win situation for app development since it saves money, time, and effort. Overall, it results in a phenomenally successful mobile app development which is the chosen framework for the developer's application.

3 Requirements

3.1 Introduction

The goal of the requirements phase is to provide developers with a chance to determine what the program should be able to achieve. As opposed to the developer defining what is necessary, it is crucial to learn what the users would like the program to do.

To weigh the benefits and drawbacks of pre-purchase car inspection applications, the developer examined related programs. This enables the user to understand what does and does not function.

The developer conducted interviews to determine the key features for the user and the mechanics of the app. This will be extremely beneficial as it may highlight numerous difficulties that emerge during the several interviews.

The developer designed several personas which helped assist in realizing that various people have unique needs and expectations, as well as assist in identifying the users you are defining for. It also helped support the developer in achieving the objective of potentially giving the target audience a positive user experience.

The project's needs, including their functionality and performance levels, will also be highlighted, and covered in the requirements chapter. The developer will conduct a feasibility analysis to assess the projects viability, outlining the suggested application's advantages and drawbacks.

3.2 Requirements gathering

3.2.1 Similar applications

3.2.1.1 Car Experts

Car Experts is a website that allows users to schedule appointments for pre-purchase vehicle checks. Car Experts Ireland is a company that is not affiliated with automakers. Their major objective is to make sure that the vehicle you buy next is safe. A snippet of the Car Experts webpage is featured below in Figure 2.

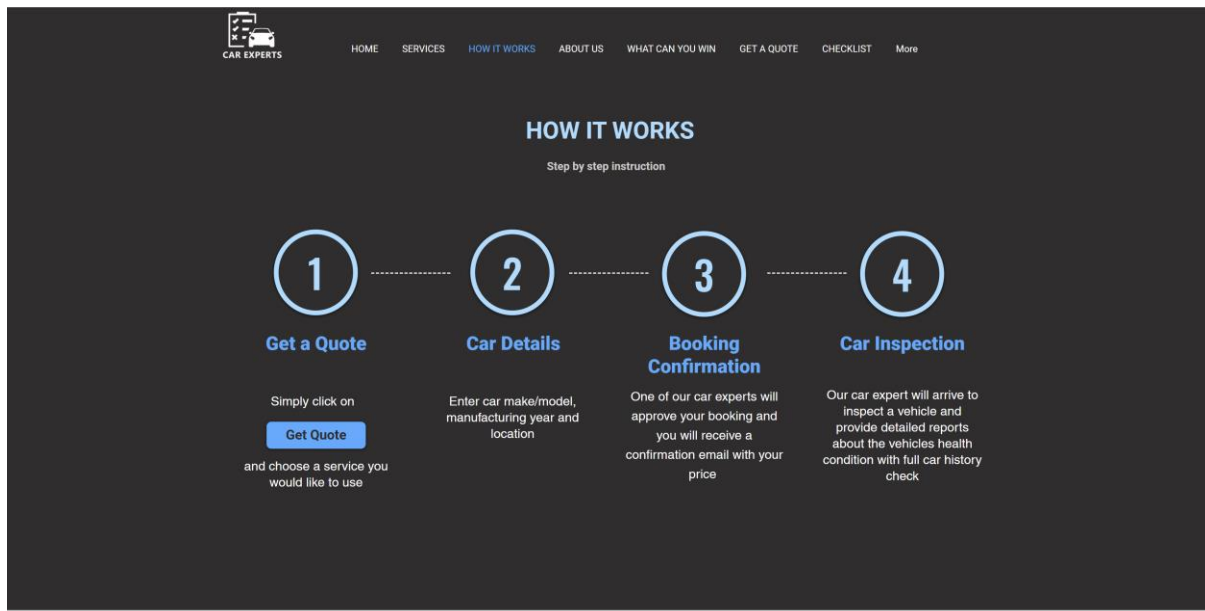


Figure 2 - Car Experts Website

Advantages:

Car Experts offers mobile capability which is extremely beneficial for the user. It is an overly simplistic webpage and straight to the point.

Disadvantages:

Car Experts although offers a mobile version does not offer a mobile application. It is extremely limited as to what you can do on their site, being a single webpage. The only choice is to fill out a form to seek a call-back, and that is to obtain a quotation. You are unable to request an online payment or schedule an appointment with available dates shown. Going back and forth can generate a lot of unnecessary effort for the user.

3.2.1.2 Car Inspections Ireland

Car Inspections Ireland founder and lead engineer Noel Maher, who began working in the auto and insurance sectors in the 1980's. To provide knowledge for making buying decisions and to serve as a protection and guide for customers in the auto industry, Noel launched Car Inspections Ireland.

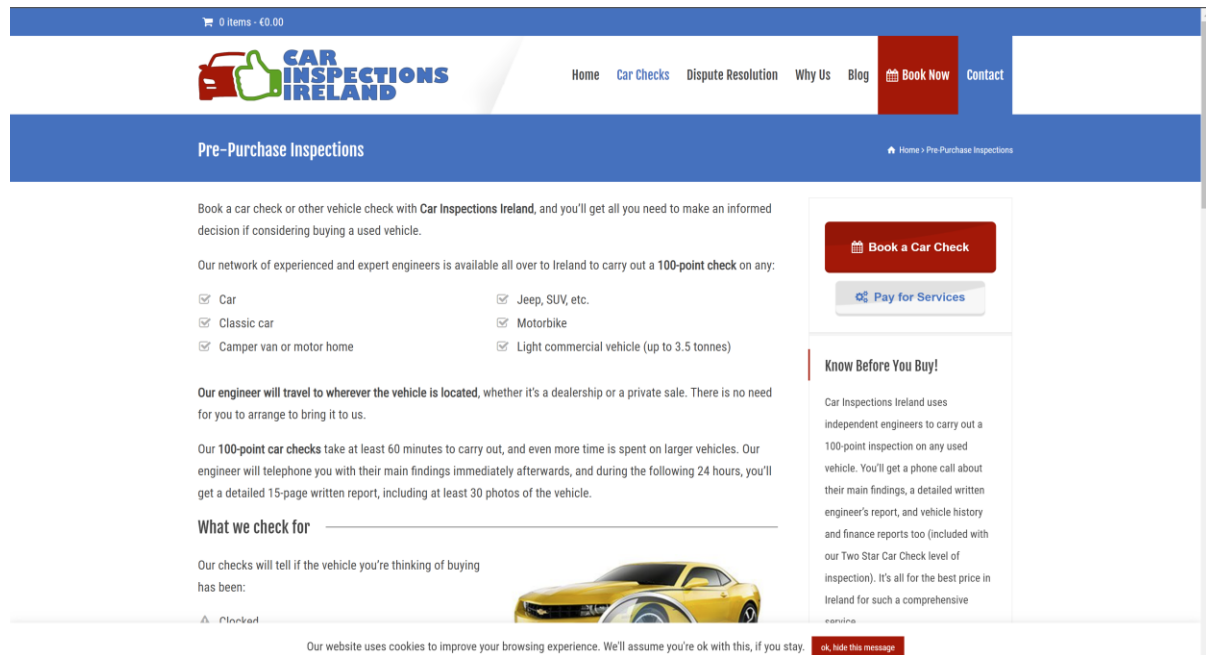


Figure 3 - Car Inspection Ireland Website

Advantages:

Offers post-repair Inspections following a crash with various levels of payment. Car Inspections Ireland has an online appointment booking with the option to pay for the service afterwards using a reference number. Their website is highly informative in terms of what they provide. Blog entries with the latest information for that area are also published on their website.

Disadvantages:

The website feels extremely chaotic and is a little overloaded making it hard to concentrate on the key points. Despite being quite informational, the websites layout makes it difficult to navigate.

3.2.2 Interviews

3.2.2.1 Interview with a technician from the company.

Conduct interviews with 3 or 4 users to find out what the key features for them for the app are. There may be various issues that arise in multiple interviews. These can be grouped together into several themes.

3.3 Requirements modelling

3.3.1 Personas

Personas are imaginary characters that aid the developer in understanding the requirements of the user. They also assist in determining who the relevant users are. To develop outstanding products, developers must have a thorough grasp of their target market. One of their most crucial questions they ask is “Who are we developing for?”

Typically, the following details should be included when constructing a user person.

- ✚ Persona name
- ✚ Photo
- ✚ Demographics
- ✚ Goals & Needs
- ✚ Frustrations
- ✚ Quote or Slogan

3.3.1.1 Persona 1 – Sarah Browne

Student and part-time waitress Sarah Browne is searching for a new, dependable vehicle that is also environmentally friendly. She is in desperate need of assistance since she understands little about vehicles and does not want to waste money on a car that is untrustworthy. She finds the process of car shopping to be quite intimidating.

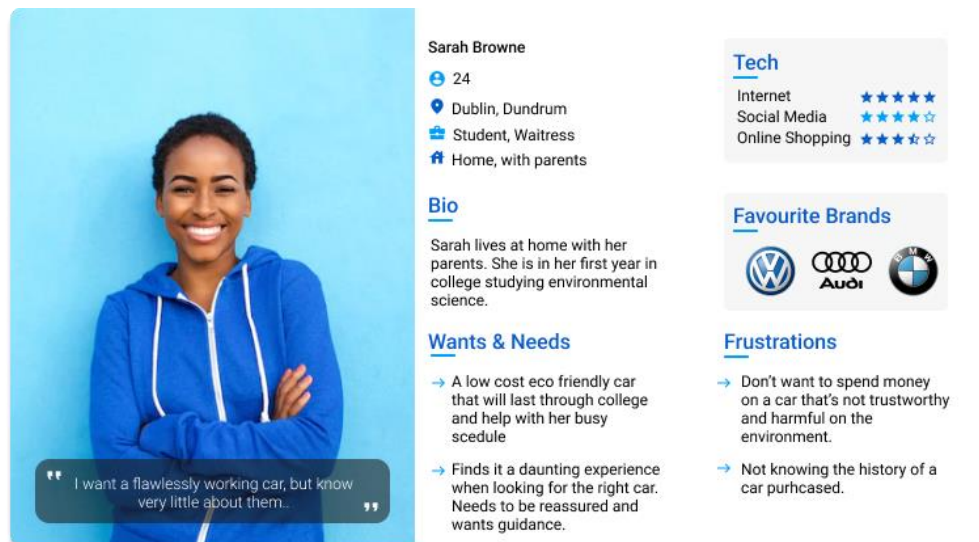


Figure 4 - Persona Sarah Browne

3.3.1.2 Persona 2 – James O' Leary

Banker James O' Leary is now retired and lives at home with his wife. He is looking for a conventional, vintage car. He is not very technically gifted and would like something a little bit more straightforward. He has been taken advantage of throughout the years and has purchased expensive vehicles without researching their histories. He must be assured before purchasing, especially because he is seeking for something so vintage.

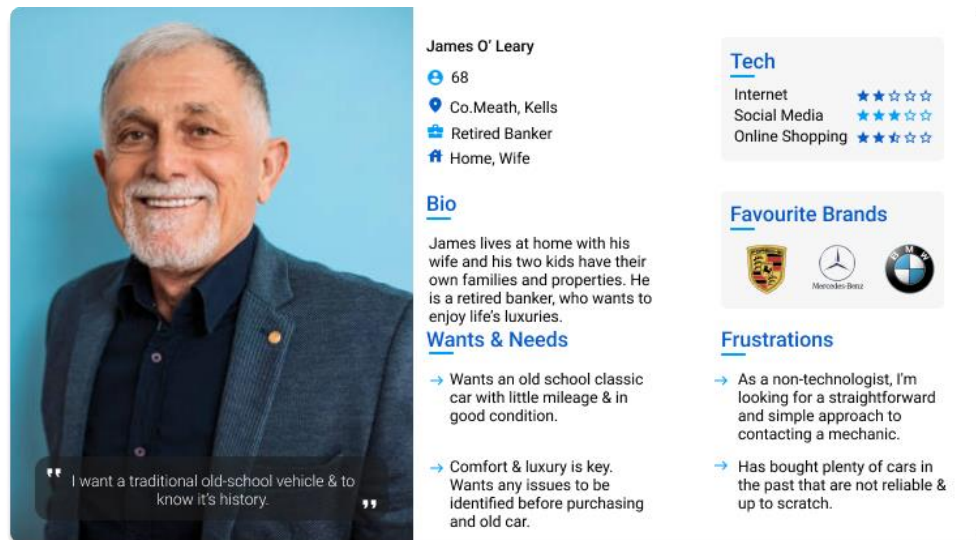


Figure 5 - Persona James O'Leary

3.3.1.3 Persona 3 – Danny Thomas

Danny Thomas, a licensed mechanic with expertise in Hyundai who is currently employed by Intel as an engineer, wants to take the next step and launch his own company to help provide for his family. Danny requires a simple method for communicating with clients and indicating his availability. He wants to help clients buy cars with confidence. When buying vehicles, he finds it frustrating when sellers conceal prior damage with shoddy repairs and absence of service records, to name just a few of the many issues.

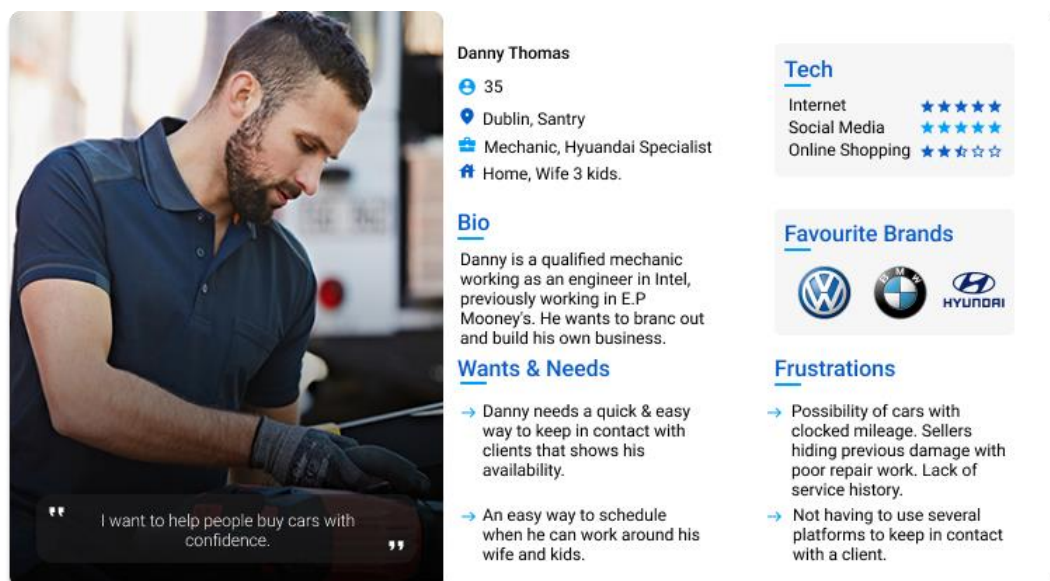


Figure 6 - Persona Danny Thomas

3.3.2 Functional requirements

The intended operations of a program or system are referred to as functional requirements. Functional requirements often explain how a system will behave under circumstances.

3.3.2.1 User:

ID	REQUIREMENTS
1.0	Register
1.1	Login
2.0	Choose A Specific Mechanic
2.1	Choose A Plan
2.2	Book An Appointment
2.3	Pay for the service they chose
3.0	Chat with the mechanic
4.0	See upcoming appointments
5.0	See past & present reports

3.3.2.2 Mechanic

ID	REQUIREMENTS
1.0	Register
1.1	Login
2.0	See upcoming appointments
3.0	Chat with the user
4.0	Confirm Booking
4.1	Upload documents for a specific user
4.2	Upload images for specific user

3.3.3 Non-functional requirements

Non-functional requirements are a criteria that, if not satisfied, do not prevent the application from functioning, but do indicate that the application is not performing as well as it should. They are often concentrating on concerns such as:

-  Usability
-  Performance
-  Security
-  Availability

ID	REQUIREMENTS	RESPONSE
1.0	Availability	The application should be compatible with both Android & iOS
2.0	Security	The application should only show user-generated material that has been contributed by that user and any material published by the mechanic, and only registered users should be allowed to log in.
3.0	Performance	The programme must function well, with loading times of no more than two seconds for each screen.
4.0	Usability	The application should be easy to use for the customer and mechanic. The essential actions should be completed quickly for the users.

3.3.4 Use Case Diagrams

A use case diagram's objective is to show the many ways in which a user may engage with a system. An effective diagram should list the actions and variations used to accomplish the objective. To help demonstrate and describe the applications complete systems needs and the important components, the developer created two use case diagrams using draw.io. The two illustrations show how a user and a mechanic interact with the suggested application.

3.3.4.1 User:

This use case diagram highlights the interactions between the user and the application. The user must register if they have not already before they may log in. If you are already logged in, you can proceed directly to this section. The user, once logged in can Book an appointment, chat with the mechanic, view past and present reports, view what services the application provides and view information about each individual mechanic. The diagram also shows the process in booking an appointment.

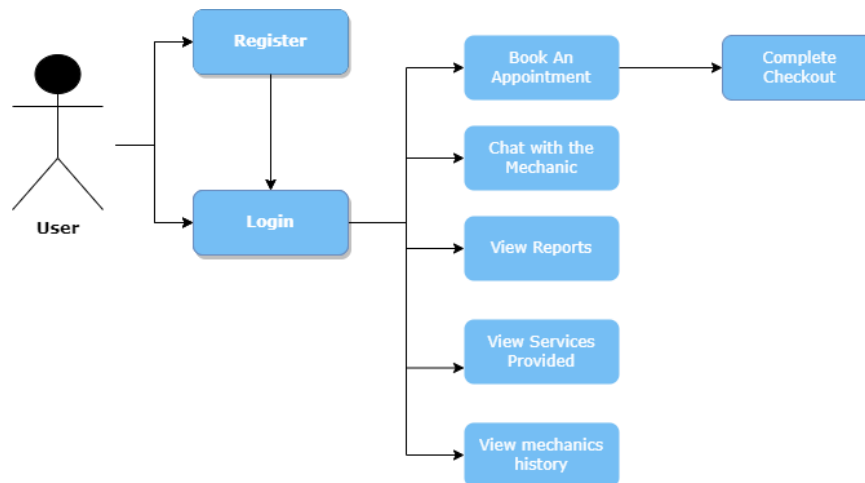


Figure 7 - User use case diagram

3.3.4.2 Mechanic

The interactions between the mechanic and the application are depicted in the use case diagram as shown below in Figure 8. Once the mechanic is logged in, they can confirm incoming bookings from the user. They can chat with the user. Once an inspection is complete the mechanic will upload the final report and can also upload images.

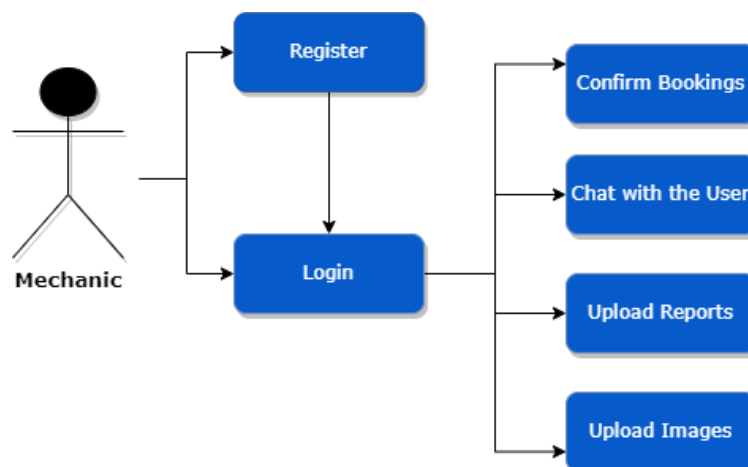


Figure 8 - Mechanic use case diagram

3.4 Feasibility

The technologies that will be utilized to create the application are covered in this section. It then explains if there are any issues in terms of the technical feasibility of the project, for example, if there are two diverse types of software which may have compatibility issues.

The developer has some prior experiences with Express.js and Node.js which would be helpful for the application's development.

The developer previously used React.js and has a good knowledge of this technology from a previous project. Nevertheless, developing a final-year application with limited experience using React Native might be difficult. A solution to this problem is to study the technologies being utilized by following free online tutorials and consulting the online documentation on the React Native website.

The developer will be using visual studio code, which is a free source code editor made by Microsoft.

3.5 Conclusion

In this chapter, the developer has discussed the suggested application. The application is a pre-purchase car inspection mobile app. This mechanic has the equipment and expertise to inspect and perform diagnostics on the next vehicle the user is about to purchase. Allowing the user to have peace of mind when buying a used car. The user and mechanic can chat to each other through an instant messenger within the app. The user can book an appointment, pay for the service they would like, see past and present reports from the mechanics and chose which mechanic they would like for the job.

Three different applications where analysed that had numerous similarities to the suggested application. The first application the developer investigated was Car Experts, this is an overly simplistic webpage which lacks major functionality. The form where you may request an online quotation and wait for a response is the only key component that the user can interact with. For the user who is waiting for a response, this slows down the entire procedure. The website itself has a fantastic aesthetic appeal and a smooth flow and gets right to the point. Another comparable program that the developer examined is called Car Inspections Ireland. Compared to the previous example, it has a lot more functionality. It also offers users the opportunity to have post-repair inspections following a crash with various levels of payments. The website's developer noticed that it is excessively cluttered and challenging to read. Although the two samples are mobile friendly, the developer was unable to locate a mobile app that was comparable to the proposed application.

The developer designed three personas, were created by the developer, and were based on a mechanic, an elderly user, and a youthful user. One is a college student, while the other is retired. These were established to impact design concepts and aid in decision making. Two use case diagrams were designed to see the users and mechanics interactions with the application. This aids in determining the systems major characteristics.

4 Design

4.1 Introduction

This chapter will go through the design of the application. In the life cycle of developing systems, the design phase is crucial. The goal of the project's design phase is to provide developers the opportunity to create an application design that satisfies the requirements outlined in the Requirements chapter.

Typically, the design phase is often broken down into two categories

1. Program design
2. User Interface design.

Firstly, the developer will discuss the technologies been chosen and why they were suitable for the development of this application.





The developer will go through the application's user interface design in this chapter. The developer will discuss two user flow charts that will show the users and mechanics processes that they will go through while using the program. The developer will outline the style system utilised to construct the application. Wireframes that will be designed using Figma, will be exhibited in this section.

For the database design the developer created an entity relationship diagram this is essential for modelling the data stored in the database. This provides a visual starting point for database architecture and is also used to help identify the needs of the applications information systems.

4.2 Program Design

4.2.1 Technologies

The application will be created by the developer utilizing the MERN stack development architecture which is a more modern stack.

-  MongoDB (Database)
-  Express (webserver framework)
-  React Native (front-end library)
-  Node (server-side environment)

These technologies were chosen because the developer has some understanding of MERN stack being used in a previous project.

Node.js is an open-source and cross-platform JavaScript runtime environment. This is a platform for developing web applications, application servers. It is often used to develop

API's. In this case the developer will be creating an API for the backend using Node.js. Node.js is not limited to web applications but is also for mobile applications which is a crucial part of the development of this application.

Expo is a framework for React Native which provides you with tools that help you develop code and deploy code. Allows the developer to ignore Java and Swift. Everything can be written in plain JavaScript and will look fantastic on both iOS and Android. The developer does not need to be concerned about any native components, such as Android or iOS code.

Stripe is a payment processing platform which have APIs to accept credit cards and payments. This is extremely beneficial to the developer as the application has a payment system. The creation of native apps for the Apple and Android device platforms is facilitated by mobile device helper libraries, commonly referred to as mobile device SDKs. The developer may include Stripe into React Native-built iOS and Android applications with the aid of the React Native SDK.

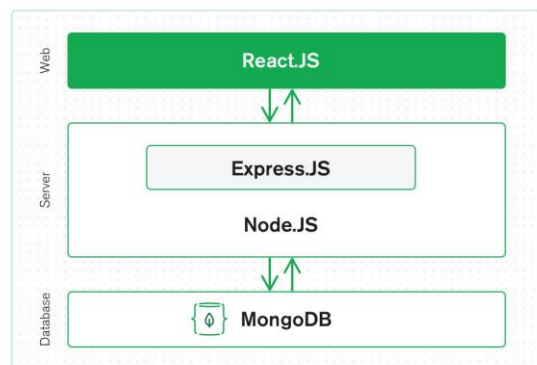


Figure 9 - MERN Stack Technologies

4.2.2 Design Patterns

The application consists of a MVC, Model View Controller. The core concept of MVC is that each component of your code serves a distinct purpose. The app's data is stored in a portion of the developers code, which also contributes to its attractive appearance alongside a portion of its functionality. As indicated within the name, MVC had three core components, Mode, View, and Controller.

All the user's data-related logic is represented by the Model component. It processes requests from the controller and is independent of the view component. The model oversees dealing with data, regardless of if it comes from a database, an API, or a JSON object.

The view is the user-interface, typically an interactive graphic user interface (GUI). Model information, interactive interfaces, and data are displayed in View. For instance, the client view will have access to all UI elements, including buttons, dropdowns, and text fields.

Data collection, modification, and delivery to the user are all duties of the controller. The controller connects the view and model. The Controller serves as the primary "brain" of this system and decides what it will perform next after getting "requests" from the View. (The Model View Controller Pattern – MVC Architecture and Frameworks Explained, 2021).

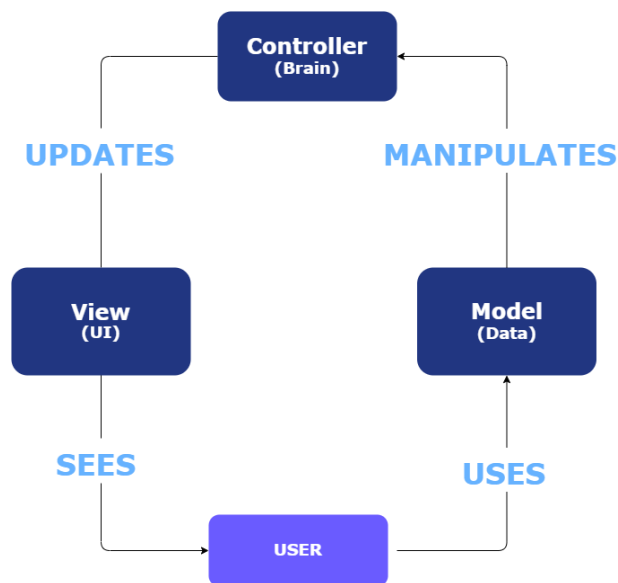


Figure 10 - Model View Controller

4.2.3 Application architecture

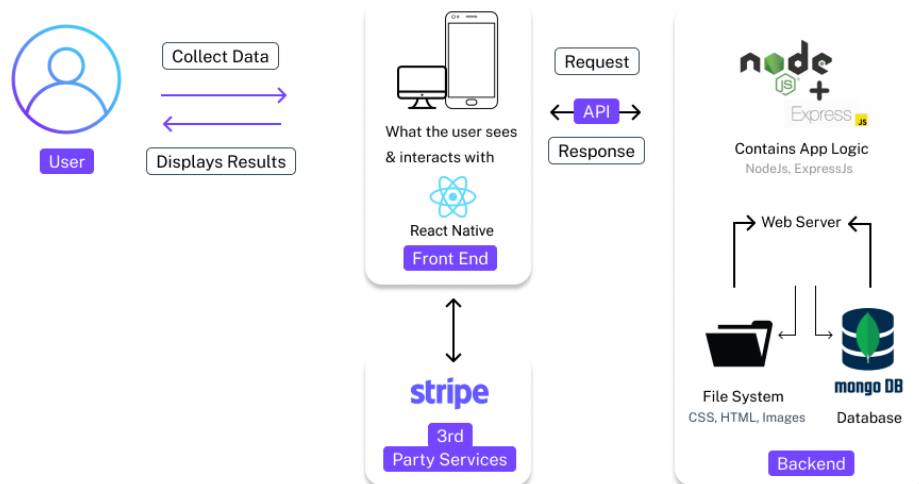


Figure 11 - Application Architecture

Web architecture consist of a client, a sever with the application logic and a database for the storage and processing. The key objective is to provide a clear logic for the application and the relationships between the various technologies.

User views data from the front end, frontend sends a request from the user to the backend and the backend sends back a response to the frontend and is displayed to the user.

The server-side language varies, the developer will be using JavaScript (Node.js). While the database holds the data, the application logic directs the program to do the necessary tasks and responds to user requests.

Stripe notifies the front end that the transaction was successful, and the front end notifies the back end that the transaction was accepted.

4.2.4 Database design

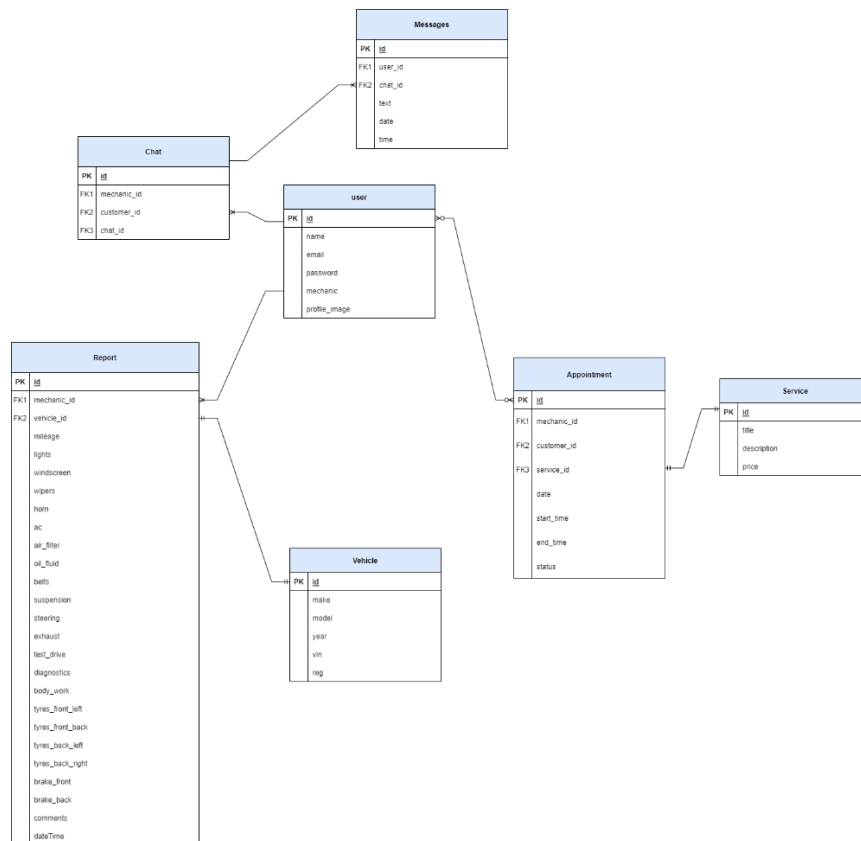


Figure 12 - ERD

4.3 User interface design

This section will describe how the interface is designed. It will include the user interface design elements, wireframes, user flow diagrams and style guides.

4.3.1 Wireframe

Wireframing is essential in UI design. A wireframe shows the content and functionality for the layout, information architecture, user flow and intended behaviours. Wireframing is a critical part of the interaction design process as the developer can use wireframes to test if their conceptual ideas transition effectively in practice.

Each wireframe designed by the developer was created using Figma, which is a web application tool for interface design. The developer chose a grid system to align page elements based on the sequenced columns. To arrange text, graphics, and functionalities consistently across the design, the developer adopted a column-based layout. The grid system is illustrated below in Figure 13 as an example. Within the wireframes shown below the developer has include many different elements which include search fields, headers, buttons, some text etc. The wireframes are created in a greyscale, using lighter shades of grey to represent lighter colours and darker shades to represent darker colours.



Figure 13 -login, homepage, appointment screen

The wireframes below show three separate pages, Figure 14 - Messenger, shows the user's active chat rooms in a list format. A search box allows users to look for a specific chat they require from the past. The user has the option to start a new chat message and return to the homepage via the bottom nav bar.

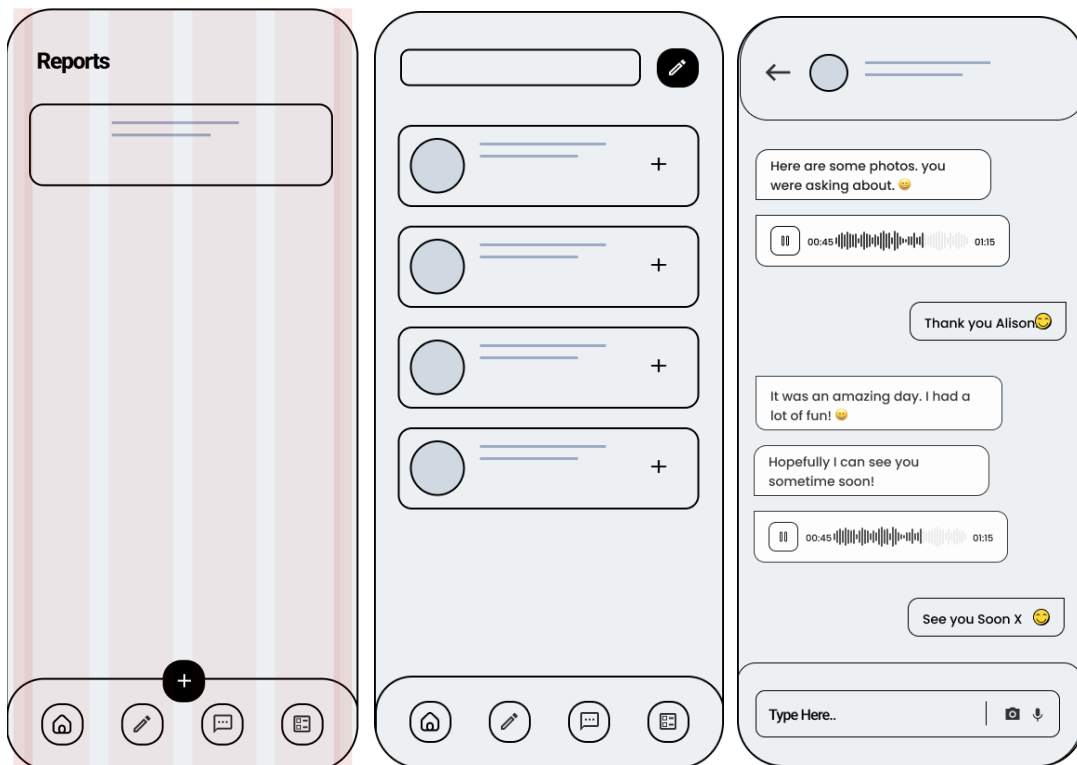


Figure 14 – Reports, Messenger, Chat Screen

4.3.2 User Flow Diagram

A users flow chart depicts the tasks involved in the process as well as the users' flow. A user flow diagram is usually created after you have thought about the customers experience and needs. It is crucial to map out and visualise the experience you want users to have, to completely comprehend it. Without a diagram, the developer would be forced to maintain an internal mental representation of the workflow, which might result in poor design and technological choices.

The diagram below in Figure 15 shows how the user flows through the application. If the user is already signed up, they can input their email and password otherwise they must set up an account and fill in their details before proceeding. Once logged in, the user is sent to the homepage, where they may explore a variety of features.

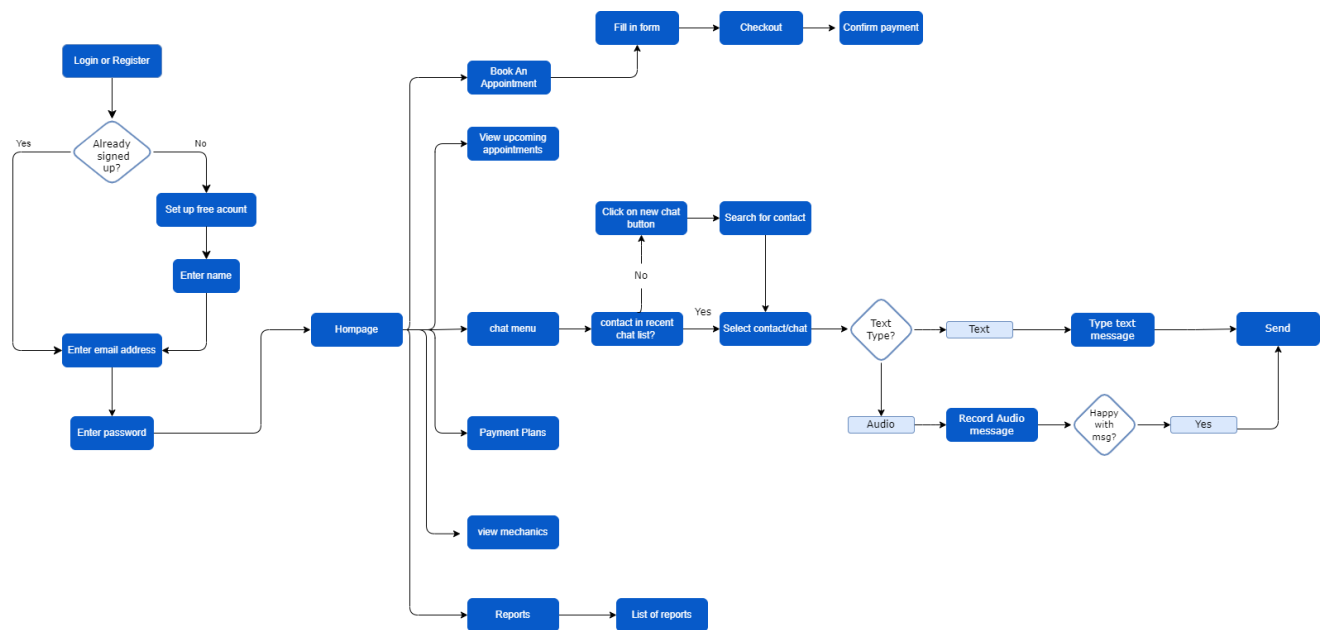


Figure 15 - User flow diagram

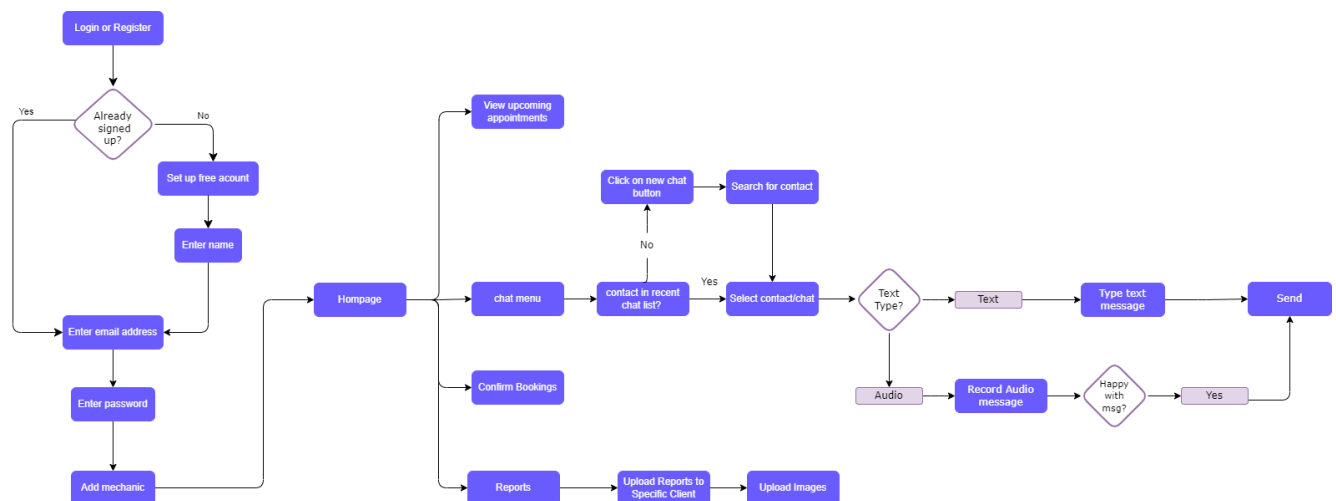


Figure 16 - Mechanic flow diagram

4.3.3 Style guide

Style guides show the colours, typography, and layout of the application. There is a general theme that is used throughout all the pages in the app. The developer chose an analogous colour scheme to create a sense of harmony for the user. The colours are close to each other on the colour wheel so are uniform and calm. This colour scheme can be low contrast so white is also used as an accent to create contrast and text is kept to black or white throughout.

The Hex colour #0F0445 (Federal Blue) is a very deep blue in RGB consisting of 27% blue. This is the dominant colour chosen where gradients are applied and throughout the application. Navy blue is a cool colour seen as authoritative and conveying a sense of trust which is why it is often used in uniforms. It has also been used in marketing for its depth and to show professionalism and sophistication. It is a reliable and stable colour which is non gender specific.

Purple is used as a supporting colour. In colour psychology purple denotes luxury and mysticism adding an air of interest to the app. It is a powerful and energetic colour. (Braam, 2022). Purple was chosen by the creator because colour has a relaxing impact on the mind and body. It is frequently viewed as an uplifting and inspirational colour.

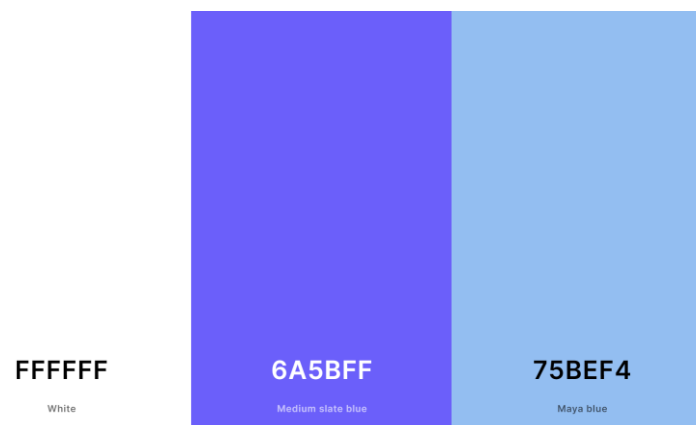


Figure 17- Colour Palette 1.0

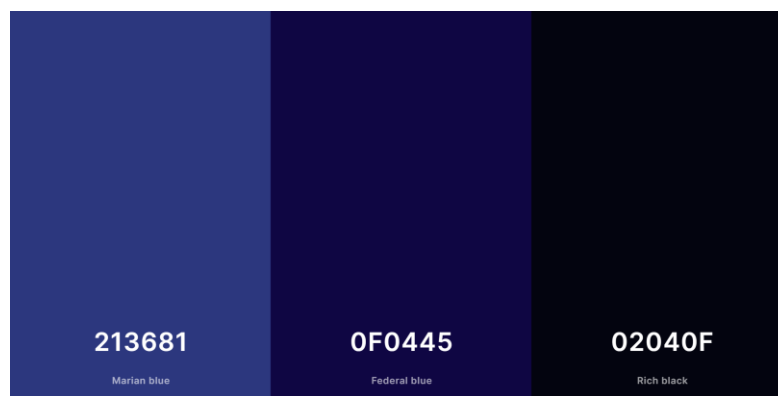


Figure 18 - Colour Palette 2.0

When developing any application, whether mobile or web, a solid layout in UI Design is essential. The developer built a visual system that looked coherent and helped the user navigate by using components of consistency with established relationships between them.

The developer applied the flexbox layout module to ensure stability throughout the application. Flexbox utilises columns and rows along a horizontal and perpendicular axes. It is an extremely useful tool that guarantees exact alignment. There are two axes of flexbox, the main axes and the cross axes. According to Basic Concepts of Flexbox the cross axis is perpendicular to the main axis. (Basic Concepts of Flexbox - CSS: Cascading Style

Sheets | MDN, n.d.) To accomplish the desired layout, the developer employed a variety of distinct flexbox properties. These properties include, flex, flex-direction, justify-content and align-items.

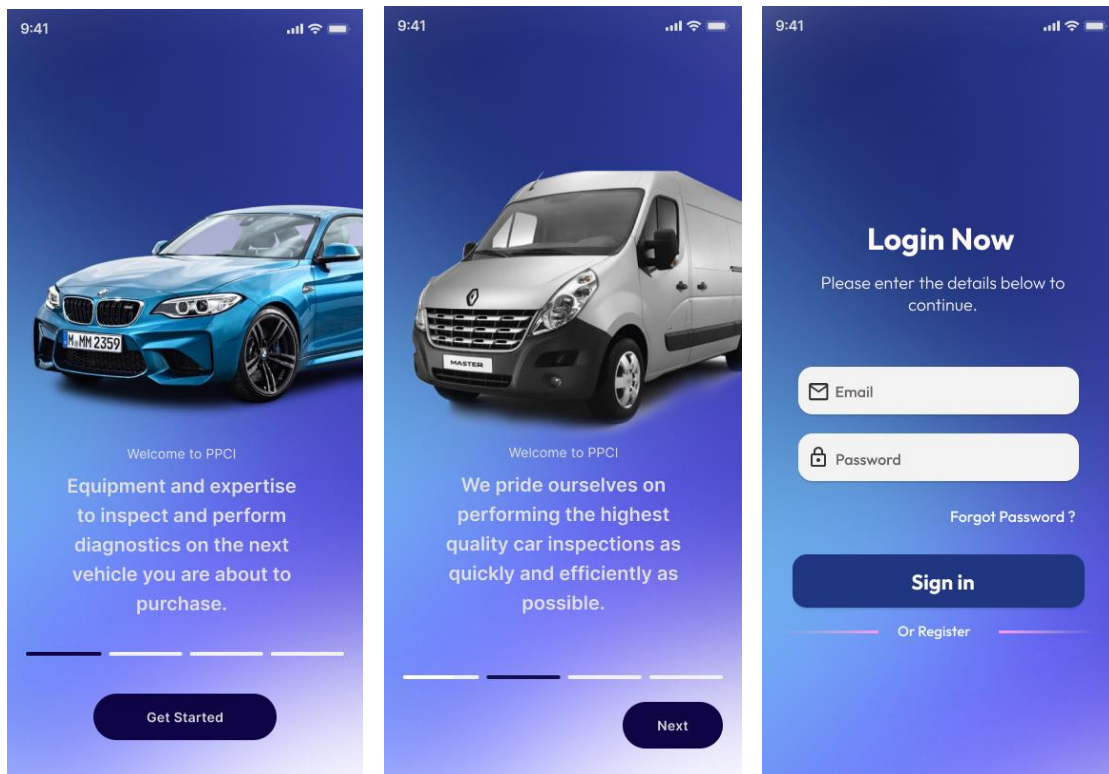


Figure 19 - Welcome Screen, Get Started, Login Screen UI Design

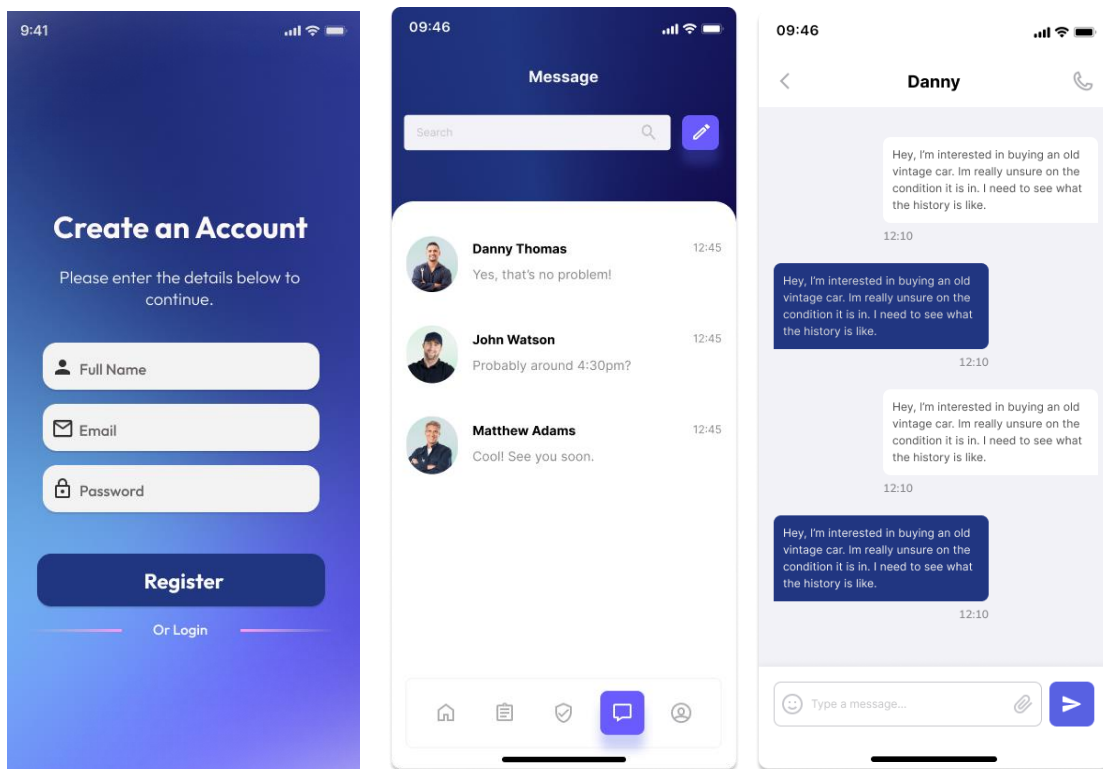


Figure 20 - Register Account, Messenger, Chat Screen UI Design

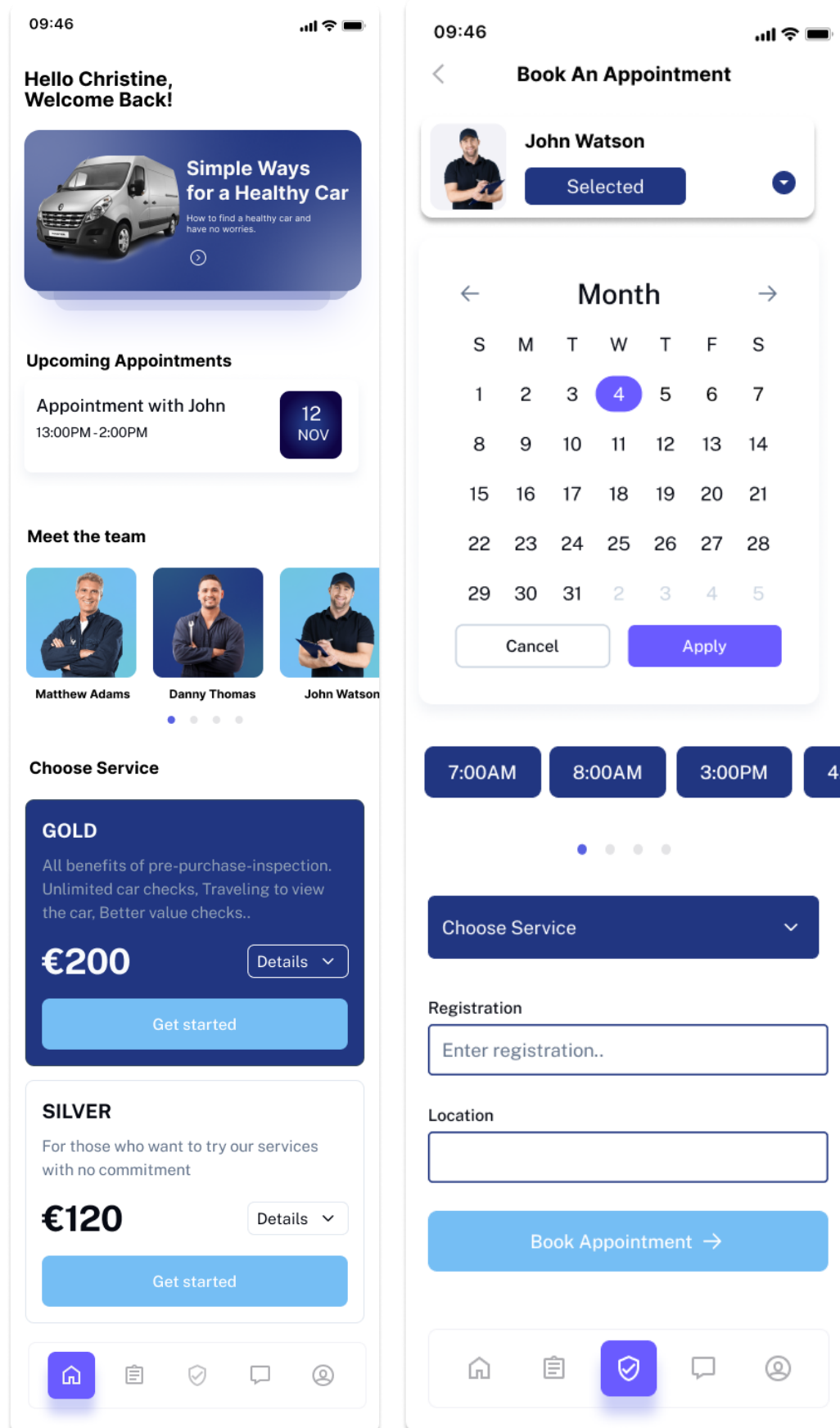


Figure 21 - Home Screen, Book Appointment Screen

Figure 22 and 23, show the logo designed by the developer. Sometimes, a company's logo is the most recognisable aspect of a project or business. A brand's logo is its emblem, and although it does not occupy very much room, it has a significant impact. The designer chose to design a remarkably simple but effect logo, keeping with the colour scheme can be eye-catching and memorable.



Figure 22 - Marian Blue Logo



Figure 23 - Medium Slate Blue Logo

4.4 Conclusion

In this chapter the developer discussed the technologies and the key reason they chose them and the benefits of using the MERN Stack model. One of the main reasons the user chose these technologies is to learn more about developing mobile applications.

The developer also examined the structure of MERN Stack and the structure of Expo and React Native.

Another section the developer discussed in this chapter was the user interface design. The developer went into detail with several high-fidelity wireframes which were designed using Figma. The wireframes are made up of different components, headers, text, calendar, buttons etc. The wireframes aided the developer in thinking about and communicating the structure of the application being constructed. Nailing a strong interface structure is the most crucial step in application design. By creating these wireframes in this phase, time and painful adjustments will be avoided later.

By this stage, the user's goals and where they are coming from have been explored in the requirements section, and the developer is focused on creating a good user flow diagram. The user flow diagrams gave the developer an overview of the process of the application. It interprets how the user and mechanic will interact with the application.

The style guide shows the design system for the application. The style guides show the colours, typography, and the layout of the application. The chapter explains why these were chosen and how they suit the application. There is a general theme that is used throughout all the pages of the application. The developer chose to use repetition within the design, reusing colours, patterns, fonts, and images. Repetition brings rhythm and unity to the application.

5 Implementation

5.1 Introduction

In this chapter the developer will describe the implementation for the application. The developer will also discuss the development environment being used for this project and explain the various stages of each sprint, the goal, the issues, and then knowledge gained.

The application has been developed using the following technologies:

MERN stack – Mongo Db, ExpressJs, React Native and NodeJS. Expo is a free and open-source framework for developing universal native apps for Android, iOS, and the web. Another technology being utilised is Stripe, an online payment infrastructure. The technologies have been explored in greater depth in the preceding design chapter.

The developer has been tasked with creating and designing a mobile application for an up-and-coming business, Pre-Purchase Car Inspection (PPCI). The technicians have the equipment and expertise to inspect and perform diagnostics on the next vehicle a customer is about to purchase. Each inspection examines over 100 parts from the engine, gearbox and chassis to smaller items such as light bulbs etc. The technicians will take care of examining and inspecting the vehicle for you, giving you peace of mind while purchasing a vehicle.

A user can book an appointment with one of the company's highly qualified technicians, interact with them through instant messaging, view present and previous reports, and choose what service they would like & pay for the service through the application.

A technician can interact with a user through instant messaging, confirm bookings, upload images and reports for specific clients. Additionally, they can view upcoming appointments.

5.2 Scrum Methodology

According to Digite, Scrum is an agile development methodology which is used in the development of software. (*What Is Scrum Methodology? & Scrum Project Management*, 2022).

When adding new functionality, teams use the agile development approach to reduce risk which include glitches, funding issues and changing requirements. Agile methodologies include the Crystal method, Scrum, Dynamics System Development among several others.

Scrum is executed in temporary blocks that are short and periodic, called Sprints. These usually range from two to four weeks. Each sprint is an entity, producing a full outcome.

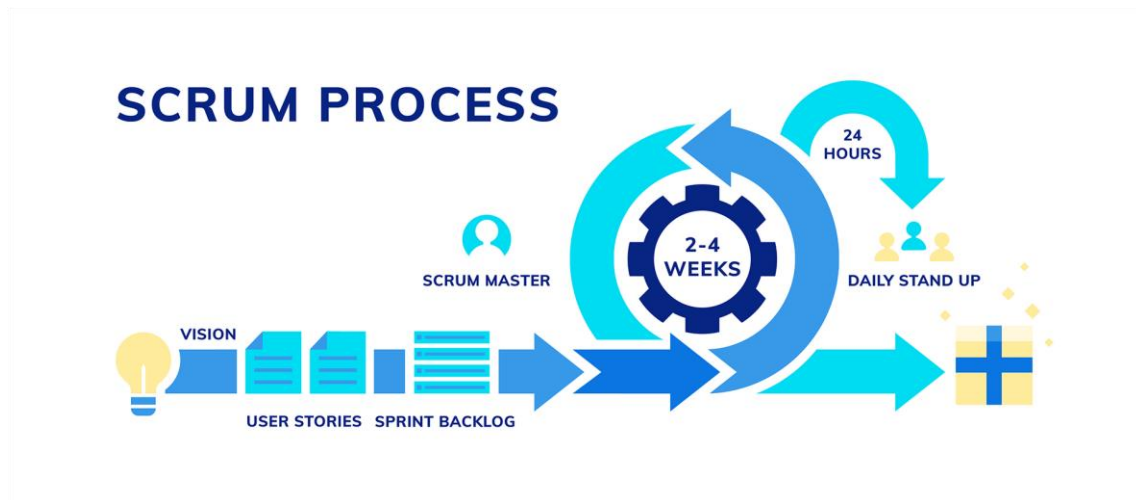


Figure 24 - Scrum Methodology Process

The implementation phase for the development of this application consists of 9 sprints overall, each sprint being 2 weeks long.

The requirements for the application were listed in a product backlog. Each item on the product backlog was broken down into a series of tasks which formed a sprint.

The Roles

5.3 Development environment

An integrated development environment (IDE) is a piece of software that aids in the productive creation of software code by programmers. By integrating functions like software editing, building, testing, and packaging in a user-friendly program, it improves developer productivity.

The developer for this project will be using Visual Studio Code which is a free and open-source IDE. Vs code has features like syntax highlighting, bracket matching, auto-indentation, box selection, snippets etc. Debugging is a crucial part of the programming process. Vs Code provides interactive debugging, which allows you to move through the source code, examine each variable, and run commands in the console. You may also add third-party extensions using Vs Code. This enables you to personalize each aspect and make it unique.

Vs Code also has Git commands built in. A Git Repository window, which displays all the information in your repository, including local and remote branches and commit history, is also included. Deploying with confidence and ease.

The frontend application's developer set up a GitHub repository. The project started with a GitHub repository, and each time a developer made modifications, those changes were committed and published to the repository.

5.4 Backend & Data Management

5.4.1.1 Creating a Database

The database is on MongoDB servers so it will be hosted. According to MongoDB is an open-source document database used to build available and scalable internet applications. (*Why Use MongoDB and When to Use It?*, n.d.). MongoDB allowed the developer to immediately start building the application without spending time configuring a database.

Firstly, In the projects tab, the developer created a new project called MajorProject_backend. Once the developer created the project, they could then build a database using the free shared version as seen below in Figure 25.

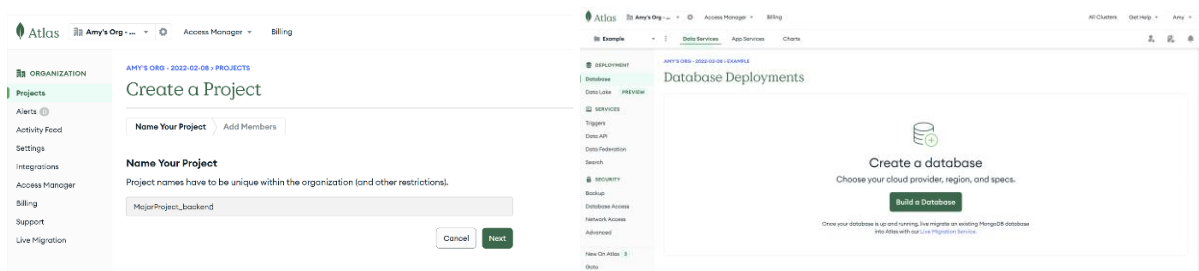


Figure 25 - Creating a project on MongoDB & building a database

The developer went through each step to create a free cluster. The developer created a username and password that gave permissions to read and write any data for the project. The developer added the current IP Address and 0.0.0.0/0 which allows all IP Addresses to access the project's clusters.

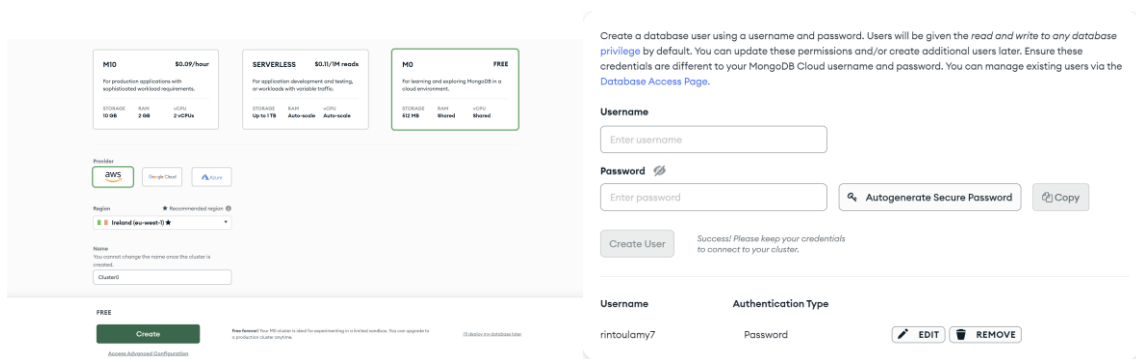


Figure 26 - Creating a cluster & adding authentication

Once the cluster was created, the developer was able to choose an option in collections to add their own data. As seen below in Figure 27, the developer filled in the appropriate input

fields. The collection name is the actual table within the database, so for example the database needed a reports table to store all the data regarding reports.

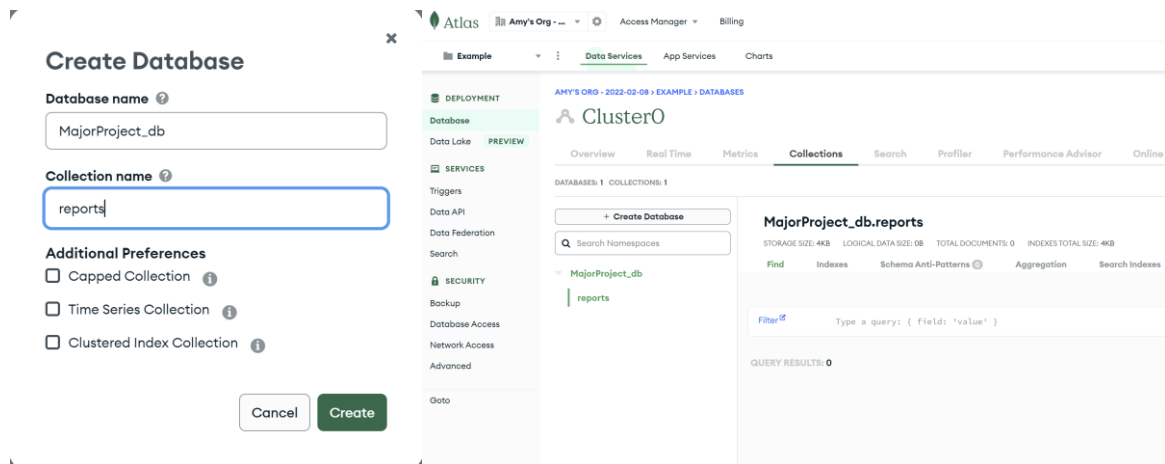


Figure 27 - Creating a database & adding a collection

In the database tab, there is a connect button which gave the developer options to connect to MongoDB. The developer chose the option 'Connect to your application' which gives a link that allowed the developer to connect to the database. Figure 28 below illustrates the link.

```
DB_ATLAS_URL = 'mongodb+srv://arintoul:secret1@cluster0.tjlhanp.mongodb.net/MajorProject_db?retryWrites=true&w=majority'
```

Figure 28 - Link connecting to the database

5.4.1.2 Backend Setup

The developer created a folder that contains the backend for the application. The developer ran the following node package manager commands. In Figure 29 below displays the commands the developer used to create the backend application. Once the folder was created, the developer created a package.json file for their application by using the command `npm init`. This command will prompt the developer for several things, name, and version etc. The developer then installed Express in the projects directory and saved it in the dependencies list. Express is a lightweight and adaptable Node.js web application framework that offers a comprehensive range of functionality for online and mobile apps.

```

rinto@DESKTOP-ESL7158 MINGW64 ~/Downloads/IADT/Year 4
mkdir MajorProject_backend

rinto@DESKTOP-ESL7158 MINGW64 ~/Downloads/IADT/Year 4
cd MajorProject_backend

rinto@DESKTOP-ESL7158 MINGW64 ~/Downloads/IADT/Year 4/MajorProject_backend
npm init
this utility will walk you through creating a package.json file.
it only covers the most common items, and tries to guess sensible defaults.

See npm help init for definitive documentation on these fields
and exactly what they do.

Use npm install <pkg> afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (majorproject_backend)
version: (1.0.0)
description: Major Project Backend
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
error: license should be a valid SPDX license expression (without "licenseRef"),
"UNLICENSED", or "SEE LICENSE IN <filename>".
license: (ISC)
output to write to C:\Users\rinto\Downloads\IADT\Year 4\MajorProject_backend\pack
age.json:

{
  "name": "majorproject_backend",
  "version": "1.0.0",
  "description": "Major Project Backend",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)

rinto@DESKTOP-ESL7158 MINGW64 ~/Downloads/IADT/Year 4/MajorProject_backend
npm install express

added 57 packages, and audited 58 packages in 1s

packages are looking for funding
  run npm fund for details

found 0 vulnerabilities

rinto@DESKTOP-ESL7158 MINGW64 ~/Downloads/IADT/Year 4/MajorProject_backend
code .

```

Figure 29 - Backend project setup

Additionally, the developer installed dotenv, mongoose, and cors. Any cors-related issues are handled by cors on the backend server. Mongoose is the data model which is a MongoDB driver. Using Mongoose, the developer builds models that interact with MongoDB. Our database is MongoDB, however the driver that links to the rest of the application is Mongoose. Dotenv is a way to store certain code in an environmental variable. For instance, the URL provided by MongoDB includes a username and password. Since not everyone should have access to the password, a .env file environmental variable may be used to conceal it.

```

rinto@DESKTOP-ESL7158 MINGW64 ~/Downloads/IADT/Year 4/Major Project/MajorProject_Backend
$ npm install express cors mongoose dotenv

added 162 packages, and audited 163 packages in 8s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

rinto@DESKTOP-ESL7158 MINGW64 ~/Downloads/IADT/Year 4/Major Project/MajorProject_Backend

```

Figure 30 - Installing mongoose, cors & dotenv packages

The developer installed nodemon as a dev dependency, programme that automatically updates the server anytime a change is made, eliminating the need to restart the server. The developer also modified the start script to nodemon server.js, which contains all the application's routes.

```

"scripts": {
  "start": "nodemon server.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
"author": "Amy Rintoul",
"license": "ISC",
"dependencies": { ...
},
"devDependencies": {
  "nodemon": "^2.0.20"
}
}

```

Figure 31 - Nodemon installation

5.4.1.3 Connecting to the Database

Firstly, the developer created a server.js file and a db.js file. The database connection code is contained in the db.js file. The developer imported mongoose and created a method named connect that runs when the developer wants to initialise the database, this is shown in Figure 32 below. The developer also exported the connect method. The code in the connect function establishes a connection to the environment variable file's DB_ATLAS_URL. The developer then provided two incredibly popular parameters, which ensure that the database contains the correct data objects. Once the database has established a connection, it will respond with the message "Successfully Connected to db."

```

tils > dbjs > ...
1  const mongoose = require('mongoose');
2  mongoose.set('strictQuery', false);
3
4  //runs when wants to initialise the database
5  const connect = async () => {
6    let db = null;
7
8    //Connecting to the database
9    try {
10     //connect to the URL within the .env file
11     await mongoose.connect(process.env.DB_ATLAS_URL, {
12       useNewUrlParser: true,
13       //makes sure the database has the correct data objects.
14       useUnifiedTopology: true
15     });
16
17     //when connected to database logs 'Connected Succesfully to db'
18     console.log("Connected Succesfully to db");
19     db = mongoose.connection;
20
21   }
22   catch(error) {
23     console.log(error);
24   }
25   finally{
26     if(db !== null && db.readyState === 1) {
27       //await db.close();
28       //console.log("Disconnected successfully from db");
29     }
30   }
31 }
32
33 };
34 //exporting the connect method
35 module.exports = connect;

```

Figure 32 - Connecting to the database, db.js file

A terminal window with a dark background and light-colored text. The text shows the output of a nodemon command, indicating that the application is watching for file changes and has successfully started, listening on port 3000 and connecting to a database.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  COMMENTS

[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Example app listening at http://localhost:3000
Connected Successfully to db
```

Figure 33 - Connected successfully to the database.

5.4.1.4 Folder Structure

For good project management, the developer created separate folders for the models, routes, and controller to give structure the project. This makes the code easier to understand and manage, especially as the project grows. The folder structure for the project can be found in figure 34 below.

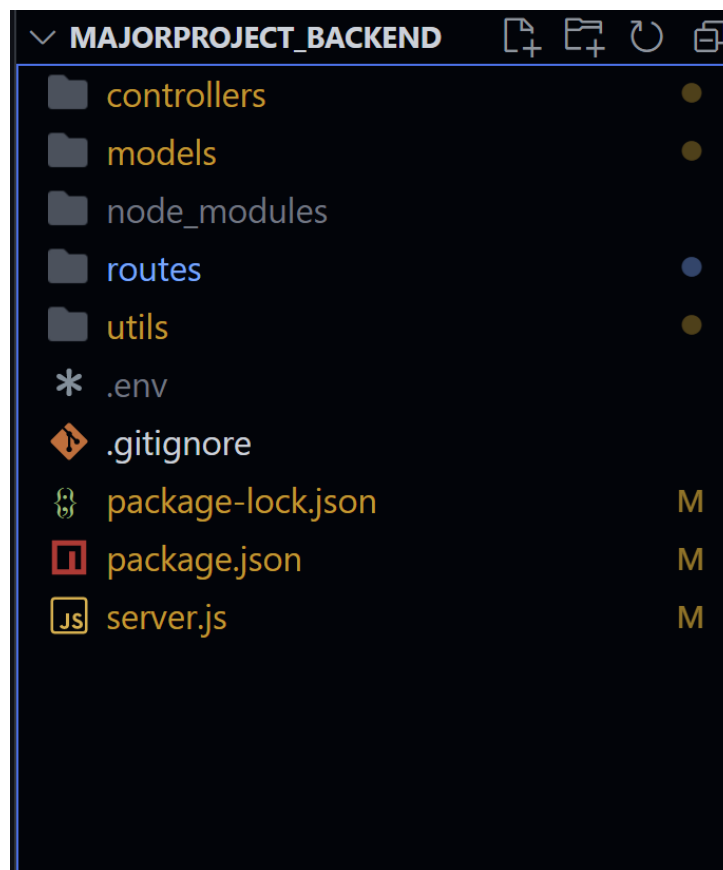


Figure 34 - Backend Folder Structure

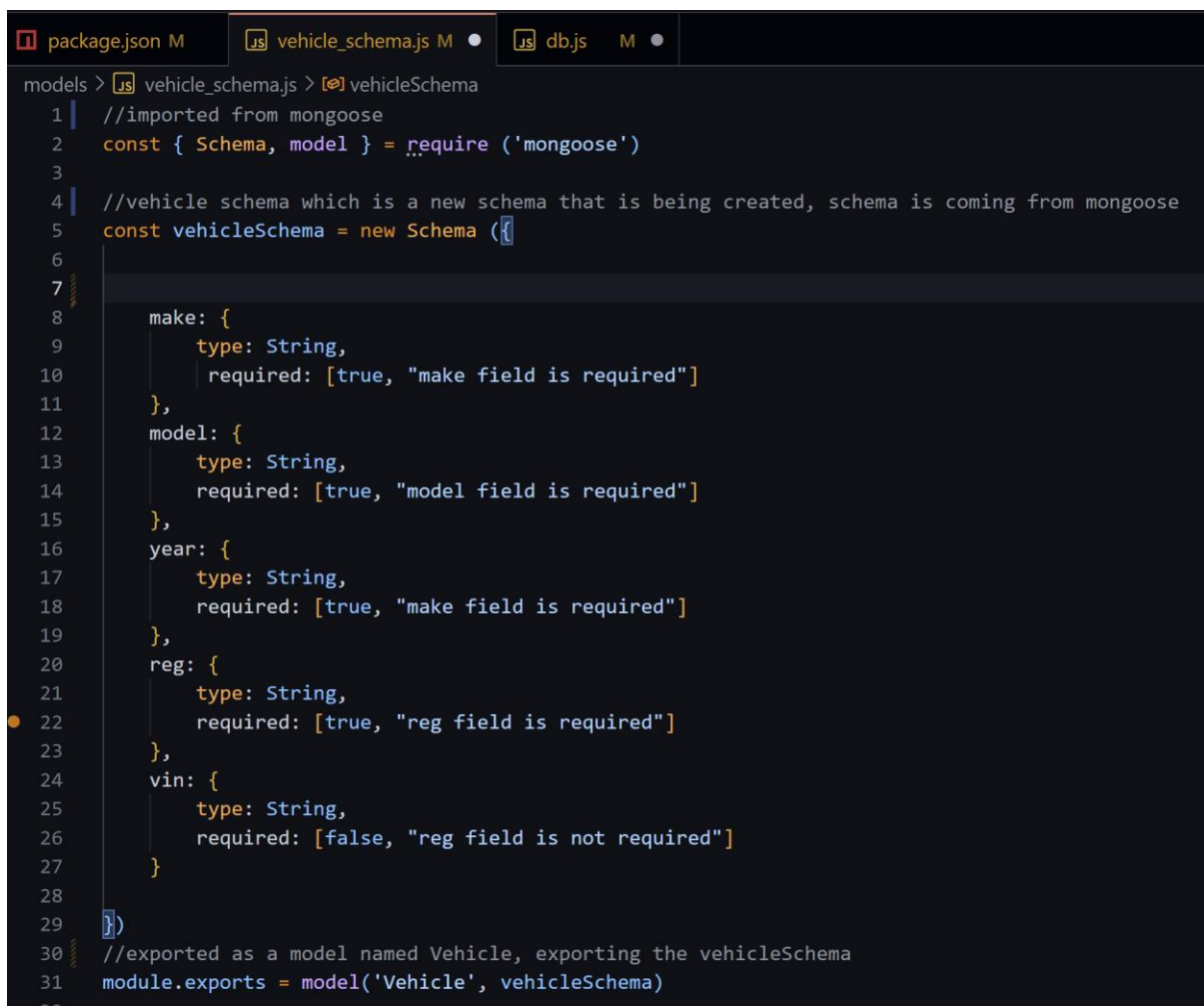
5.4.1.5 Setting up models, routes and controllers

Firstly, within the controller folder the developer created a `vehicle_controller.js` file. This file contains all the code that permits the routes to make POST, GET, PUT, and DELETE requests to the controller as well as the actions that are to be taken in response to those requests.

Within the model's folder, the developer created a `vehicle_schema.js` file which defines all the properties for that vehicle and casts it is associated SchemaType, for example title, description etc, and whether it be a String, Date, ObjectId, etc.

Vehicle Schema - Model

Starting with the vehicle schema which is the model. All the code within this file is mongoose and connects to the mongoose MongoDB driver. The developer imported Schema and model from mongoose and created `vehicleSchema` which is a new Schema. The developer exports it as a model giving it the model's name `Vehicle` which is a string, and it exports the `vehicleSchema`. Exporting the model allows the developer to refer to the model as just `Vehicle`. The developer adds properties to the schema that correspond to a SchemaType. For instance, the vehicle schema in figure 35 includes the make, model, year, and registration, all of which have the data type String and indicate whether they are required or not.



```

package.json M  vehicle_schema.js M  db.js M
models > vehicle_schema.js > vehicleSchema
1 //imported from mongoose
2 const { Schema, model } = require('mongoose')
3
4 //vehicle schema which is a new schema that is being created, schema is coming from mongoose
5 const vehicleSchema = new Schema({
6
7
8   make: {
9     type: String,
10    required: [true, "make field is required"]
11  },
12  model: {
13    type: String,
14    required: [true, "model field is required"]
15  },
16  year: {
17    type: String,
18    required: [true, "make field is required"]
19  },
20  reg: {
21    type: String,
22    required: [true, "reg field is required"]
23  },
24  vin: {
25    type: String,
26    required: [false, "reg field is not required"]
27  }
28
29 })
30 //exported as a model named Vehicle, exporting the vehicleSchema
31 module.exports = model('Vehicle', vehicleSchema)
32

```

Figure 35 - Vehicle Schema

Vehicle Controller

The next phase of the application, the developer implemented the vehicle controller file. Firstly, the developer imported the vehicle schema. The first method is `getAllVehicles`, in this

method if the user tries to get all vehicles it will get all the vehicles from the database and return them as a JSON. If everything is in order, the developer receives the data that is the vehicle in the .then block after verifying that the data exists. If an error occurs, respond with a 500-error code and a JSON message stating that none were discovered. The developer exports the file along with a list of methods that must be exported to utilise.

Secondly, the developer runs the method `getSingleVehicle` which finds a single id by its `_id` field. The method checks if the id exists in the database and returns the vehicle data for that id in a JSON. If ID is successful, a status of 200 is returned. A 404 error is generated with the message "Id number not found" if the id is not found.

Thirdly, the developer created a method, `addVehicle` which is a little different as it is a POST request. A POST request expects JSON within the body. That JSON should have the properties in vehicle schema, make, model, reg etc. `Vehicle.create` is what creates a vehicle and `vehicleData` is the data being implemented in the request body. Vehicle data is equal to the request body which means whatever the request body is, is now equal to the `vehicleData`. If the creation of the vehicle is successful, it responds with 201, indicating that there has been a change. If it is not successful, it responds with a 422 which is a unprocessable entity. This indicates that there is an error in the request body that is attempting to be submitted.

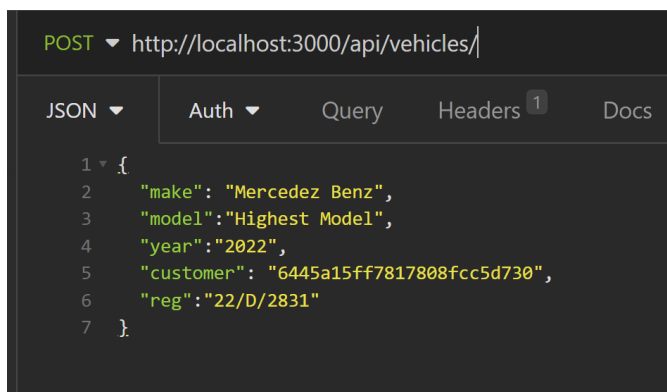


Figure 36 - Request body in Insomnia

Below in Figure 37, is a validation error, which is also part of mongoose. If the developer attempts to add data but the data that is trying to be added in the request body, does not follow the rules of the schema that you are trying to add that data to, a Validation Error will be returned.

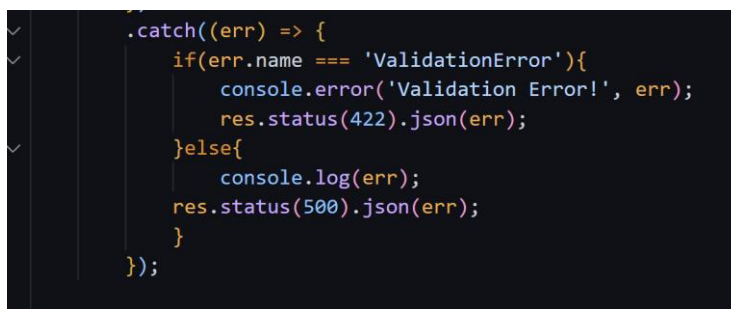


Figure 37 - Validation Error

Vehicle Routes

In the routes file the developer imported the vehicle controller. If we get vehicles, we want to get all the methods that were exported in the controller file, for example, `getAllVehicles`, `getSingleVehicle`, `addVehicle` and `editVehicle` as seen in Figure 38 below.

```
//Requiring modules
const express = require('express');
const router = express.Router();
const { loginRequired } = require('../controllers/auth_controller');

const {
  getAllVehicles,
  getSingleVehicle,
  addVehicle,
  editVehicle
} = require('../controllers/vehicle_controller');

/////////ROUTES/////////
router
  .get('/', loginRequired, getAllVehicles)
  .get('/:id', getSingleVehicle)
  .post('/', addVehicle)
  .put('/:id', editVehicle)

/////////

module.exports = router;
```

Figure 38 - Vehicle Routes

5.4.1.6 User – Login & Register.

If a user is not registered or logged in, they should be unable to make any application requests.

User Schema

Firstly, the developer created the user model, which is the schema. The user schema has the name, email, password, and mechanic. The mechanic data type is Boolean, which means that if the mechanic is logged in, `mechanic = true`; if the customer is logged in, `mechanic = false`. The developer introduced a few more rules, including trim, lowercase, and unique, as seen in Figure 39 below. Trim is used to remove any whitespace between a string. Lowercase is set to true, indicating that it will be converted to lowercase and saved in the database. Unique ensures that the email address is unique in the database, so that if someone has already registered with that email address, they cannot re-register with the same email address. The timestamp indicates when a user was created or modified, mongoose assigns `createdAt` and `updatedAt` to the user schema.

```

models > [JS] user_schema.js > [JS] userSchema > [JS] mechanic
1  const {Schema, model} = require('mongoose')
2  const bcrypt = require('bcryptjs');
3  const jwt = require('jsonwebtoken');
4
5  const userSchema = new Schema ({
6      name: {
7          type:String,
8          trim: true,
9          required: [true, 'Full name field is required']
10     },
11     email: {
12         type: String,
13         unique: true,
14         lowercase: true,
15         trim: true,
16         required: [true, 'Email is required']
17     },
18     password: {
19         type: String,
20         required: [true, 'Pasword is required']
21     },
22     mechanic: {
23         type: Boolean,
24     }
25 }, { //time updated, created at
26     timestamps: true
27 })
28 userSchema.methods.comparePassword = function(password){
29     return bcrypt.compareSync(password, this.password, function(result){
30         return result;
31     }); //Synchrony tests a string against a hash
32 };
33
34 };
35 module.exports = model('User', userSchema)

```

Figure 39 - User Schema

User Controller

Register function needs to run when a user goes to /register, the controller then connects to the model and adds that user to the database. If the model, then realises that the email is already in use it will send back an error.

Firstly, the developer created a newUser object which gets the users details from the request body. Creating a new user that has a name, email, password. To encrypt the password the developer installed and imported bcrypt, newUser.password refers to the password being used to register the user. The password is then going to be hashed and then

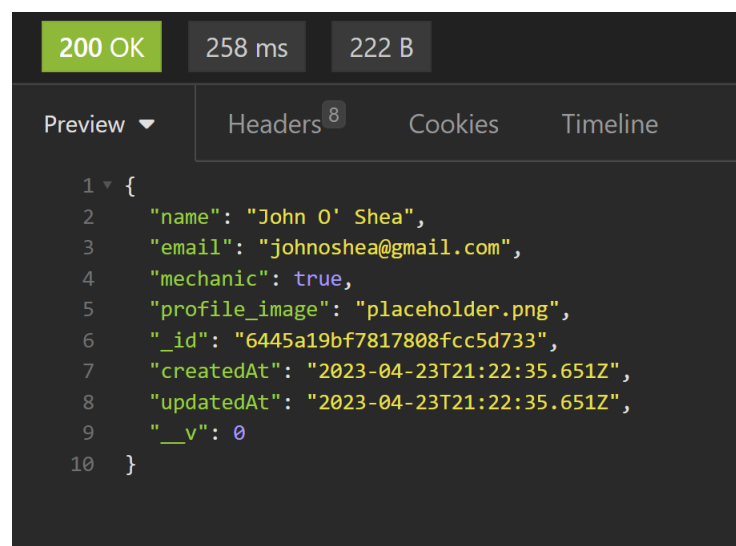
stored as the user's password. The 10 makes the encryption more difficult to crack by adding a little amount of salt. `newUser.save` saves the user; if an error occurs, it returns a 404-error message; otherwise, it returns the user's information in JSON, minus the password, which is set to undefined as seen in Figure 40. For security purposes, the password is encrypted in the database but not returned as JSON.

```
const registerUser = (req, res) => {
  //check if the user is valid an be stored
  //gets the users details from the request body
  let newUser = new User(req.body)

  //save user to the db
  //password will be encrypted and then saved as the users password in the database
  newUser.password = bcrypt.hashSync(req.body.password, 10);

  newUser.save((err, user) => {
    if(err){
      return res.status(400).send({
        message:err
      })
    }
    else{
      user.password = undefined
      return res.json(user)
    }
  })
}
```

Figure 40 - Register User



```
200 OK 258 ms 222 B
Preview Headers Cookies Timeline
1 {
2   "name": "John O' Shea",
3   "email": "johnoshea@gmail.com",
4   "mechanic": true,
5   "profile_image": "placeholder.png",
6   "_id": "6445a19bf7817808fcc5d733",
7   "createdAt": "2023-04-23T21:22:35.651Z",
8   "updatedAt": "2023-04-23T21:22:35.651Z",
9   "__v": 0
10 }
```

Figure 41 - password not returned in JSON, stored in the database

The user is not going to write an encrypted password, the password they will use is their actual password. For example, secret123. The backend will receive the user's password and

use it to decrypt the password stored in the database before verifying that it is the right password.

User.FindOne retrieves the user from the database and checks to see whether the email in the request body matches the email for that user. The developer created a custom function to compare the password which was created in the user schema. The function takes in a password, sees if they match & returns the results. Using a bcrypt method compareSync the developer compared password with this.password which refers to the current user and returns results.

```
userSchema.methods.comparePassword = function(password){
  return bcrypt.compareSync(password, this.password, function(result){
    return result;
  }); //Synchrony tests a string against a hash
};
```

Figure 42 - Compare Password function

After that, the developer installed jsonwebtoken and imported jwt to the user controller. Once this was installed it became possible for the developer to generate the token. Sign will create a token and it takes in two parameters, the first is the payload, the second is the secretOrPrivateKey. Whatever is in the method will be used by Sign to create a token, such as an email address, a name, or an ID mechanic. It uses the app key which is the secret or private key from the .env file, which is the name of the project, to validate the token the same key is used.

```
    else {
      let token = jwt.sign({
        email: user.email,
        name: user.name,
        _id: user._id,
        mechanic: user.mechanic
      }, process.env.APP_KEY);
      //Once logged in displays message all good.

      user.password = undefined;
      res.status(200).json({
        msg: 'Logged in to PPCI',
        token,
        user
      });
    }
  }
```

Figure 43 - JSON web token

User Routes

The developer created a file `user.js` within the routes folder. The developer imported the user controller and added the login and register routes which are both post requests as seen below in Figure 43.



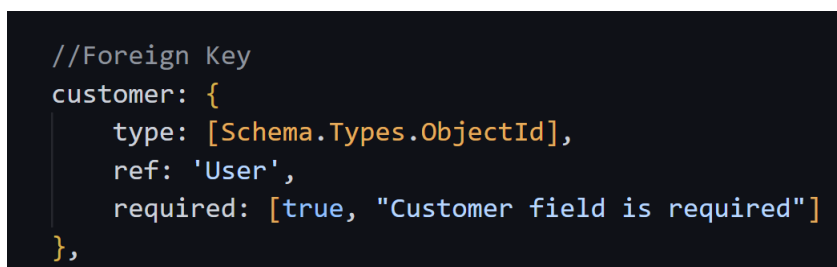
```

package.json M  vehicle_schema.js M  vehicle_controller.js  JS
routes > JS users.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  const {
5    loginUser,
6    registerUser
7  } = require('../controllers/user_controller');
8
9  router
10   .post('/login', loginUser)
11   .post('/register', registerUser)
12
13
14  module.exports = router;

```

Figure 44 - User Schema

The developer used the same method as vehicle schema, controller, and routes to build the other collections. Once the collections grew, the more collections connected or had relations. Using the Entity Relationship that was created in the design chapter the developer was able to tell easily what objects related to each other within the system. For example, one customer can have many vehicles. Using foreign keys within the schema aloud the developer to make these connections.



```

//Foreign Key
customer: {
  type: [Schema.Types.ObjectId],
  ref: 'User',
  required: [true, "Customer field is required"]
},

```

Figure 45 - Vehicle schema, customer FK

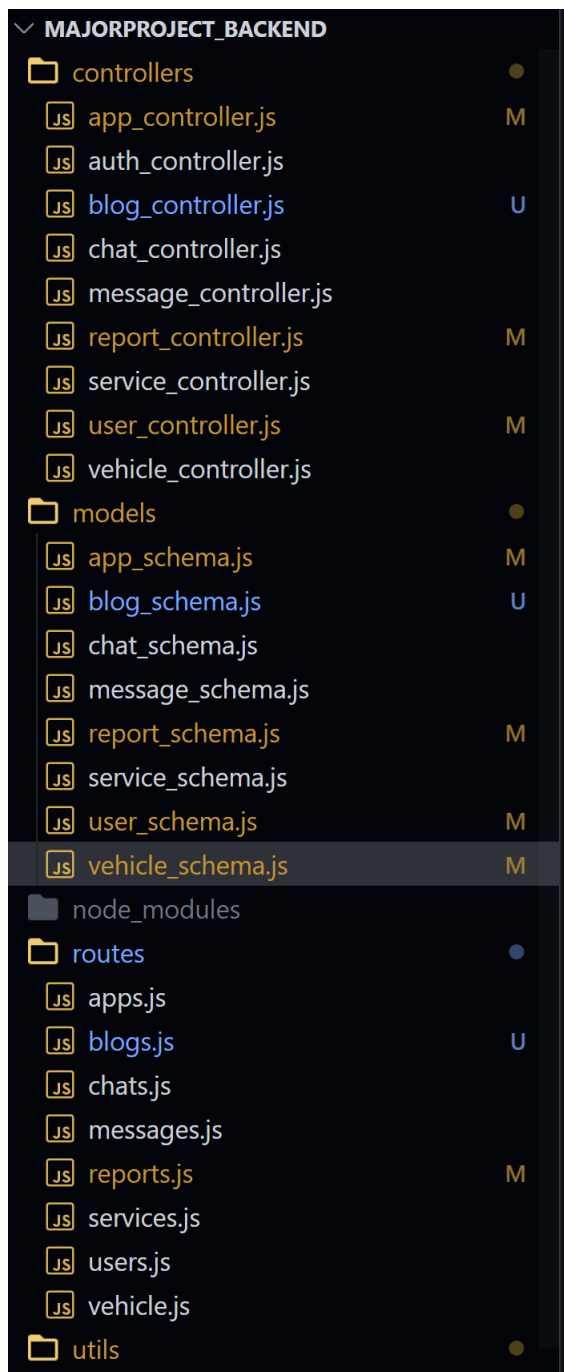


Figure 46 - Folder structure of the controllers, models and routes

5.5 Setting up the frontend environment

5.5.1.1 Expo Go.

As the developer was new to React Native, they chose to use Expo Go. Expo is a collection of React Native-based tools and services. It made it possible for the developer to begin writing code right away. Firstly, the developer downloads the Expo Go app from the apple play store. With the condition that the phone was linked to the same wireless network as the computer, this app permitted the developer to run the application on their mobile device.

On iOS, the developer scanned the QR code from the application's terminal using the camera's built-in scanner.

5.5.1.2 Creating a new React Native project

The developer began running the react native application by using the command 'npx create-expo-app MajorProject_Frontend,' the developer then ran the command `cd MajorProject_Frontend` which changes the current working directory. Following this, the developer ran the command `npx expo start` which successfully ran the application and it was ready to be modified.

To begin, the developer experimented with the `app.js` file, adjusting text values, adding pictures, and buttons, among other things, to acquire a feel for the technology.

Firstly, the developer got started with the navigation through to create structure within the application. React navigation is a standalone package that allows developers to include navigation capabilities in their applications. The developer installed the react-navigation library into the application. Each navigation in React Navigation resides in its own library, which serves as the navigators' common framework and building blocks.

```
> Stopped server
PS C:\Users\rinto\github-classroom\IADT-projects\y4-project-amyrintoul\Frontend_MajorProject> npm install @react-navigation/native

added 11 packages, and audited 1609 packages in 5s

102 packages are looking for funding
  run `npm fund` for details

5 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\rinto\github-classroom\IADT-projects\y4-project-amyrintoul\Frontend_MajorProject>
```

Figure 47 - Installing react navigation package

```
PS C:\Users\rinto\github-classroom\IADT-projects\y4-project-amyrintoul\Frontend_MajorProject> npx expo install react-native-screens react-native-safe-area-context
> Installing 2 SDK 48.0.0 compatible native modules using npm
> npm install

added 4 packages, and audited 1613 packages in 3s

102 packages are looking for funding
  run `npm fund` for details

5 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\rinto\github-classroom\IADT-projects\y4-project-amyrintoul\Frontend_MajorProject>
```

Figure 48 – React native screens

```
PS C:\Users\rinto\github-classroom\IADT-projects\y4-project-amyrintoul\Frontend_MajorProject> npm install @react-navigation/native-stack

added 2 packages, and audited 1615 packages in 8s

102 packages are looking for funding
  run `npm fund` for details

5 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\rinto\github-classroom\IADT-projects\y4-project-amyrintoul\Frontend_MajorProject>
```

Figure 49 - Installing react native stack

The application was able to transition between screens and handle the navigation history by utilising React Navigations' native stack navigator. Based on the path a user has travelled inside the programme, the stack navigator controls the appearance of the relevant screen.

The first thing the developer did was use the `createNavigationStackNavigator` method to build a native stack navigator. This function produces an object with the values screen and navigator. As you can see in Figure 50, the navigator contains screen elements as it is children which define the configuration routes. The screen component accepts a name which allows the user to navigate to that specific screen. As seen below in Figure 50, the 'welcome' route corresponds to the `WelcomeScreen` component, the `SecondScreen` route corresponds to the `SecondScreen` components, and so forth.

```

<NavigationContainer>
  /* if logged in is true, show homepage otherwise show the other screens */
  {isLoggedIn ?
    <Stack.Navigator initialRouteName="Root" screenOptions={{headerShown: false,}}>

      <Stack.Screen name="Home" screenOptions={{headerShown: true,}}>
        {props => <Tabs {...props} authUser={authUser} onisLoggedin={onisLoggedin} isLoggedIn={isLoggedIn} isMechanic={isMechanic} />}
      </Stack.Screen>
      <Stack.Screen name="MechanicProfile" component={MechanicProfile} screenOptions={{headerShown: true, }} onisLoggedin={onisLoggedin} />
      <Stack.Screen name="AppointmentScreen" component={AppointmentScreen} screenOptions={{headerShown: false}} onisLoggedin={onisLoggedin} />
      <Stack.Screen name="ReportScreen" component={ReportScreen} screenOptions={{headerShown: false}} onisLoggedin={onisLoggedin} />
      <Stack.Screen name="SingleReport" component={SingleReport} screenOptions={{headerShown: false}} onisLoggedin={onisLoggedin} />
      <Stack.Screen name="BlogPost" component={BlogPost} screenOptions={{headerShown: false}} onisLoggedin={onisLoggedin} />

    </Stack.Navigator> :
    <Stack.Navigator screenOptions={{headerShown: false}} >
      <Stack.Screen name="Welcome" component={WelcomeScreen} />
      <Stack.Screen name="SecondScreen" component={SecondScreen} />
      <Stack.Screen name="Login">
        {props => <LoginScreen {...props} onisLoggedin={onisLoggedin} />}
      </Stack.Screen>
      <Stack.Screen name="Register" component={RegisterScreen} />
    </Stack.Navigator>
  }
</NavigationContainer>

```

Figure 50 - Navigation Container

The developer set `isLoggedIn` is true show the homepage (which contains the tab navigation), the mechanic Profile, Appointment Screen, Single Report, Blog Post and Report Screen, otherwise show the welcome screen, second screen, login and register screen. To access the crucial screens, the user must first register and log in. To access the crucial screens, the user must first register and log in.

To move between screens the developer used the navigation prop that is passed down to the screen components.

Once the user is logged in the developer implemented a bottom tab navigator. The user may navigate between several routes using this straightforward tab bar at the bottom of the screen. To utilise this navigator, the developer installed the dependencies as seen in Figure

51.

```

PS C:\Users\rinto\github-classroom\IADT-projects\y4-project-amyrintoul\Frontend_MajorProject> npm install @react-navigation/bottom-tabs
>>
>>

added 8 packages, and audited 1661 packages in 5s

110 packages are looking for funding
  run `npm fund` for details

5 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

```

Figure 51 - Bottom tab bar navigation

Instead of a `Stack.Navigator` and `Stack.Screen`, the bottom tab bar consists of `Tab.Navigator` and `Tab.Screen`. There are several options that the developer used to configure the screens. These are specified under `screenOptions`. The developer set the labels to false as they chose to just use tab bar icons for the overall bottom navigation. If an icon is pressed the active colour changes to purple otherwise the icons are grey.

The navigation changes whether a mechanic is logged in or a client. Both users press the same icon but are brought to different screens. The developer made this possible using if statements. When a mechanic hits the document icon, they are sent to an appointments page, however when a client presses the document icon, they are taken to the see all reports screen. If a mechanic is logged in, display the mechanic homepage; otherwise, show the client homepage.

```

    if(isMechanic){
      mechanicPage = (
        <Tab.Screen name='Appointments' children={() => <AppointmentViewScreen navigat
          options={{
            tabBarIcon: ({ color, size, focused }) => (
              <View style={{alignItems: 'center', justifyContent: 'center', top:
                <AntDesign name="filetext1" size={size} style={{color: focused ?
              </View>
            )},
          }}/>
        );
      }
    }

    let mechanicHomePage = (
      <Tab.Screen name='Home' children={() => <HomeScreen authUser={authUser} nav
        options={{
          tabBarLabel: 'Home',
          tabBarIcon: ({ color, size, focused }) => (
            <View style={{alignItems: 'center', justifyContent: 'center', top:
              <Ionicons name="ios-home" size={size} style={{color: focused ? '#
            </View>
          )},
        }}/>
      );
    }

```

Figure 52 - bottom tab bar, if statements

5.6 Conclusion

The implementation chapter discusses the scrum agile methodology and how it was implemented for the project's application. The process was sectioned into two-to-four-week sprints, these sprints were crucial as they allowed the developer to stay motivated, organised and up to date with each task that needed completed.

The integrated development environment (IDE) for this application was also covered in detail by the developer. Visual Studio Code was the IDE used for both the backend and frontend applications.

The developer began by creating a database using MongoDB. The developer went through each phase of developing the database and how the developer connected the backend to MongoDB. Once the database was created the developer began setting up the backend.

The backend was implemented using Node.js and Express.js. The first thing that happens is the backend goes through 'Routing'. The request is sent into 'Routing,' the routes job is to route to the appropriate controller. Routing will check whether 'appointments' exists on the server, if it exists it will then send that request to the appropriate controller. For example, if you try to get appointment/id number it will then find that route and connect it to that specific controller.

Majority of the work is done in the controller, it opens the request and makes sure if the user is creating an appointment, it has all the needed attributes, for example, title, date, description etc, making sure it has all the details that is required before connecting to the Model. If it finds that the request is invalid, it will respond back to the client. This is the response that the developer sees in the frontend.

Another example is the login request, if the users logs in, that request comes into routing, routing will find the /login route, it will then say that that route should run our /login controller file, it will open the login controller file and that is where all the logic is. Logic means that in the controller it is checking whether the email was passed in as a request. If the user forgets to add an email address in the controller will respond with an error.

After building a strong backend, the developer started putting it into practise and testing using Insomnia. The developer also concentrated on building the frontend using React Native

6 Testing

6.1 Introduction

The many techniques for testing the application's backend and frontend will be covered in the testing chapter. This chapter is presented in three sections: API Testing, Functional Testing and User Testing.

The developer will perform API testing for the backend. Insomnia, a REST client framework, will be used to test the application's REST API. Each API route will be tested and discussed by the developer.

To validate the application's functioning, the developer will conduct functional tests based on the criteria provided in the requirements chapter. Functional testing is a method of testing that aims to determine if each application feature functions in accordance with the needs of the program. Each functionality is compared to the associated requirement to see if its output matches the end user's expectations. Testing is carried out by supplying sample inputs, recording the outcomes, and confirming that the actual results match the predicted results. (Microfocus).

User testing covers the entire range of user experience a client has with the product. This will include all their responses to the item, from when the client obtains it to when they stop using it. The developer will give the client a set of goals to see their outcome and conduct a set of results.

6.2 API Testing

API testing is the process of sending requests to an API and monitoring the responses to ensure they are behaving as expected. API testing is intended to get access to an API's functionality, dependability, performance, and security. As a result, this is a key component of the API development lifecycle. (HubSpot, Anna Fitzgerald. 2022). API testing may be approached in numerous ways, including functional testing, load testing and validation testing.

The developer chose Insomnia as an API testing tool which helps build high quality API's. Insomnia is a REST API, REST stands for (Representational State Transfer). This means when a client requests a resource using a REST API, the server transfers back to the current state of the resource in a standardized representation. XML, PHP, JSON, and plain text are just a few of the many ways in which this data might be expressed. (HubSpot, Jamie Juviler. August 2022)

The developer began testing the API routes before hosting the backend & beginning on the frontend. JSON is the most popular file format used for testing APIs, as you will see below, the developer implements this format.

API CALL	ACTION
POST/ api/users/login/	Allows the user/mechanic to login
POST/api/users/register	Allows the user or mechanic to register
GET/api/apps/	Gets a list of all appointments
GET/api/reports	Gets a list of all reports
GET/api/reports/users	Get all reports associated to a certain user
Get/api/reports/{id}	Gets a specific report by id
POST/api/reports	Creates a report
PUT/api/reports/{id}	Updates a specific report by id
GET/api/vehicles	Get all vehicles
GET api/services	Get all services
GET/messages	Get all messages

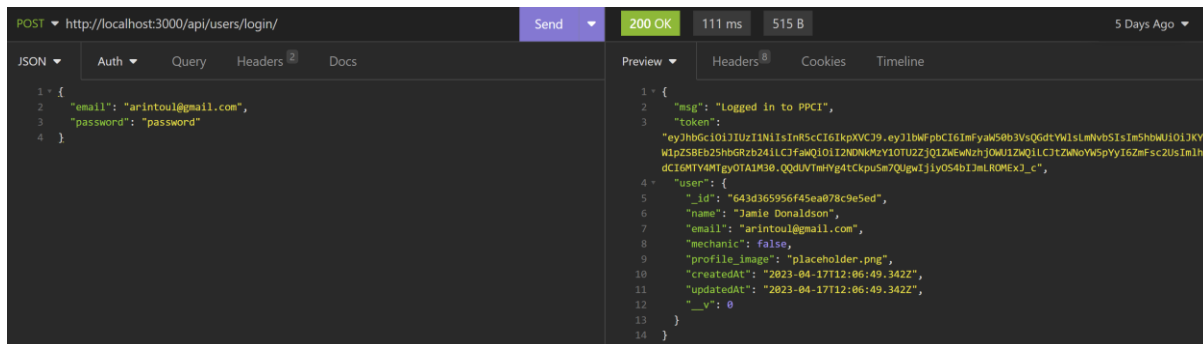


Figure 53 - Logged in request and response

Figure 53 above displays a POST request which is sent to the hosted applications API. The API expects an email, password as shown in the JSON body. After the request is sent, the expected response should return a msg stating the client is logged in to Pre-Purchase Car Inspections, a JSON web token, and data that is related to the users that has just logged in. Once the user is logged in, they get a HTTP status code of 200 meaning the response has been successful.

6.2.1 GET request for all reports

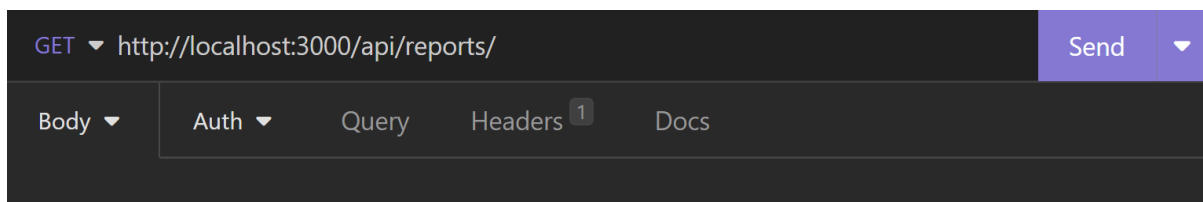


Figure 54 - GET request for reports

Figure 54 shows a GET request to the hosted applications API. This request should show a list of all the reports that are in the MongoDB database and any relational data.

```
{
  "_id": "643ea7d7cf2f3e156b32b9ef",
  "customer": {
    "_id": "643ea7bfcf2f3e156b32b9ed",
    "name": "Amy",
    "email": "alrintoul@gmail.com",
    "password": "$2a$10$71NiVLlsaZ5m6l/XtddBAe916clkk3J/FfrgSknaV8FTTxLAd6jhi",
    "mechanic": false,
    "profile_image": "placeholder.png",
    "createdAt": "2023-04-18T14:22:55.201Z",
    "updatedAt": "2023-04-18T14:22:55.201Z",
    "__v": 0
  },
  "mechanic": {
    "_id": "6426a8b8d29e319caf632710",
    "name": "Amy",
    "email": "mechanica@gmail.com",
    "password": "$2a$10$Mb.yf9ba0RWo6s13J9OpduGVh2H0Q1ybondd2D/1xBU4osCqPZwm",
    "mechanic": true,
    "profile_image": "placeholder.png",
    "createdAt": "2023-03-31T09:32:40.853Z",
    "updatedAt": "2023-03-31T09:32:40.853Z",
    "__v": 0
  },
}
```

Figure 55 - GET Response of all reports

```
*   "vehicle": {
*     "_id": "63ec07a4c426c307878e3f56",
*     "customer": [
*       "63ebac88a79f71a901cf5f3f"
*     ],
*     "make": "BMW",
*     "model": "S-Line",
*     "year": "2022",
*     "reg": "09-D-42094",
*     "__v": 0
*   },
  "mileage": "Green",
  "lights": "Yellow",
  "windscreen": "Red",
  "wipers": "Yellow",
  "horn": "Green",
  "ac": "Green",
  "air_filter": "Yellow",
  "oil_fuilds": "Yellow",
  "belts": "Red",
  "suspension": "Yellow",
  "steering": "Green",
  "exhaust": "Green",
  "test_drive": "Green",
  "diagnostics": "Green",
  "body_work": "Green",
  "tyres_front_left": "Green",
  "tyres_front_right": "Red",
  "tyres_back_left": "Red",
  "tyres_back_right": "Green",
  "brakes_front": "Red",
  "brakes_back": "Red",
  "comments": "Overall the car is brilliant, needs new set of brakes",
  "date": "2019-06-12T12:34:00.000Z",
  "__v": 0
}
1
```

Figure 56 - Continued response

The answer to this request in Figures 55 and 56 above was a success, giving the status code of 200, suggesting the resource was successfully obtained and delivered in the message body. The answer displays the specifics of each report, such as miles, lights, and so on.


```

1 * [
2 * {
3   "_id": "643ea7accf2f3e156b32b9eb",
4   "customer": {
5     "_id": "643d365956f45ea078c9e5ed",
6     "name": "Jamie Donaldson",
7     "email": "arintoul@gmail.com",
8     "password": "$2a$10$ZZ0I3f.0wVaBAVYr7KkDR0izXT6qLDOETyr1bsterGj08YdzcunJW",
9     "mechanic": false,
10    "profile_image": "placeholder.png",
11    "createdAt": "2023-04-17T12:06:49.342Z",
12    "updatedAt": "2023-04-17T12:06:49.342Z",
13    "__v": 0
14  },
15  "mechanic": "6426a8b8d29e319caf632710",
16  "vehicle": { ← 7 → },
27  "mileage": "Green",
28  "lights": "Yellow",
29  "windscreen": "Red",
30  "wipers": "Yellow",
31  "horn": "Green",
32  "ac": "Green",
33  "air_filter": "Yellow",
34  "oil_fulids": "Yellow",
35  "belts": "Red",
36  "suspension": "Yellow",
37  "steering": "Green",
38  "exhaust": "Green",
39  "test_drive": "Green",
40  "diagnostics": "Green",
41  "body_work": "Green",
42  "tyres_front_left": "Green",
43  "tyres_front_right": "Red",
44  "tyres_back_left": "Red",
45  "tyres_back_right": "Green",
46  "brakes_front": "Red",
47  "brakes_back": "Red",
48  "comments": "Overall the car is brilliant, needs new set of brakes",
49  "date": "2019-06-12T12:34:00.000Z",
50  "__v": 0
51 }
52 ]

```

Figure 60 - Response from the GET request

Figure 60 shows the response to the GET api/reports/user request. It shows the list of reports associated with the person that is logged in. In this case one report is related to the user that is currently logged in. In Figure 10 it shows the details of the user that is currently logged in & their token. The token is then used in the Headers Authorization along with Bearer as the login is required to make this specific request. The response shown in Figure 12 shows the customers details which you can see correspond to the details shown in Figure 10. If a mechanic is logged in the response would show the reports associated to that specific mechanic.

6.2.3 GET request for a report with a specific id.

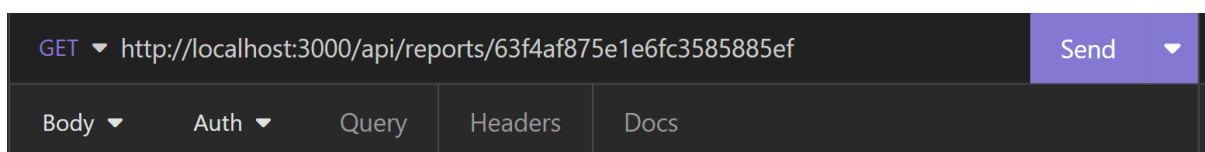


Figure 61 - Request for a single report

Like the GET requests for all reports, the request for a single report does not need a JSON body. Figure 61 shows the developer requested the report with the id as shown above. This response shown show a single report with that specific id.

```
1 * {
2   "_id": "643ea7accf2f3e156b32b9eb",
3   "customer": { ← 9 → },
14  "mechanic": { ← 9 → },
25  "vehicle": {
26    "_id": "63ec07a4c426c307878e3f56",
27    "customer": [ ← 1 → ],
30    "make": "BMW",
31    "model": "S-Line",
32    "year": "2022",
33    "reg": "09-D-42094",
34    "__v": 0
35  },
36  "mileage": "Green",
37  "lights": "Yellow",
38  "windscreen": "Red",
39  "wipers": "Yellow",
40  "horn": "Green",
41  "ac": "Green",
42  "air_filter": "Yellow",
43  "oil_fullds": "Yellow",
44  "belts": "Red",
45  "suspension": "Yellow",
46  "steering": "Green",
47  "exhaust": "Green",
48  "test_drive": "Green",
49  "diagnostics": "Green",
50  "body_work": "Green",
51  "tyres_front_left": "Green",
52  "tyres_front_right": "Red",
53  "tyres_back_left": "Red",
54  "tyres_back_right": "Green",
55  "brakes_front": "Red",
56  "brakes_back": "Red",
57  "comments": "Overall the car is brilliant, needs new set of brakes",
58  "date": "2019-06-12T12:34:00.000Z",
59  "__v": 0
60 }
```

Figure 62 - Get report by Id

If the response is successful, a single report, as seen above, should be shown. As you can see it has the same Id as shown in the GET request made in Figure 62. It also shows any data that is related to this specific report i.e., mechanic, customer, and vehicle.

6.2.4 POST request which creates a report.

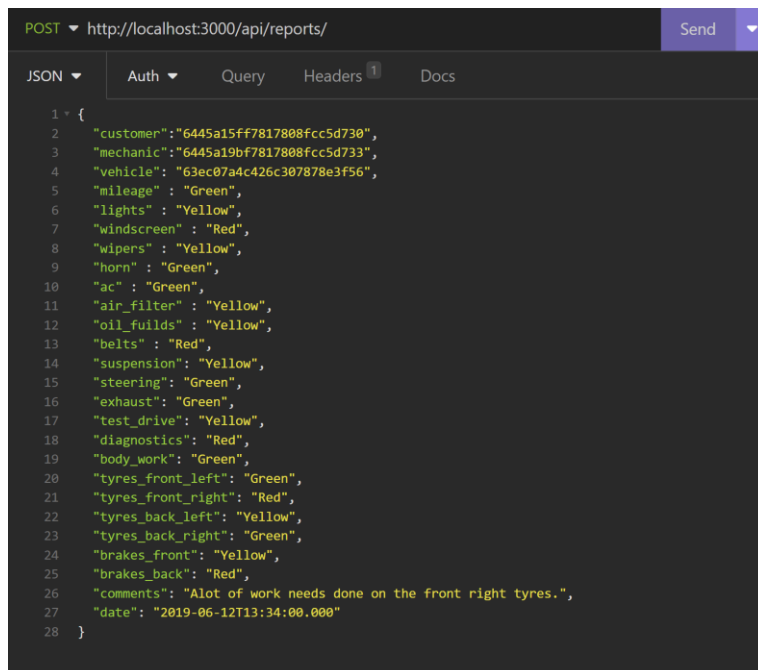
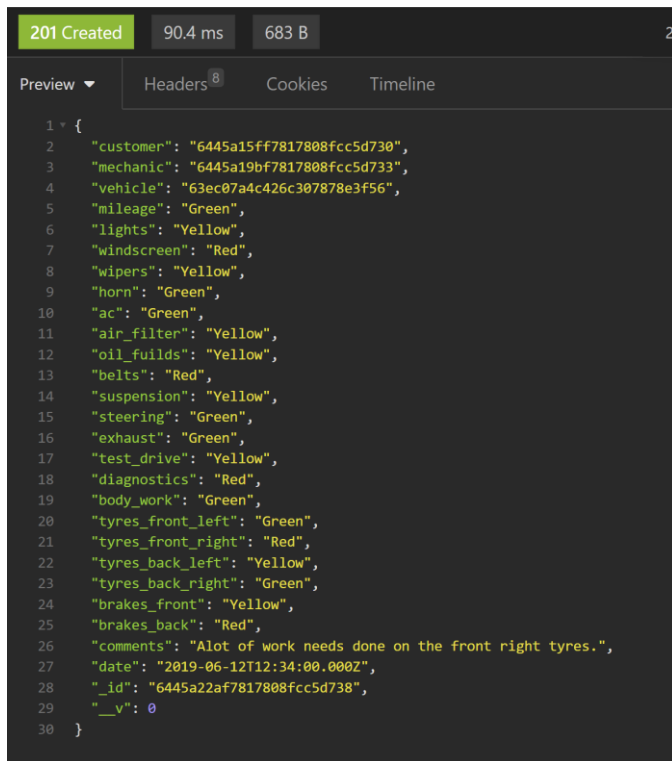


Figure 63 - POST request to create a report

Figure 63 shows the developer making a POST `api/reports` request. Creating a report requires a JSON body as shown above. The JSON body requires specific text to make the request i.e., mileage, lights, customer ID, mechanic ID etc. The request will not be successful without these inputs.



```

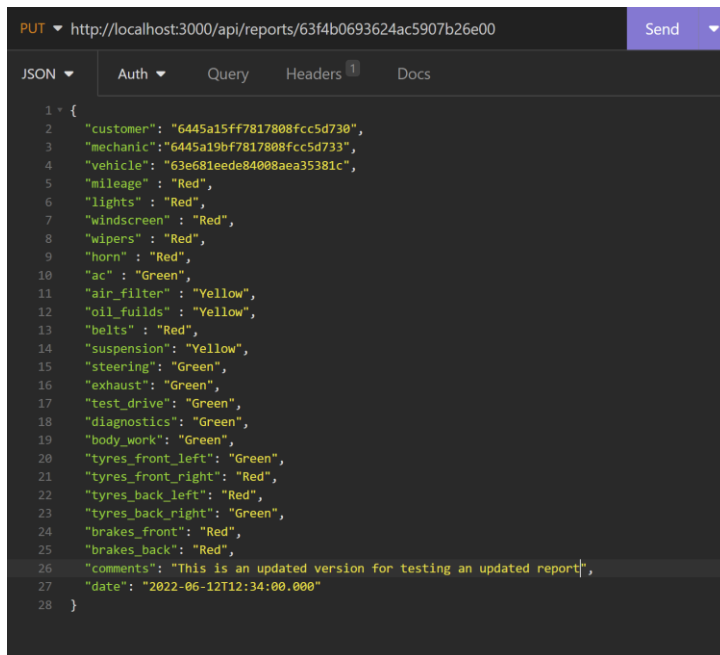
201 Created 90.4 ms 683 B
Preview Headers Cookies Timeline
1 {
2   "customer": "6445a15ff7817808fcc5d730",
3   "mechanic": "6445a19bf7817808fcc5d733",
4   "vehicle": "63ec07a4c426c307878e3f56",
5   "mileage": "Green",
6   "lights": "Yellow",
7   "windscreen": "Red",
8   "wipers": "Yellow",
9   "horn": "Green",
10  "ac": "Green",
11  "air_filter": "Yellow",
12  "oil_fuids": "Yellow",
13  "belts": "Red",
14  "suspension": "Yellow",
15  "steering": "Green",
16  "exhaust": "Green",
17  "test_drive": "Yellow",
18  "diagnostics": "Red",
19  "body_work": "Green",
20  "tyres_front_left": "Green",
21  "tyres_front_right": "Red",
22  "tyres_back_left": "Yellow",
23  "tyres_back_right": "Green",
24  "brakes_front": "Yellow",
25  "brakes_back": "Red",
26  "comments": "Alot of work needs done on the front right tyres.",
27  "date": "2019-06-12T12:34:00.000Z",
28  "_id": "6445a22af7817808fcc5d738",
29  "__v": 0
30 }

```

Figure 64 - Response for creating a report

Figure 64 shows the response when making the POST api/reports/ request. Once the report has been created you may request a single report with this ID; it will appear when the developer requests all report and, in each example, I have shown thus far.

6.2.5 PUT request which updates a specific report.



```

PUT http://localhost:3000/api/reports/63f4b0693624ac5907b26e00 Send
JSON Auth Query Headers Docs
1 {
2   "customer": "6445a15ff7817808fcc5d730",
3   "mechanic": "6445a19bf7817808fcc5d733",
4   "vehicle": "63e681eede84008aea35381c",
5   "mileage": "Red",
6   "lights": "Red",
7   "windscreen": "Red",
8   "wipers": "Red",
9   "horn": "Red",
10  "ac": "Green",
11  "air_filter": "Yellow",
12  "oil_fuids": "Yellow",
13  "belts": "Red",
14  "suspension": "Yellow",
15  "steering": "Green",
16  "exhaust": "Green",
17  "test_drive": "Green",
18  "diagnostics": "Green",
19  "body_work": "Green",
20  "tyres_front_left": "Green",
21  "tyres_front_right": "Red",
22  "tyres_back_left": "Red",
23  "tyres_back_right": "Green",
24  "brakes_front": "Red",
25  "brakes_back": "Red",
26  "comments": "This is an updated version for testing an updated report",
27  "date": "2022-06-12T12:34:00.000"
28 }

```

Figure 65 - Request to update a specific report

Figure 65 shows the developer sending a PUT request with a specific ID, this allows the developer to test whether the user can edit a report. As seen above the developer edited the comments input, using the JSON body. The response should show the updated version of this report.



```

1 {
2   "_id": "63f4b0693624ac5907b26e00",
3   "vehicle": "63e681eede84008aea35381c",
4   "mileage": "Red",
5   "lights": "Red",
6   "windscreen": "Red",
7   "wipers": "Red",
8   "horn": "Red",
9   "ac": "Green",
10  "air_filter": "Yellow",
11  "oil_fuils": "Yellow",
12  "belts": "Red",
13  "suspension": "Yellow",
14  "steering": "Green",
15  "exhaust": "Green",
16  "test_drive": "Green",
17  "diagnostics": "Green",
18  "body_work": "Green",
19  "tyres_front_left": "Green",
20  "tyres_front_right": "Red",
21  "tyres_back_left": "Red",
22  "tyres_back_right": "Green",
23  "brakes_front": "Red",
24  "brakes_back": "Red",
25  "comments": "Updated version for testing",
26  "date": "2022-06-12T11:34:00.000Z",
27  "__v": 0
28 }

```

Figure 66 - Updated version of a specific report

Figure 66 depicts the developer's modified version of the previously submitted request. answer displays the returned data for this report, along with the freshly modified comments.

6.2.6 DELETE request made for a specific report

6.2.7 Bad request, errors

To highlight how important testing may be at times, the developer adjusted several requests to highlight the difference in the response.

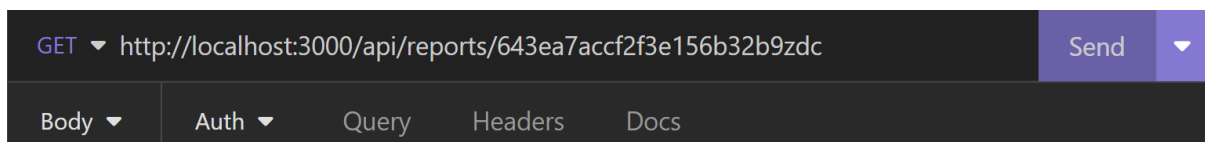


Figure 67 - Id with adjustment

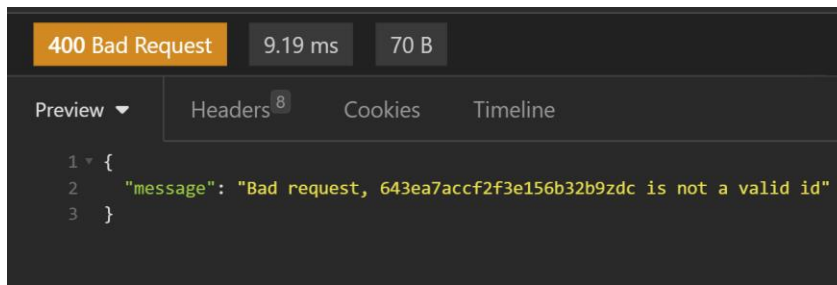


Figure 68 - 400 bad request

The ID was changed by the developer to illustrate that the ID is invalid and what the response will be. The ID is not valid, and a 400 Bad Request response code is shown.

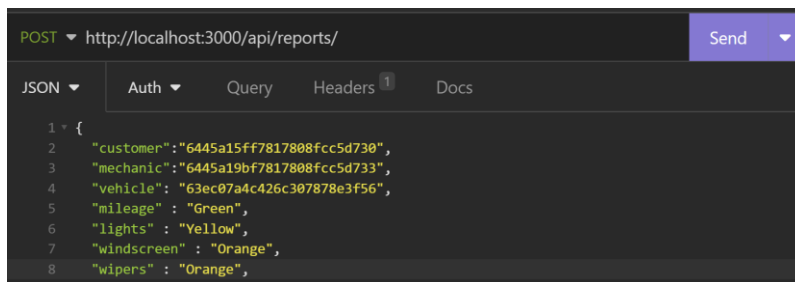


Figure 69 - Reports altered to show error

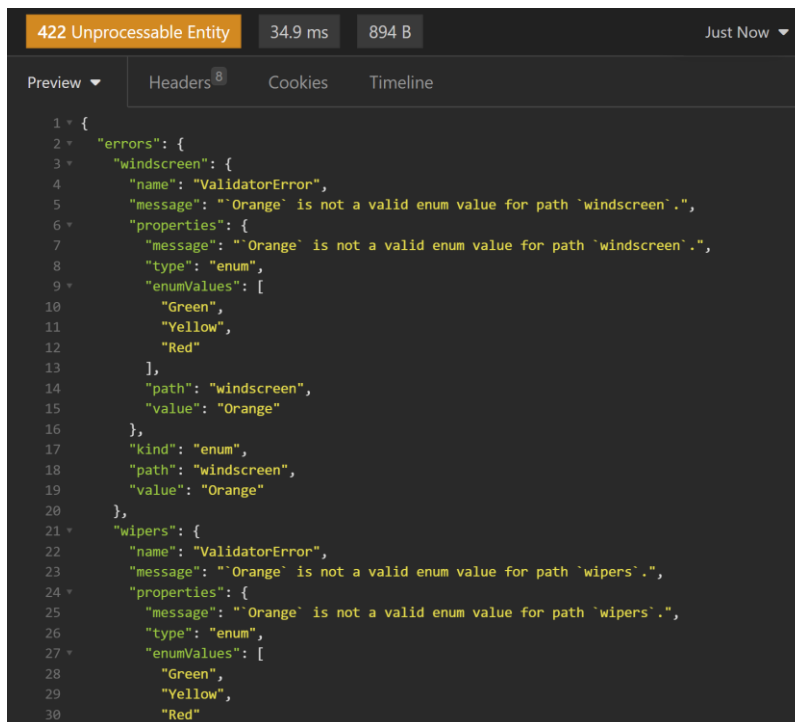


Figure 70 - 422, Unprocessable Entity

The developer changed the windscreen and wipers text inputs in the JSON body to orange. It returns a status code of 422, meaning there is a validation error. The validation error occurred because the developer changed the colour to orange when the input should have been Green, Yellow, or Red, as indicated in Figure 69 above. The message indicates to the

developer that orange is not a valid Enum value for the path which allows the developer to solve the issue with ease.

6.3 Functional Testing

This section outlines the functional tests performed on the app. These functional tests are classified as follows:

- Navigation
- CRUD

Functional testing ensures that the software produces the results that an end user is looking for by comparing each program function or feature to the requirements standards.

(TechTarget, Corinne Bernstein. Oct 2021). Functional testing was implemented by the developer as a technique for quality control. Functional testing analyses an application's ability to successfully carry out a particular task.

Functional testing is often performed using a Black Box Testing approach, which means that the tester is uninterested in the core logic of the system being evaluated. The sole outcome that the tester is concerned about is whether the actual output matches the intended output.

6.3.1 Navigation

Text No	Test Case	Input	Expected Output	Actual Output	Comment
1.	Welcome Screen	Get started button is pressed	Redirects to the login screen	Redirects to the login screen	User can successfully redirect to the login screen once they pressed the get started button
2.	Login Screen	Login button is pressed, or register link is pressed	Redirects to register screen or to the homepage once logged In	Redirects to register screen or to the homepage once logged In	User successfully redirects to the homepage once they pressed the login button and redirects to the register

					screen once the register now link was pressed
3.	Register Screen	Register button is pressed	Redirects to the login screen	Redirects to the login screen	User successfully redirects to the login screen when they pressed the register button
4.	Homepage	Blog card is pressed, mechanic profile is pressed, service button is pressed	Redirects to the blog screen, mechanics profile, appointment page	Redirects to the blog screen, mechanics profile, appointment page	User successfully redirects to the blog screen once they pressed the blog card, successfully redirect to the mechanic profile of the specific user they pressed, successfully redirected to the book an appointments page once they pressed the service button. Each page they could redirect back to the homepage.
	Mechanic Profile	Mechanic is pressed on the homepage	Views all information about the specific profile pressed, redirects back to the homepage	Views all information about the specific profile pressed, redirects back to the homepage	User successfully redirected to the mechanics profile and redirected back to the homepage

	Blog Screen	When the blog card is pressed on the homepage	Views all information about the blog and redirects back to the homepage	Views all information about the blog and redirects back to the homepage	User successfully viewed all the information on the blog screen and redirected back to the homepage once finished.
	Reports	The reports tab icon is pressed on the bottom tab bar	Views all reports, redirects back to the homepage or to a single report	Views all reports, redirects back to the homepage or to a single report	User successfully redirected to the reports page once they pressed the reports icon on the bottom tab bar
	Single Report	The reports card is pressed on the reports page	Redirects back to the reports page	Redirects back to the reports page	User successfully views the single report page and can redirect back to the reports screen
	Appointments	The plus icon is pressed on the bottom tab bar	Book an appointment. Redirects to the homepage or to the confirmation page	Book an appointment. Redirects to the homepage or to the confirmation page	User successfully redirected to the book an appointments page once they pressed the plus button on the bottom nav bar
	Messenger Screen	The messenger icon button is pressed on the bottom tab bar	Views all messages on the messenger screen	Views all messages on the messenger screen	User successfully redirected to the messenger screen once they pressed the messenger icon on the

					bottom tab bar
	Log out	Logout button is pressed on the bottom tab bar	Redirects back to the welcome screen once pressed logout	Redirects back to the welcome screen once pressed logout	When the user pressed the logout button on the bottom tab bar they successfully logged out and redirected to the welcome screen

6.3.2 CRUD

Test No.	Test Case	Input	Expected Output	Actual Output	Comment
1.	Add a new user to the application	Email Address, Username, Password	New user is added	New user added	User is successfully created & can now log in.
2.	Login to the application	Username, Password	Logged in & redirected to the homepage	Logs in & is redirected to the homepage	User successfully logs in and is redirected to the homepage
3.	Customer can choose a service	Chose Elite, Premium	Customer can click on a service and is redirected to the create appointment page.	Redirects to the appointments page	User can redirect to the appointments screen but does not show which service they chose
4.	Customer can create an appointment	Mechanic ID, Vehicle ID, Service ID, Location, Start time, End time Date.	Appointment is successfully created and published.	Customer is unable to create an appointment	Customer cannot create an appointment, the developer never managed to get that functionality working

5.	View all appointments for that specific user.	View appointments on homepage	A list of all the apps for that specific user is displayed	Customer can view all appointment in the database, not for specific user	Customer can view all appointment in the database, not for specific user
6.	Mechanic can create a report and assign it to a specific customer.	Customer ID, Vehicle ID, Mileage, Lights, Windscreen, Wipers, Horn, Ac, Air filter, Belts, Suspension, Steering, Test Drive, Diagnostics, Body work, Tyres, Brakes, Comments, Date	Report is successfully created and is published, assigned to a specific customer.	The mechanic can create a report and assign that report to a specific customer	The mechanic can create a report and assign that report to a specific customer
7.	View all reports for specific customer	Go to reports page	A list of all the reports for that specific user is displayed	The customer can view all reports in the database but not ones for just that specific client	The customer can view all reports in the database but not ones for just that specific client
8.	User can view a single report	Go to reports page and click on a specific report	A single report is displayed showing all the information for that report.		
9.	Pay for the service they chose.	Go to payment page	User can confirm appointment booking and pay for the service	User is unable to pay for the service	User is unable to pay for the service, this phase is incomplete

					and needs further work
	Logout	Press the logout button	Redirected to the welcome screen	Redirected to the welcome screen	User successfully logs out of the application & is redirected to the welcome screen

6.3.3 Functional Testing Results

The outcome of the functional testing reveals each functionality and whether it was effective or not. The navigation testing proved to be extremely successful. The navigation performs as predicted in terms of the user flow diagrams in the requirements chapter. Testing with proper and legitimate inputs produced the desired results.

Although CRUD testing was not as effective, it did show what functionality needed to be improved and what functionality was a success.

6.4 Conclusion

This project's testing phase covered the numerous ways the developer tested the programme. First, the developer demonstrates API testing, which examines the functionality, dependability, performance, and security of the hosted apps API. The REST API was tested using Insomnia, which is a client framework. The developer illustrated the process by which they conducted multiple API requests and the variety of responses that were returned. This was a critical step in the development of the API; without it, the backend would not have been as successful.

Afterwards the developer conducted functional testing which consisted of two steps CRUD and navigation. For quality assurance purposes, the developer used these methods of testing. The developer conducted black box testing to test the functioning of the software without looking at the underlying code structure, implementation details, or knowledge of the product's internal routes. These functional tests were developed in accordance with the requirements given in the requirements chapter. The functionality testing showed that the navigation throughout the application was extremely successful. A few incomplete CRUD features were discovered during the CRUD testing. These features still require work, and they also require greater attention. For each test, the developer defined a test case, which was then followed by the input, expected output, actual output, and any developer-provided comments.

7 Project Management

7.1 Introduction

Project management offers structure and control over the project environment so that the agreed-upon activities create the proper results that meet the expectations of the users. Under the leadership of the developer, project management is the regulated implementation of the project plan. The developer will go through how the project was handled in this chapter.

It depicts the project phases, beginning with the project idea and progressing through requirements gathering, project definition, design, implementation, and testing. It also examines Trello, GitHub, and project developer's diaries as project management tools.

The developer will discuss their reflection of the overall project. This will conduct of the developers views on the project, what they have learnt while developing phases their application, how the project went working with a supervisor, any technical skills they learnt and any extra skills that would help with the future of the applications development.

7.1.1 Testing

7.2 SCRUM Methodology

Scrum is an agile software development technique that is focused on interactive and incremental procedures. Scrum is a quick adaptive, flexible, and successful agile framework that is intended to provide value to the client throughout the project's development. In the previous chapter, the implementation chapter, the developer went into great depth about Scrum technique.

A backlog is a phrase that originated in the agile project management system and refers to a list of tasks which the developer has still to finish to complete a project. Microsoft Planner was used to organise the backlog, as stated in section 6.4.1 of this chapter.

7.3 Project Management Tools

7.3.1 Microsoft Planner

Microsoft Planner is a planning application that is available on the Microsoft 365 platform. Microsoft Planner is a team-work orientated tool which can be used in a variety of diverse ways. The developer selected to take advantage of this application to stay structured and focused on what needed to be completed, what they were working on, and what they had achieved. This tool also allowed the developer to communicate with their supervisor, allowing the supervisor to observe how the developer was progressing with their entire project.

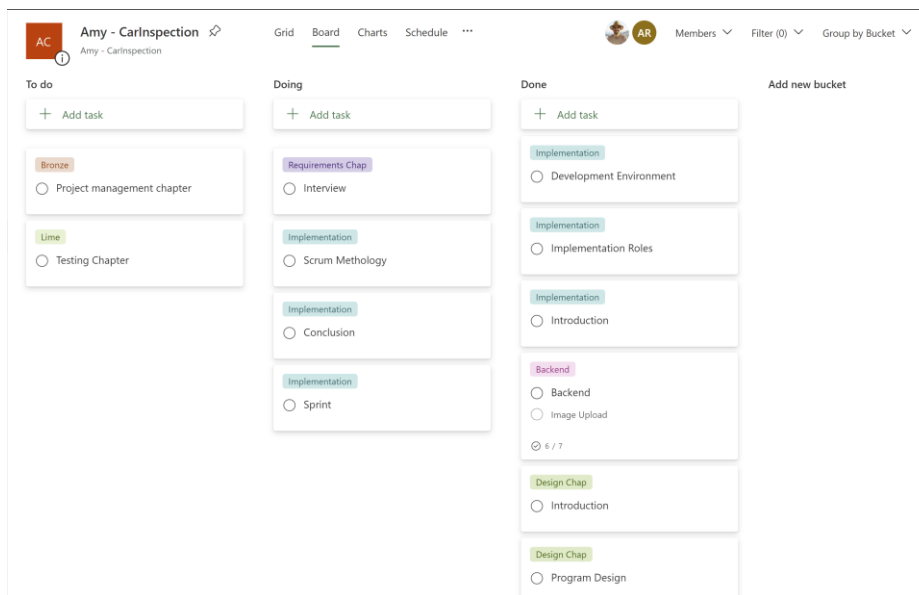


Figure 71 - Microsoft Planner

As seen above in Figure 71, the developer was able to drag and drop cards under every heading, and additionally label each one. The developer was able to assign each card a due date to allow them to keep notified on each task approaching its deadline.

7.3.2 GitHub

GitHub is an online software platform which is used for storing, tracking, and collaborating on software projects. It enables software developers and engineers to build remote, public-facing cloud repositories for free. All the code and documentation are in one location, making it easy to collaborate with others. Additionally, GitHub makes it relatively easy to keep track of modifications and go back to prior versions of your project. It increases the efficiency of coding by catching mistakes before pushing updates. (HubSpot, Jamie Juliver, October 2022)

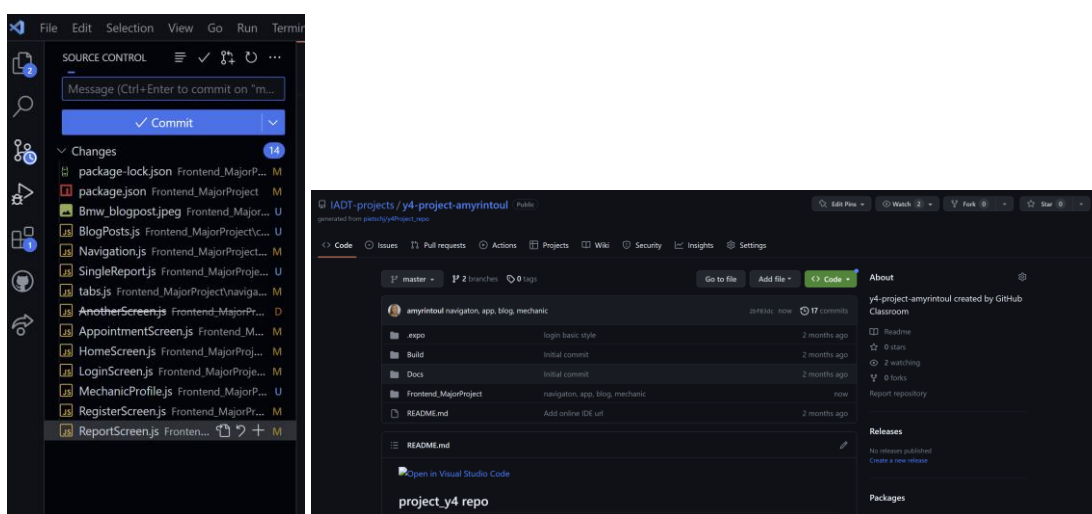


Figure 72 - Github commits waiting, github classroom for frontend

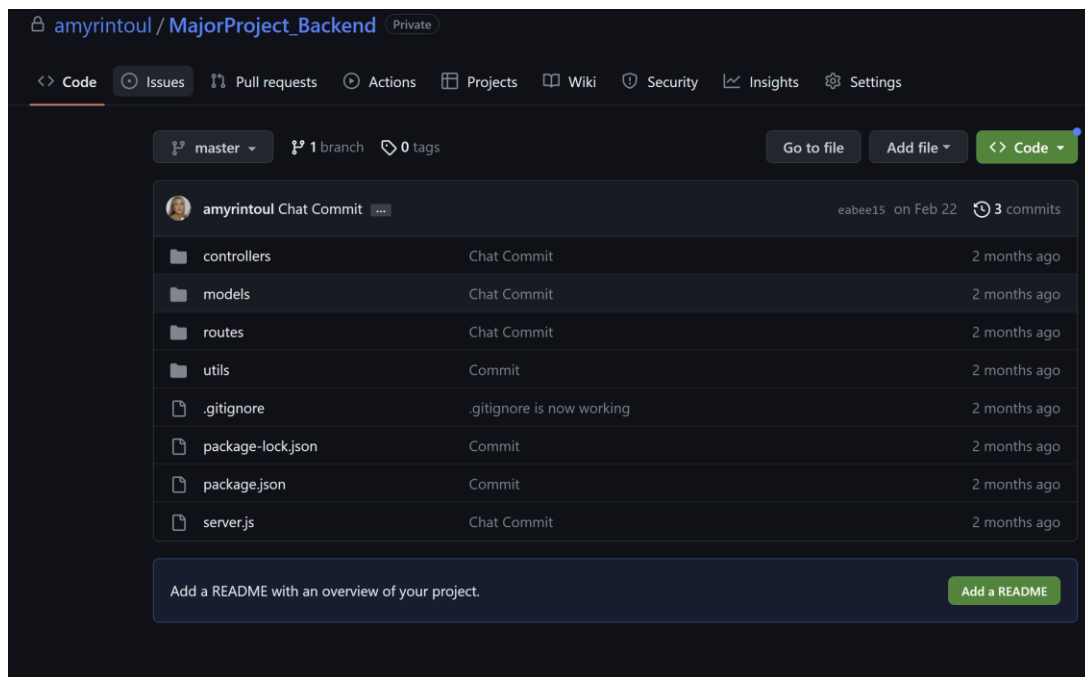


Figure 73 - Github Backend

Once the developer set up a repository for GitHub, they were able to copy it to their device, add and modify files locally, then “push” their changes back to the repository where the changes were displayed in public. The developer used GitHub for both the backend and the frontend of this application as shown above. The developer used GitHub desktop which simplified the development workflow. Using GitHub desktop, the user could commit and push their workload through visual studio code. Using VS Code and GitHub desktop allowed the developer to make regular commits at the click of a button, making the process a lot more efficient.

7.4 Conclusion

The project management chapter focuses on the overall success of project control and how these goals were fulfilled. The developer discussed each project phase and explained the issues that emerged.

As project management tools, the developer used GitHub and Microsoft Planner. GitHub made is easy to keep track of modifications. Additionally, it allowed for the updating of every version and the cloud storage of both the backend and the frontend.

The Microsoft Planner assisted in keeping track of each task that needed to be completed next, what task the developer was working on, and what task the developer had accomplished.

8 Conclusion

The project's objectives were to implement a pre-purchase car inspection mobile application. The mechanics are qualified to examine and perform diagnostics on the following car that the customer is intending to buy. This provides the consumer piece of mind when purchasing used automobiles since the technicians will assess and inspect the vehicle for you.

The client can interact with the mechanic through messenger, book an appointment with a specific mechanic. The mechanic can upload reports for a specific client. This allows the client to read past and previous reports for different vehicles they have gotten checked. The customer has a choice between two services: Premium and Elite. Each one costs differently and produces various results.

The developer chose to use MERN stack to build this application. MERN stack is a web development framework consisting of the stack of MongoDB, Express.js, React and Node.js. For the frontend, the technology the developer chose was React Native. React native lets developers create truly native app and does not compromise the user's experience. It allows developers to create native apps for Android, iOS and more. (ReactNative)

In the research chapter, the developer explored and determined software development, what is the best software development approach and what is the most suitable framework for building a mobile application. Building a project can raise many dilemmas so research allowed the developer to explore every possibility and decide what path is best for the project. This enabled the developer to conclude that the best technologies for the application were MERN Stack) and the software process (Agile development), and the developer selected native development over hybrid development and react native as the framework.

In the implementation chapter, the developer discusses the application's development environment and goes into great depth about the stages of each sprint, the aim, the difficulties, and the information acquired.

The developer conducted two parts to functional testing, CRUD & navigation. The navigation testing determined how the user and mechanic navigated through the screens. The results helped the developer to improve critical user flows and sharpen the information architecture. The developer also did API testing using Insomnia, this helped to determine any errors or issues within the backend.

The project management chapter discusses process control, the tools that were employed, and how the application benefitted. The tools the developer chose to use were GitHub and Microsoft Planner. These proved critical to the developer's ability to stay on track and structured during each sprint.

8.1.1 Your views on the project

Overall, the project went exceptionally well and has been nothing but good to both my professional and personal development. I have learned a lot of new abilities, but I have also

learned a great deal about myself. It was occasionally really challenging, mentally. The hardest part was to maintain my drive and attention.

Previously, I have worked on projects of this scale within a team setting. Knowing you are disappointing someone else is enough to keep you motivated, keeping yourself motivated is a little harder to accomplish. Nonetheless, I succeeded in achieving my goal.

At the beginning, I genuinely did not think I would get this far, I struggled to even find a concept for my project and now I am at the finish line with what I believe is a great mobile application and with further development could be a great business plan.

The application itself requires some further work to be completed. The backend, I feel, is quite solid; nevertheless, the frontend, while robust, needs further work. The CRUD functionality for the appointments and the messenger between both the client and technician was uncomplete.

Despite that, there is always opportunity for improvement, the overall UI design is one of the greatest aspects of the entire programme, and I am extremely proud of it. Building the UI/UX design became one of the parts I enjoyed the most and is something I would love to continue to develop on throughout my professional development.

8.1.2 Working with a supervisor

Working with a supervisor benefited both me and my application. It became highly reassuring to know that someone was there to guide me through the process.

Knowing you have a weekly meeting with your supervisor keeps you on track since you constantly need to bring something new to the meeting to show you have been working consistently.

Occasionally, before meeting with my supervisor, I was extremely deflated and felt unmotivated. Despite this, afterwards I always felt so much better about the situation and became more motivated to achieve my goal.

The ability to exchange views with someone of such superior knowledge and expertise was also extremely valuable, for which I will be forever highly grateful for.

8.1.3 Technical skills

Using the MERN stack, I developed Pre-purchase Car Inspections, which helped me complete a substantial number of learning objectives. Since I had no prior experience using React Native, creating the application required a significant amount of self-learning. Due to the lack of previous expertise with this technology, this was the component that was the most difficult. If I had chosen a technology, I was familiar with, I believe the process would have gone much smoother, however there would have been obvious challenges.

This programme became possible through online videos, courses but mostly technology documentation. The learning and development of the frontend of this application heavily relies on the React Native documentation and the React navigations docs.

I had minimal experience designing mobile applications, although developing a strong UI to enhance the users experience was extremely critical.

8.1.4 Further competencies and skills

Future development would include concentrating on the components that were not accomplished. The messenger between both the client and the mechanic. Also, after making an appointment, you may continue and pay for that reservation. The CRUD functionality for booking an appointment.

For developing the mobile application further, the developer would need to build their business capability skills. Business skills tend to play a critical role within this field. It is necessary for an app to be able to stand out amongst the competition.

References

Apps Far Outpace Browsers in US Adults' Mobile Time Spent. (2020, July 9). Insider Intelligence. <https://www.insiderintelligence.com/content/the-majority-of-americans-mobile-time-spent-takes-place-in-apps>

API Testing: What It Is, Why It's Important & How to Do It. (2022, February 5). API Testing: What It Is, Why It's Important & How to Do It. <https://blog.hubspot.com/website/api-testing>

Braam, H. V. (2022, December 5). *Navy Blue: Color Code, Meaning, Symbolism and Psychology - Color Psychology.* Color Psychology. <https://www.colorpsychology.org/navy-blue/>

Brewster, C. (n.d.). *13 Types of Software Development | Trio Developers.* 13 Types of Software Development | Trio Developers. <https://www.trio.dev/blog/types-software-development>

Expo. (n.d.). Expo. <https://expo.dev>

How Not to Burn Budget with Agile Scrum | Aristek Systems. (n.d.). How Not to Burn Budget With Agile Scrum | Aristek Systems | Aristek Systems. <https://aristeksystems.com/blog/how-not-to-burn-your-budget-with-agile-scrum/>

Iannace, K. (2021, October 31). *10 Best Hybrid App Examples -.* Designli Blog. <https://designli.co/blog/5-best-hybrid-app-examples/>

Kosmal, J. (2022, November 30). *How does React Native work? Understanding the architecture.* Medium. <https://medium.com/front-end-weekly/how-does-react-native-work-understanding-the-architecture-d9d714e402e0>

Mobile App Download Statistics & Usage Statistics (2023) - BuildFire. (2021, December 20). BuildFire. <https://buildfire.com/app-statistics/>

Mobile Game Market Trends for 2023 You Need to Know - Udonis. (2023, May 4). Udonis Mobile Marketing Agency. <https://www.blog.udonis.co/mobile-marketing/mobile-games/mobile-game-market-trends>

Montaño, D. (2023, February 2). *Why use Flutter: Pros and Cons of Flutter App Development - Waverley.* Waverley. <https://waverleysoftware.com/blog/why-use-flutter-pros-and-cons/>

React Native Pros and Cons [2023 Update]. (n.d.). React Native Pros and Cons [2023 Update]. <https://www.netguru.com/blog/react-native-pros-and-cons>

REST APIs: How They Work and What You Need to Know. (2022, August 30). REST APIs: How They Work and What You Need to Know. <https://blog.hubspot.com/website/what-is-rest-api>

Singh, S. (2023, January 30). *Native vs Hybrid vs Cross Platform - What to Choose in 2023?* Insights - Web and Mobile Development Services and Solutions. <https://www.netsolutions.com/insights/native-vs-hybrid-vs-cross-platform/>

Singh, R. (2022, May 20). *Waterfall Methodology*. Institute of Project Management. <https://www.projectmanagement.ie/blog/waterfall-methodology/>

Team, S. E., Freeman, C., Chakraborty, M., & Iyengar, V. (2017, March 28). *Top 4 software development methodologies*. Application Security Blog. <https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/>

The Good and the Bad of Flutter App Development. (2022, August 4). AltexSoft. <https://www.altexsoft.com/blog/engineering/pros-and-cons-of-flutter-app-development/>

The Model View Controller Pattern – MVC Architecture and Frameworks Explained. (2021, April 19). freeCodeCamp.org. <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>

Toal, R. (2014, July 2). *What Is A JavaScript Framework? | Code Institute*. Code Institute Global. <https://codeinstitute.net/global/blog/javascript-framework/>

Top Mobile App Development Frameworks 2023. (2023, May 3). Top Mobile App Development Frameworks 2023. <https://www.kellton.com/kellton-tech-blog/top-10-mobile-app-development-frameworks-in-2023>

Top 5 CSS Frameworks for Developers and Designers | BrowserStack. (2022, March 25). BrowserStack. <https://browserstack.wpengine.com/guide/top-css-frameworks/>

What is Functional Testing? Types & Examples | Micro Focus. (2023, May 4). What Is Functional Testing? Types & Examples | Micro Focus. <https://www.microfocus.com/en-us/what-is/functional-testing>

What Is Functional Testing? Definition from SearchSoftwareQuality. (2021, October 1). Software Quality. <https://www.techtarget.com/searchsoftwarequality/definition/functional-testing>

What Is Scrum Methodology? & Scrum Project Management. (2022, December 23). Nimblework. <https://www.nimblework.com/agile/scrum-methodology/>

What is software development? | IBM. (n.d.). What Is Software Development? | IBM. <https://www.ibm.com/topics/software-development>

What is System Software? | Simplilearn. (2022, November 14). Simplilearn.com. <https://www.simplilearn.com/tutorials/programming-tutorial/what-is-system-software>

What Software Development Approach to Pick: Native, Hybrid or Cross-Platform? - touchlane. (n.d.). What Software Development Approach to Pick: Native, Hybrid or Cross-Platform? - Touchlane. <https://touchlane.com/blog/software-development-approach-native-hybrid-or-cross-platform>

Wu, W. (2018, January 1). *Theseus: React Native vs Flutter, Cross-platforms mobile application frameworks*. React Native Vs Flutter, Cross-platforms Mobile Application Frameworks - Theseus. <http://www.theseus.fi/handle/10024/146232>

Vietnam, A. (2020, October 18). *Flutter Advantages: 10 Reasons Why Using Flutter For Your Next Project*. Medium. <https://agiletech.medium.com/flutter-advantages-10-reasons-why-using-flutter-for-your-next-project-d8bdd065ada4>

7 Key Benefits of Using the Agile Scrum Methodology / Jile. (2023, January 7). Jile. <https://www.jile.io/blogs/benefits-of-agile-scrum-methodology>

187 Mobile Gaming Statistics for 2023: The Ultimate List / Udonis. (2023, May 4). Udonis Mobile Marketing Agency. <https://www.blog.udonis.co/mobile-marketing/mobile-games/mobile-gaming-statistics#:~:text=As%20of%202023%2C%20there%20are,200%2C000%20on%20Apple%20App%20Store.>