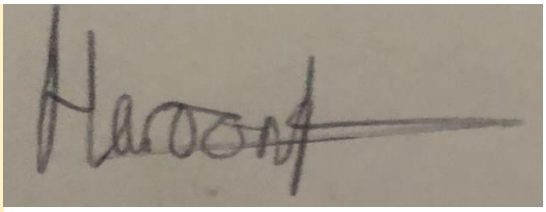
HUBCAR

Ahmad, Haroon 14

3814324 ahmadh14@lsbu.ac.uk

Declaration

"This dissertation is my own original work and has not been submitted elsewhere in fulfilment of the requirements of this or any other award. Any passages taken from my own previous work or other people's work have been quoted and acknowledged by clear referencing to author, source and page(s). Any non-original illustrations are also referenced. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this dissertation and the degree as a whole."

Signature: 

Abstract

This paper documents a third-year computer science project from idea to conclusion. HubCar, a native mobile application, was developed as part of this project. It is the goal of this project to teach new drivers and novice motorists about car maintenance. The main technology that was used for the implementation of this project was a java programming language. Android studio was used as the working environment throughout the project. The report was structured with first an introduction to the project aims, and requirements, and looking at similar works, it then moves on to a technical discussion about design, solution, and implementation. It discusses the main aspects of the project, the decisions that were made throughout, problems that were faced and what was done to resolve these problems. Lastly, the paper covers the difficulties encountered in this endeavour and what was learnt and what could be done to better. Another element of this project's work that can be implemented in the future is based on what has already been done.

Acknowledgement

Throughout this project, I received great support from those around me. I would first like to say thanks to my supervisor, Professor A. Muhammad without his support and assistance as I would not have been able to have completed this project without his knowledge.

I would also like to thank my friends who have been there for me throughout this project and checking the work to make sure it is up to a high standard and being there for me during the lows and highs.

Finally, I would like to thank my family I can't begin to express my appreciation for them for their incredible guidance and loving and supportive nature.

Table of Contents

Table of Figures

Table of Tables

Introduction

Overview

This report will document the creation of the mobile application HubCar and will provide an in-depth description of how it was completed. The report shows the problem that I am going to solve and will provide a detailed description of the design and implementation of the project the challenges I will have to overcome and any improvements that need to be added once the application has been developed. The report will also have an analysis of the technologies I used with reason and will be compared to others.

I chose to do this project because this problem I have faced personally applies to many people, especially new drivers and people who have little or no knowledge about cars. But with the knowledge, I acquired from software engineering and from my first web application I wanted to create a mobile application that will help generations. The app will create a community

Although I have done not have any experience in building mobile applications and have little knowledge on how to do so through the development of this project, I will learn how to expand my knowledge and create an application that has all the features and requirements that I want to add.

Project aim

The project aims to create an android mobile application called Hubcar. The app will be used to assist users in finding a mechanic in their area. The app focuses on solving two important problems. The first is providing users with a distribution management system for cars and a function that will show the user nearby garages and their ratings. The project aims to improve the interactions between users and Garages. The project will be compared to the likes of Uber eats and Deliveroo but will be faced solely with parts and services for cars as this issue has not been tackled before. It will feature a buying system for users just as seen on the food apps but will also feature a booking system where users can book services for their cars which currently isn't plausible unless getting directly in contact with a garage or mechanic.

Objectives

The following are the key objectives that I want to complete and are necessary for this project.

1. Learn java
   Although I already have previous knowledge of java from my 2 years of studying computer science, I still need to learn some functions which will be necessary for the development of a mobile application.

2. Learn the android studio
   Android studio is the official IDE for the Android operating system and will be used in the development of the application.

3. Design and implement the user interface
   This is the front end of the application and what the user will be interacting with.

4. Design, develop and implement pages for the application.
   This includes a log-in page, sign-up page, shopping page, map and location, couriers' profile, user profile, and payment page.

5. Implement security measures.
   With the application, the user will be inputting sensitive information. I will need to research and implement the necessary security measures to ensure the information being imputed into the application is kept safe. I will describe the security measure being taken more in-depth later in the report.

6. Learn about APIs
   I have never had experience previously with using APIs but in this project, I plan to learn and understand how to use postman APIs

7. Implement a service booking page this service will only be available to registered users so that they can book a service appointment for their cars

Objectives limited by time

1. Adding a google pay function
2. Adding a page that users can use to identify dash light codes
3. Adding a page that can link users to a video or website on how to do certain tasks

Scope

For this project, I aim to improve my creativity and develop skills in the java programming language. I will also learn and become familiar with Heroku and how to host my application on it. I will then learn more about APIs and look in-depth into them and once I understand them, I will become familiar with postman APIs. With the development of the project, I will then learn how to use android studio, so I can launch the application on a mobile phone. The aim in general is with the all the skills developed and enhanced throughout the process I will create a mobile application like that of uber eats and Deliveroo which will enable users to have more direct and easier interaction with garages and mechanics.

Constraints

The following are the constraints that will be faced during the project due to external factors which I am unable to control:

- The First constraint is time. Due to this project having a set time to be completed some of the more complex features I want to add to the project will only be added if I have time what I will focus on is the more vital features that are required.
- Second is a cost this project is cost-free I will not be spending any money on it so any features that require payment, for example, hosting the application cannot be done due to insufficient funds.

- There are also some hardware constraints due to not having a MacBook I am unable to use some development environments such as XCode and swift which are exclusively based on apple products.
- I cannot make the application completely as I do not have a development team

literature review

The literature review section will be explaining the problem that I am trying to solve and will expand on the research that I have conducted to overcome the problem at hand. In this section, I will be discussing similar applications to mine which offer services that my application also has but for a different product and how it has impacted a user's life either by making a task easier for them or solving a problem.

Background

For many businesses, internet food delivery is a significant means of communicating with customers and generating sales. Because of the emergence of Covid 19, this point was particularly visible during the lockdown that happened in 2020, with internet food delivery being given credit for helping many food businesses to survive (archive.ph, 2020a). My application would help with problems like the pandemic for mechanics and garages as there isn't a dedicated application developed yet for these types of businesses and although grades were open businesses who rely on selling car parts and services wouldn't have been able to open their businesses, this is an issue my application would solve.

In regions where the online food delivery or delivery industry is well developed, these businesses have realized that they save costs associated with the provision of space and that this space can be used to provide more room for their expanding online delivery services. The ultimate manifestation of this trend is the development of so-called ghost kitchens which have become relatively common in the UK, US, and India (archive.ph, 2020b). These food delivery businesses do not have a physical storefront at all, and online orders are their only source of income. Dispensing with a physical storefront has many advantages for restaurants including a reduction in the cost associated with premises, reception and wait staff, the ability to increase the diversity of menus, concepts or even brands, the ability to run multiple websites and to provide a diversity of dining experiences catered for virtually limitlessly by one kitchen (hanet, 2020). In addition, such kitchens can capitalize on the advantages of scale enabling them to invest more in streamlining delivery management thereby enabling them to get the food to the consumer in a more timely and cost-effective manner. This can also be applied to my application as the application has an eCommerce section if a business wants to sign up where they solely distribute and sell car parts, they can do that and now have a platform to help attract consumers and expand their business. The same point goes toward the garage business delivery industry helps promote these businesses on a wider platform even if they do have a physical shop.

Delivery workers are among the numerous people who have found work in the online Food delivery business because of the industry's rapid growth. Meituan and Eleme, two of China's largest Food delivery platforms, employ 1.17 million people in China to operate as delivery persons. A similar number of individuals are employed by Swiggy in India (17 thousand delivery people) (sachitanand 2020) and Uber Eats in the United States (over ten thousand employees) (archive.ph, 2020b). This in addition has a positive effect on the economy. Therefore, my application would be necessary for delivery people it will provide more jobs which is currently vital due to the covid 19 pandemic that the world has faced and will provide more delivery drivers on another platform opening more jobs.

The application is needed for end-users as this application doesn't have any competition on the market. There are already delivery services as shown before and all have great success as it makes a user's life easier. However, the need for my application is it will enter another market that hasn't been utilised yet. With mechanics and garages, they are mostly in person or on the phone calls and to buy such parts usually requires a lot of research online to find them or having to go to stores. The application will combat the issue previously faced by food delivery services and as stated above will provide more jobs and business during such a crucial time. The application makes car servicing and knowledge simpler and easier for those who do not know about their vehicles or where to start with their vehicles.

Similar Applications

This project does not have any applications that are like it. However, the functions that it does have other applications do have, such as the delivery service my application will feature, which have applications that are targeted and dedicated to this, such as UberEATS, Postmates, and Deliveroo.

When it comes to applications based on finding mechanics and garages from the searches, I did on the play store and app store, there weren't any that I could find that would help a user.

Some applications can help identify warning codes a user might be met with; however, these applications usually charge the user, and the application is only for a certain make of car, for example, BMW.

Postmates

In 2011, the AngelPad incubator launched Postmates. Customers can utilise a smartphone app to place orders from any participating restaurant or their home or office computer; no internet connection is required. Since its inception in 2010, they've grown rapidly in more than 40 major metropolitan areas. The company's mission statement reads: "On-demand delivery service Postmates is reshaping the way goods move across cities. With our new Urban Logistics platform, customers can quickly and easily order from any business or restaurant in the city and have it delivered right to their door. Using our API, customers can purchase locally without having to stand in line, and businesses can offer delivery." (Postmates.com, 2019) the website of Postmates local businesses that don't own restaurants, which account for the lion's share of the delivery market, have a sizable untapped market.

"Postmates quickly reached high valuations after their initial financings between 2011 and 2013," states the business. Every order includes a delivery fee that the customer must pay. Price may vary depending on the type of restaurant and the distance between the pickup place and the customer's location. This service, which charges $3.99 for delivery within a set radius of the restaurant, is known as "Postmates Plus." "Special partner restaurants" refer to restaurants that participate. To enjoy free delivery on purchases of $25 or more, users must pay a $25 monthly subscription fee to Postmates Plus Unlimited. Subscriptions to this programme, like Amazon's "Amazon Prime," encourage customers to spend more because they don't have to pay postage on any orders.

Many apps are dedicated to one of the features of my application as shown above post mates which is very popular in the United States popular ones in Britain include amazon, uber eats, just eat Deliveroo, etc. all of these provide simple delivery service through an application which provides the user with any service and not only attracts business for smaller companies also provides a wide range of jobs. My application has the same features as these applications with placing orders, seeing

when the delivery will arrive, but offers the extra features for the knowledge of cars which another application has not done before and applications do not exist.

Technical review

This section of the report discusses the differences that I will be using for my project. As there are many technologies available for me to use for the development of my project, I need to ensure I am selecting the correct ones. The section begins with a look into the different app development environments and then moves on to the databases that could be used for the project. And then to finish this section, it will talk about testing where it will talk about the automated and manual testing for the technologies that were discussed throughout. As there are many technologies to choose from and discussion and comparison are needed to make this choice.

Web apps vs native apps vs hybrid app

Mobile development means creating an application or creating software that is intended for use on smart devices such as phones or smartwatches. In mobile application development, there are many technologies to choose from so a discussion on the advantages and disadvantages needs to be made to select the correct technologies.

The options that are most used in mobile application development are web applications, native applications, and hybrid applications. In this different development, there are different options to choose from each has its reasons for and against. In the next section, a description of each application will be made, and their advantages and disadvantages will be listed

Web applications

Web applications are applications that have been created with the use of web technologies such as HTML and CSS. A browser such as google chrome and an internet connection is all that are needed. To host them all is needed a hosting service such as a windows server however the difference with them is they are made to look like how they would on a mobile phone and the way they would function. However, they are still able to work on anything that has a browser. The experience that a user can gain from hosting a server is immense however if the user's server is slow, it means that the loading will be slow and hurt the user's experience. However, a benefit of having your server is that your application can be updated at any time, and it saves users from having to manually update to the latest version.

Other drawbacks of web applications are they are slower than native applications and will negatively affect the user's experience due to browser interaction. Users will face a loss in features such as swiping, features like this are available for web apps however they don't have some support as they would on a native application. Web applications generally don't need features like these however if they are needed it would be a better option to go for a native application format

With the testing aspect of web applications, it takes a lot more time to do so as you must test the application on multiple web browsers and different versions just to ensure that it can be used by every user, and just to get certain functionality a lot more code needs to be written.

Native applications

Native development is where software can only be used on a certain device and can only perform a certain function. Although this can be seen as a drawback it means that it could be the fastest compared to the others. Due to native applications only being used on certain devices it means they

can run faster and smoother and can utilise every feature on that device, for example, camera or microphone. Native apps can operate in the absence of an internet connection, which is useful for some applications with specific features. All the tools and functionality that come with the device are available to native applications.

When the user uses the function available on the application, they will be able to quickly get how the application functions as they would have previously used such feature on other applications already on the device. It is possible to run an application on one of two main platforms. Using Google Play for Android and the App Store for iOS Over 97% of this market belongs to them. (Mobile Operating System Market Share Worldwide.)

To make android-based apps the main programming languages are Java and kotlin, both are very similar as kotlin is based on java however it is more efficient than java as it can require fewer lines to complete a task and kotlin is known to have a cleaner and shorter syntax compared to java. However, this doesn't necessarily mean kotlin is better than java. (Mobile Operating System Market Share Worldwide.)

Hybrid applications

The final option is a hybrid application which is the newest of the three. A hybrid application is essentially a combination of native and web applications and uses the same code base for any platform. To create a hybrid application a web technology is needed to achieve this and then once completed is wrapped in a container through a framework such as react.

Developing a hybrid application is a much cheaper option than native applications. This is because the written code can work for platforms and getting developers for hybrid applications is cheaper than for native applications. However, the cost for a hybrid application can sometimes go up due to fees which then can amount to more than a native application. These fees can be having to get developers and programmers to work more for coding or when testing the user interface design. This obviously won't apply to all projects however is a disadvantage to think about. Another disadvantage of hybrid applications is that their performance is much lower compared to native applications

A drawback that is faced during the development is cross-platform coding which can sometimes be a challenge. This will all depend on how complex the application is and how much you want it to simulate a native user's experience, and to access certain functionalities extra help is needed as their and lack of available resources. The user's experience will also be negatively affected as they will be able to fully utilise the native only features for the best user experience, this is due to different factors such as screen layouts and functionality.

The component of the code that is dedicated to the user experience and user interface Every developer should think about it. Each platform and manufacturer require a different set of features. For example, pressing a back button or using features that are different for each platform. There are frameworks and APIs that can help with this, but you'll still need to write your code.

With every mobile application, updating is a big part. As hybrid applications have the same source code, they do not need to be updated however the issue arises when the operating system has an update. A native application developer has access to early beta versions which allows them to then update their apps with the latest features and then can update their application alongside the update of the operating system. This feature cannot be done with hybrid applications. Developers

would first have to wait for the release of the update on the operating system and would have to find resources from a third party to add the features they want or code it themselves. Once they have achieved this only then can they update their application?

On the whole hybrid applications are good depending on the project they are being used for. I would have liked to create a hybrid application and made my project available for all platforms. However, through doing more research and testing I found that the best option for me would be to create a native application. The reason for this decision is that I believe that the user experience and performance is the biggest thing to focus on. A major reason why hybrid apps are chosen over native apps is solely due to them being cheaper however with this project there is a budget or limitation. As stated in the arguments and points above native applications are the best option for me to achieve creating my application.

Back-end

All mobile applications and almost every software project is comprised of 2 parts the front end and the back end of an application.

The front of the application oversees the user interface and user experience whereas the backend of the application will be in control of the security, data storage and business logic. The backend of an application almost acts like a server because it will store and manage information that the user will not see. Not every application will need to store data however in the case of my application it will need to store user information and other information needed for the functionalities. Every piece of software should be kept up to date and changed over time by the engineer. Another thing that may happen is that some data will be used for analysis and studies by the organisation or by other people. There aren't many chances to make an app's back end.

For my application, I could have used the phone’s local storage however my future with this project is to make it available for multiple platforms, and another reason for this methodology not being chosen is that the data that has been saved can only be used locally and nowhere else.

With the choice of servers, there are three options custom server, backend-as-a-service and cloud server.

A custom server is a fully customized service that is built specifically for your mobile application. In the case of my project, however, a custom server is not necessary as the functionalities I am adding to my project are simple and can be managed through other options. It also takes up a lot of my time as I would need to get experienced with it which time is limited and would require a lot more effort to understand.

Backend-as-a-service provides web and mobile developers the skills to connect the application to a backend cloud storage and APIs. At the same time, it will provide features such as user management and authentication.

Cloud servers are pre-built and ready to be used, these can be customized however they cannot be fully customized they have a limit. A reason why these are chosen is due to low cost, the range in sizes and they are secure, examples of these would be the google app engine.

Final decision

For my project, the application will use android studio and java programming language on the backend code. I chose to develop an android application over IOS due to the reasons stated above. I do plan to, later, build an IOS based application. The reason I chose to use java is I do have previous experience with it through university however I did need to enhance my skills and develop my knowledge, and due to the fact java has many available resources online and a large community compared to that of kotlin which is new and doesn't have the same resources as java.

Android studio is the IDE I chose for the development of my application. I chose it as it is googling official IDE for android. It is also very user friendly and easy to understand and grasp as it has a drag and drop layouts, you don't have to restart your application every time just to add changes and it supports auto-completion. Android already comes with an emulator built-in which can allow you to choose between different smartphones, watches, or tablets to test your mobile application another great use for the emulator is you can apply changes without having to restart your application. Out of the IDEs I have previously used such as NetBeans I have found an android studio to be the most suitable for my project

I have decided to go for the backend-as-a-service option as I find for my project it is the most suitable for the backend portion of my application. the reason for choosing this option is due to the fact it will help decrease the development time, which is needed due to the deadline, standardised coding environment and reduces boilerplate code. Other advantages are that it has a high performance and is very secure. The most popular options would be firebase, parse server or Heroku. All three are great options however for the project Heroku was chosen. The Heroku platform has a set of features that add more value. Using Heroku, you don't have to learn about how to set up a server, how to manage a network, or how to improve a database. It makes it easier for developers to make the app faster because Heroku takes away barriers.

Heroku is a cloud service platform that has become more popular in the last few years. So many people choose Heroku because it's so easy to use. It makes it easy to make and put up apps. Because the Heroku platform takes care of hardware and servers, developers say they can spend more time working on their apps when they use Heroku. And not the infrastructure that helps them work. More time is spent making sure that users have the best possible experiences.

Methodology

In this chapter, the methodology for software development is discussed. This chapter will state the different methodologies and have a description of each one first and then a comparison of all the available methodologies. Once each one has been discussed and compared the chapter goes into detail about the chosen methodology and the concepts that will be implemented. There are three software development methodologies to choose from and each one has its advantages and disadvantages. The question that needs to be answered is which one is the best. To answer this, it comes down to the project at hand whether is it a team-based project, are you engaging with the client throughout the project and does your project has a budget. A methodology is rarely fully followed in practice; instead, the best practices from several methodologies are combined based on the specifics of a given project. This, too, is true. During the development phase, I plan to mix many ideas. However, before we get started, let's take a quick look at the most popular approaches.

The main methodologies for software development are the waterfall and agile methodologies the approaches and quite different, however. Their differences lie first in each one has a different framework; each one is suitable for a different type of project. Agile has multiple frameworks such as extreme programming, prototype framework and rapid application development. This section will now go into detail about the one picked for this project. It will begin with a definition of each methodology and a description then an explanation of each of their components and how they can be applied to the project

Waterfall model

An example of the waterfall model is when a project is broken down into successive phases, each of which is dependent on the previous phase's output. The method is common in some engineering design fields. Software development is typically considered to be one of the less progressive and adjustable methodologies, as progress is primarily in one line through the phases of conception, initiation, analysis, design, construction, testing, deployment, and maintenance, with little room for improvement.

Due to the reason stated I will not be choosing this methodology as it would mean the project needs to have clear and fixed requirements which now it does not. Most likely throughout the project, the requirements will change throughout the development and the use of this methodology will cause me to lose time which is scarce as each step would need to be done from the beginning which is a lot of effort as well.

Agile software development

In agile software development, requirements and solutions are continually iterated by self-organizing, cross-functional teams in collaboration with their customers. It promotes a faster and more flexible response to the changes, as well as adaptive planning, developmental growth, early delivery, and continuous improvement.
The agile methodology is seen to mostly value individuals and their interactions over the processes and their tools. The working software over the documentation process has good customer collaboration and interaction and will accommodate and respond to change rather than follow the plan they have been given.

Prototype methodology

Software prototyping is the process of producing working but unfinished versions of a software programme to test ideas and gather feedback. Prototypes are often limited to a few characteristics of the final product and may even be altogether different.

Scrum framework

People can solve complicated adaptive challenges inside the Scrum framework while still producing high-quality solutions that provide value to their customers' lives. Scrum is preferred and chosen as it is simple to understand however is not preferred in some cases as it is difficult to master which requires time and effort. The Scrum framework is comprised of Scrum Teams and the roles, events, artefacts, and rules that relate to them. Each component of the framework performs a distinct function and is critical to the success and widespread adoption of Scrum.

Methodology chosen

Due to my project not having well-defined requirements it does not make it a suitable option for the waterfall model. It also wasn't suitable for rapid agile development meant due to the same reasons as the waterfall model, it needs defined requirements, and it makes customer involvement an important step from the beginning of designing the application through to the development of the application. Other methodologies also require customer involvement however it is not as important as it is for rapid application development.

Looking at the available methodologies the one I have opted to go for is the scrum framework along with prototype methodology. The reason for this decision will be described by first giving more detail about each framework and then how I will be applying them in the software development process.

There are three main areas for the scrum framework: scrum team, scrum events, and scrum artefacts.

The scrum team composes of a team as stated in the name it will have the owner of the product, the development team and the scrum master. This would be the best option however in the case of my application there is only one person who must perform all these roles. Scrum events are compiled from different components such as the sprint, sprint planning,

The main aspect of scrum is a sprint, which is where there is a time from 1 week to a month where a completed and usable and could even have product can be released in increments. Through the development process, sprints will have sprint will always have a consistent time frame. Once a sprint has been completed a new one will start straight away. Each sprint is comprised of sprint planning,

daily scrums, development, sprint review, and sprint retrospective. Sprints are usually 2 weeks long and are created in an hour-long allocated time.

With my project, I do not have daily meetings with a team but just a weekly meeting with a supervisor instead for feedback and to keep track of my progress and tasks, I used Jira to measure this.

Product backlog, Sprint backlog, and Increment are all part of the Sprint Artefact. All of the features that will be included in the final product are listed in the Product Backlog. It's the only place to look for product specifications if something needs to be changed.

The criteria for my project were defined in the form of user stories. The sprint backlogs are used to prioritise the most important user stories from the product backlog. The backlog contains all of the user stories from the section that follows

Scrum was designed to concentrate on delivering the increment, which is delivering a completely functional piece of software at the end of each sprint. During a Sprint, the Increment is the sum of all Product Backlog items completed during the Sprint as well as the value of all prior Sprint increments.

If there is a sprint that hasn’t been completed successfully or hasn’t been completed on time, a review of the sprint needs to be made and then needs to be sent to the backlog.

I chose to use this methodology for these reasons as requirements are not the final ones and are open to having changed and with the use of this framework, these changes can be applied with eases.

One of Scrum's primary benefits is the speed with which it generates outcomes. The Increment lets us have a working product at the end of each sprint, as previously stated. Requirements, functionality, and design can all be revised using this product.

Plan of implementation

To achieve a common goal or objective, a team should develop a plan of implementation that outlines the steps they should take. This plan will include all aspects of the project and will combine strategy, process, and action to achieve success.

Organization

For the organization of my project I have used 2 different products to help me plan how the 20 weeks to do the project will be played out, I first used a Gantt chart which has 10 different categories: research, introduction, literature review, technical review, methodology, design, implementation, evaluation, finial research and reflection, and will show how much time I plan to allocate each section from week 1 to week 20 based on how much time I believe each section needs. The next product I used is Jira which will have every task I need to complete for both the report of the project and the development of the software for my project.

Gantt chart

To make the Gantt chart I used Microsoft Excel to create I used it as it isn't time-consuming, I don't have to learn anything and will illustrate how I plan to use my time for the report section of the project.

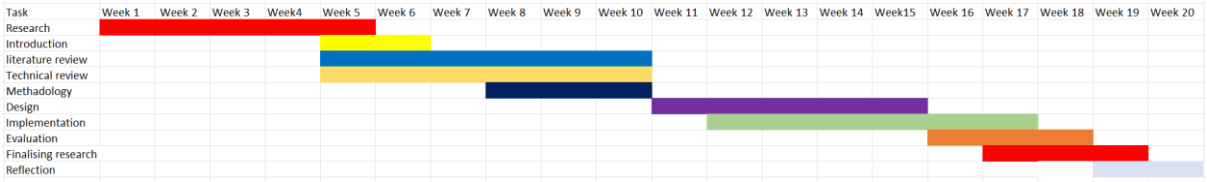


Figure 1

For research, I have allocated 5 weeks I have done this as stated in my introduction and requirements to use some products I need to learn to use them first for example with a programming language I first needed to decide which to use which I concluded to be java and to use java I needed to enhance my knowledge and develop it which is why I have chosen 5 weeks. For other software such as postman API, I also need time to learn as I have never used an API before.

For the introduction, I have only allocated 2 weeks from week 5 to week 6 as this section doesn't require much time as it will just be stating the problem the project I am creating is going to address and what the project is and will just lay out the aims.

For the literature and technical review, I have both given them the same time frame of 5 weeks from week 5 to week 10. I have done this as both these sections can be done simultaneously in this time frame as the literature review is me making the argument that my project is worthwhile and supporting each argument I have made with a reference. The technical review also needs the same amount of time as this section as here I will be comparing the languages and software I am using for my project and have already decided before during research however will explain why I've chosen them and a small comparison to the others.

For methodology I have chosen 3 weeks from week 8 to 11 I believe this may take less time however for research and options I have chosen 3. In this section I have spoken about the methodology I have chosen and how it will be beneficial to the project and how I will make use of them in practice.

In the design section, I have given a time frame of 5 weeks this will be where I will design how my application will look. I have given it this much time because I will need to choose which UI design tool to use and whether it will be high fidelity or low fidelity. For the report, I will be choosing to use balsamic which is a low fidelity UI design.

For implementation, u have given it the longest time frame of 6 weeks from week 12 to week 17 I have done this as implementation will require the completion of the application with screenshots explaining each one and then the code to explain how it will work.

Evaluation I have given a time frame of 3 weeks from week 16 to 18 as

Finishing research I have given 3 weeks

Conclusion I have given a time frame of 2 weeks

Jira

Below are the screenshots for all the tasks I had to do and placed on Jira. Task 1 to Task 35 focuses on the first area of the report up to the design section of the application. Tasks 36 to 169 are all the tasks that were based on the development of the application including both the front end and back end. Finally, tasks 169 to task 185 are the second part of the report from implementation to a conclusion. All the task for the report is also in here each heading I created was noted. The only task not in the Jira is the early stages of research

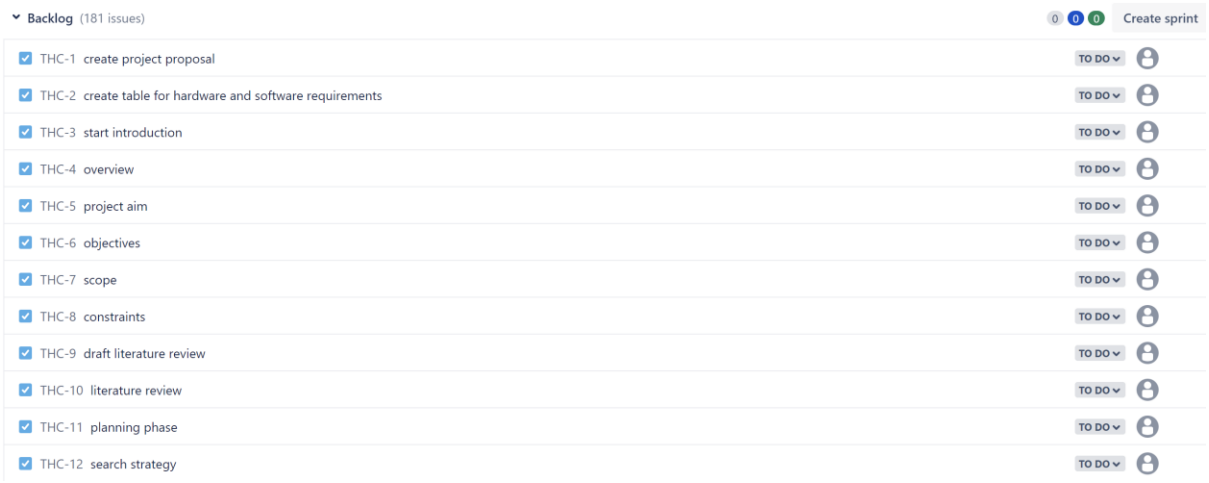


Figure 2

Figure 3

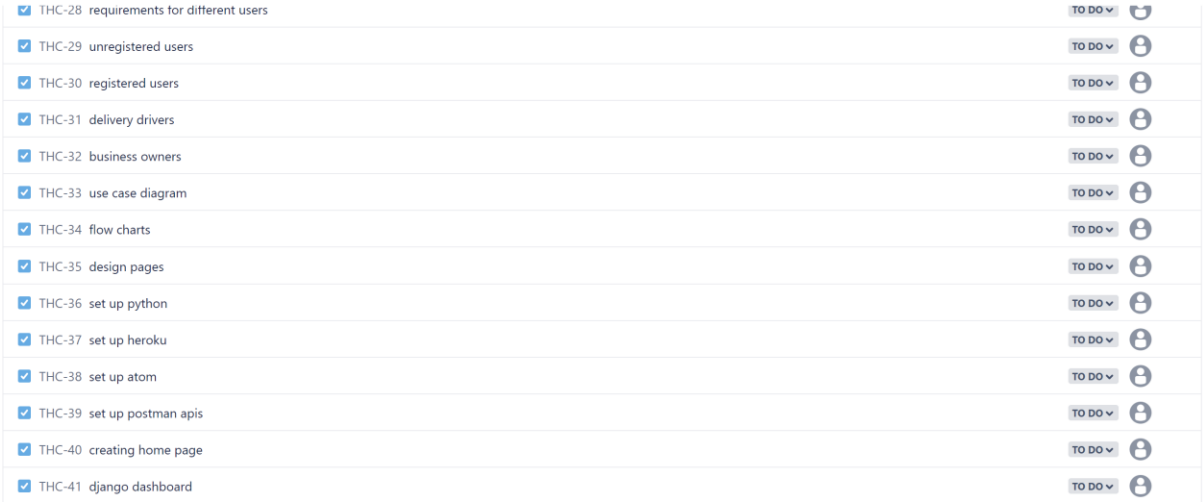


Figure 4

Figure 5

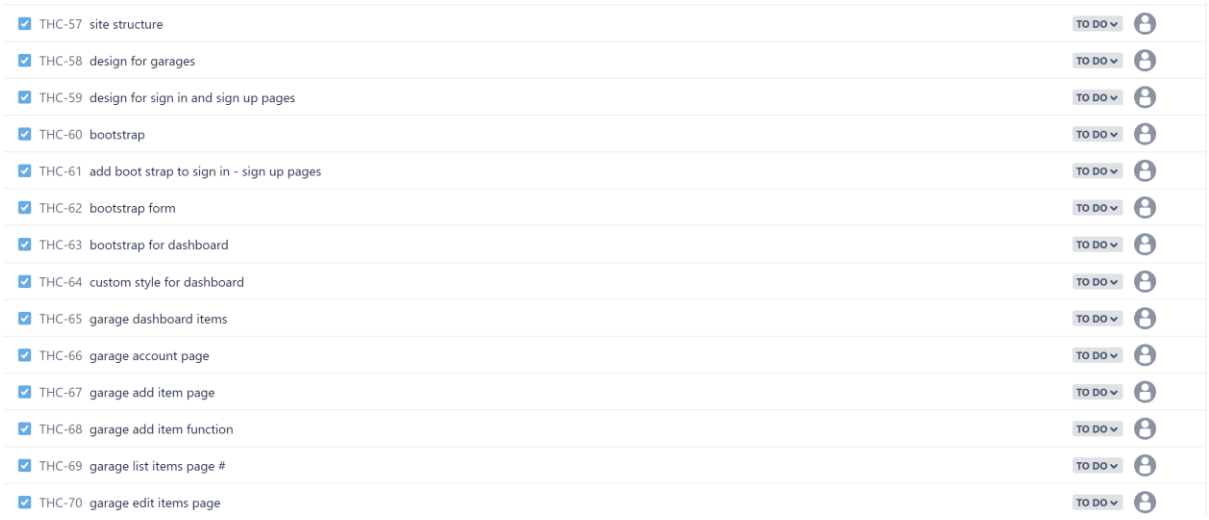


Figure 6

Figure 7

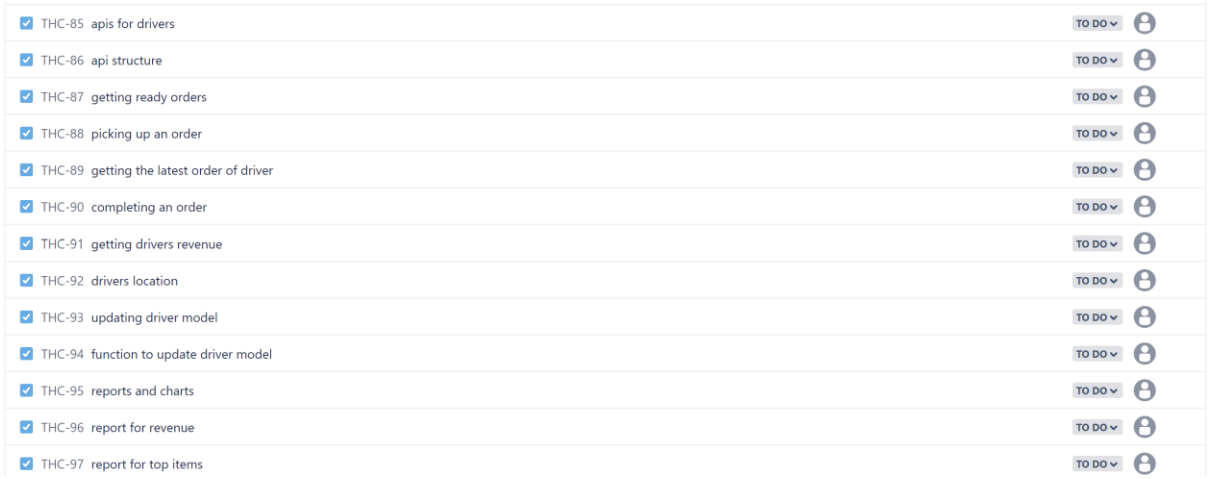


Figure 8

Figure 9

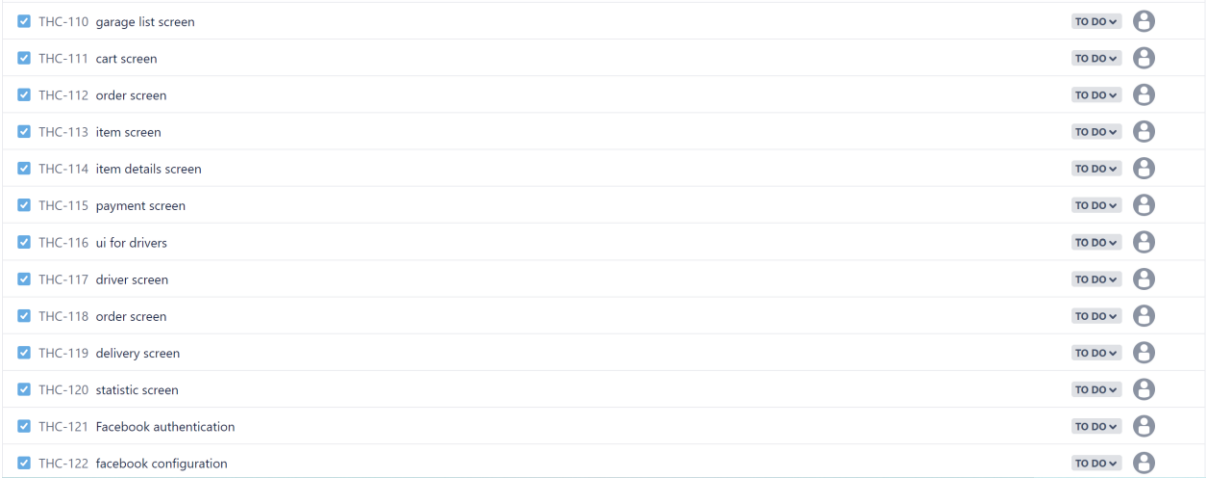


Figure 10

Figure 11

Figure 12

Figure 13

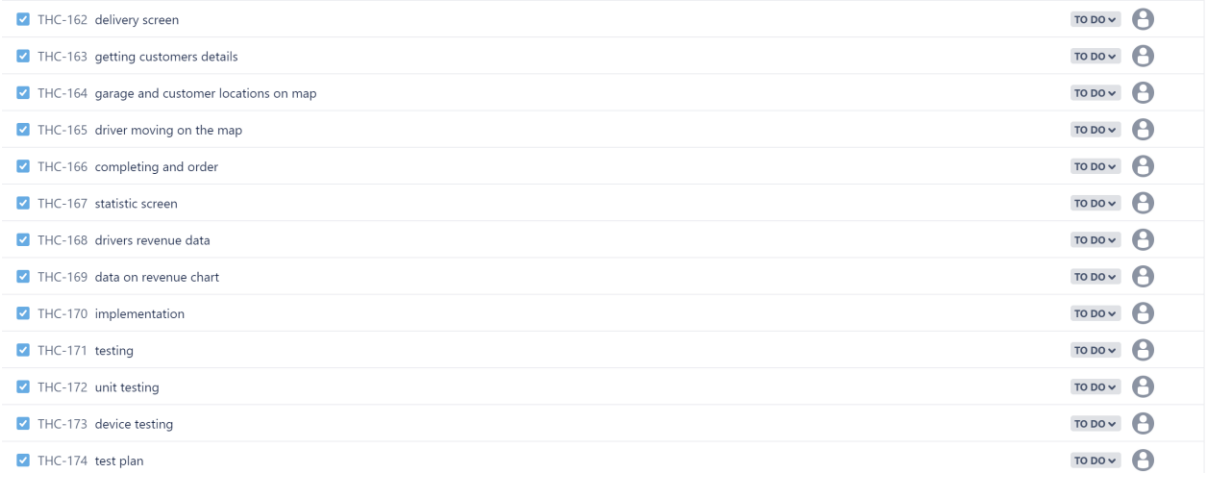


Figure 14

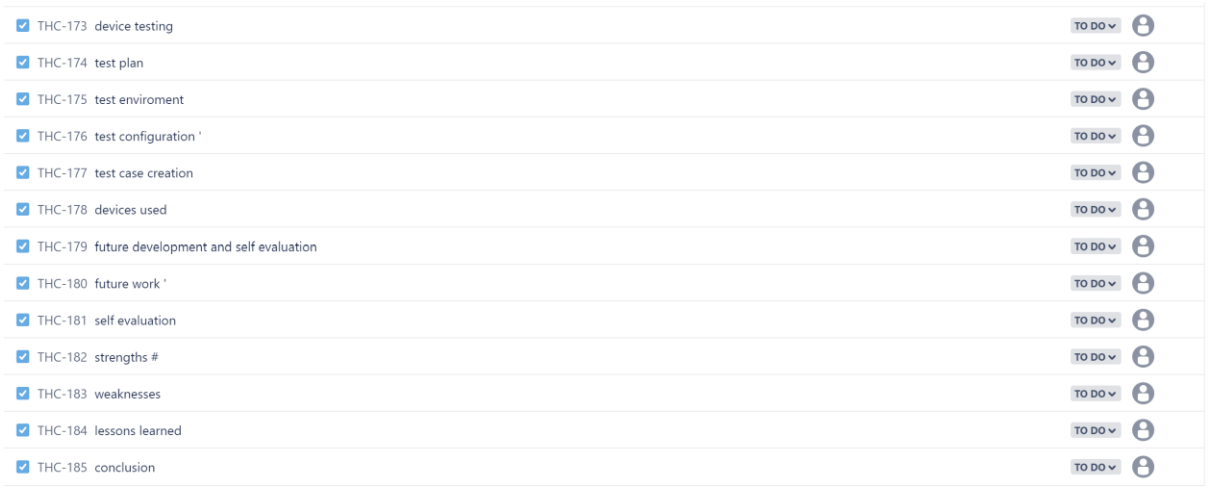

THC-173 device testing
TO DO
THC-174 test plan
TO DO
THC-175 test enviroment
TO DO
THC-176 test configuration '
TO DO
THC-177 test case creation
TO DO
THC-178 devices used
TO DO
THC-179 future development and self evaluation
TO DO
THC-180 future work '
TO DO
THC-181 self evaluation
TO DO
THC-182 strengths #
TO DO
THC-183 weaknesses
TO DO
THC-184 lessons learned
TO DO
THC-185 conclusion
TO DO

Figure 15

Requirements and design

Requirements

In the requirements section, I will be using different diagrams to discuss the system requirements. A use case diagram will be used to show how the requirements will work. As I am completing the project myself there is no client that I can use to obtain requirements, a lot of the requirements are based on what I would like to be featured on the app and how it will help potential clients. Some of the requirements may exceed the scope of the project and may not be able to be completed due to the time limit. But these requirements will still be added to see how the app would function with more time and be fully completed.

Requirements for different users

Unregistered users

User story: As an unregistered user…	Acceptance criteria
I can view the different mechanics around my area	To achieve this the application will have a feature that allows the user to enter their postcode and bring the application to the area
I can see the rating for different mechanics in my area	When a user has made a purchase or used a service, they will have the option to leave feedback which will feature a 5-star rating that will accumulate an overall rating based on the users who have left one
I can see pricing for different services	Price will be featured on each product that is featured on the product

Table 1

Registered users

User story: As a registered user…	Acceptance criteria
I can leave a rating on a service based on my experience	Users can leave feedback once they have used a service
I can input my location and be presented with locations near my area	Users will have a bar where they input their location nearby results appear
I can input personal information and make bookings using my details	Users will have already inputted their details when they sign up
I can make payments using a credit card or google pay	When users click to check out, they will have different methods of pay
I receive notifications for my purchases	A notification will be sent when a purchase has been completed
I can purchase multiple products	browsing users can add multiple items to their cart
I receive a confirmation on any bookings I make	Users will receive a notification anytime they use the booking system as confirmation
I can filter the categories of what I am searching for	Users can use the category section to filter for a specific service or product
I can use a search bar to browse	A search bar will be used to search for various services

I can purchase a product and choose to either collect it or have it delivered	Users will have the option of whether to collect or deliver a product
I can view where my package is once I have made a purchase	The user will have a delivery page that will show the location of their order
I can stay signed in once I have signed up	Once the user has signed up, they will remain logged in till they sign out

Table 2

Delivery driver

User story	Acceptance criteria
I can input my location and receive only jobs based in the area	Once the driver goes online, they will receive jobs only in the area they are in
I can choose to reject or accept a certain job	Drivers will be able to accept or reject a job
I can view my past deliveries and current job	Drivers can go on their page and view the deliveries they have made
I can see a direction on maps to a user's location	When the job is accepted drivers will receive directions to the user's location
I can choose when to start and finish	Drivers can choose when to go online and when to go offline

Table 3

Business owner

User story	Acceptance criteria
I can view when an order comes through	Businesses will see when a user has made an order
I can view how much I have accumulated	Businesses have a page where they can see the orders and how much they have accumulated
I can view previous orders	All previous orders to the business can be viewed
I can view a customer's profile when they have purchased	When a purchase has been made the order can be viewed
I can view when a driver is arriving for pickup	The business will be alerted when a driver is arriving for pickup

Table 4

Use case diagram

This section will show the use case diagrams for each user and the relationship between the users. To create these diagrams visual paradigm will be used as it is free and has been previously used in other projects.

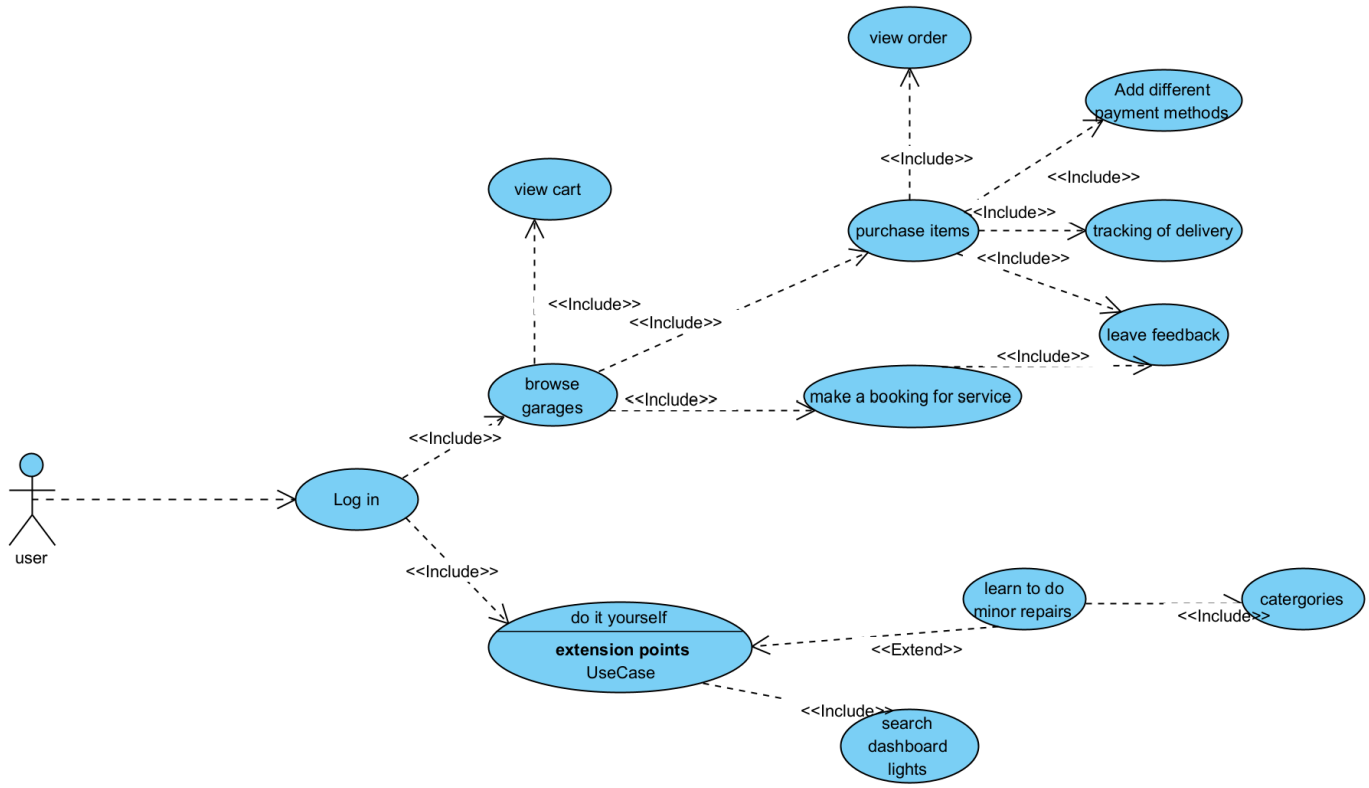


Figure 16

Figure 16 represents a registered user and represents all the features they have access to. Only if they sign up can they access this. When the user logs in they'll have a tab bar where they can select to go to different pages. The user will always be brought to the browsing section where they can view garages and either purchase an item or make a booking. If the user puts an item in the basket and doesn't purchase it, they can always return it to the cart through the menu. If the user goes through with the purchase, they will have a payment method screen and once that has gone through, they can track the item and once it has been delivered, they can leave feedback. If the user closes the application before the delivery, they can open the application and view the order through the tab menu. Through the menu, they are also able to view the do it your self-section where they can select the issue and be presented with a solution.

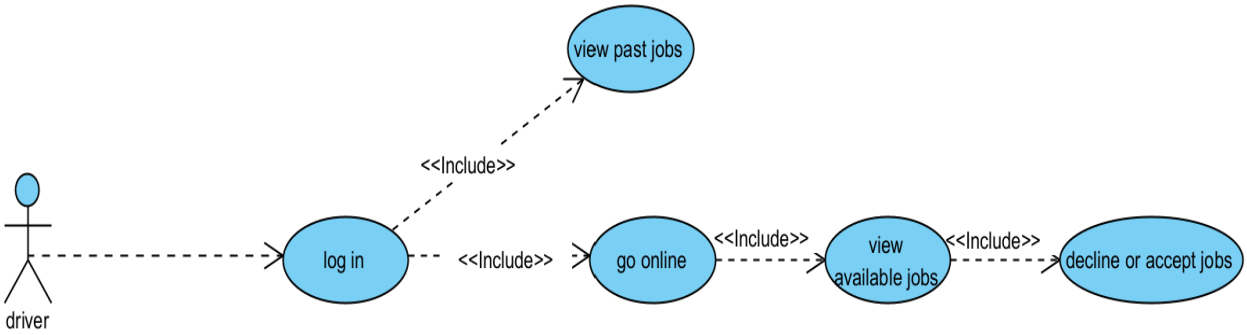


Figure 17

Figure 17 shows the delivery driver's process described here. It includes everything the driver can do once logged in he will be able to view his past orders see which jobs are available and accept or decline them this process will happen every time a driver logs in. he also can see the past deliveries he has done.

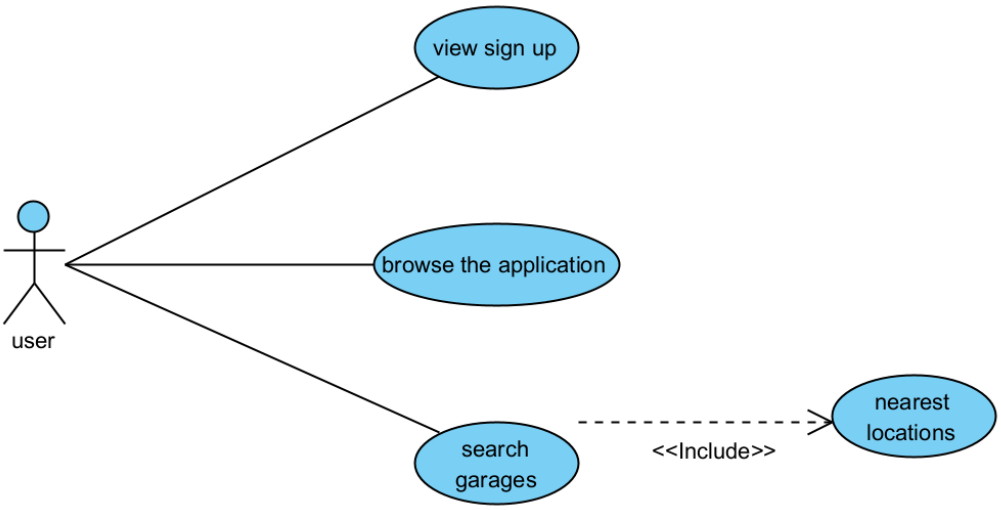


Figure 18

Figure 18 is for unregistered users as shown they don't have access to many features. The only access they have is to browse the application view the garages and locations available near them and they can view the sign-up page. They have been given only these features as it will allow them to get as much of an experience with the application as they can without giving up any of the crucial features only intended for those who sign up.

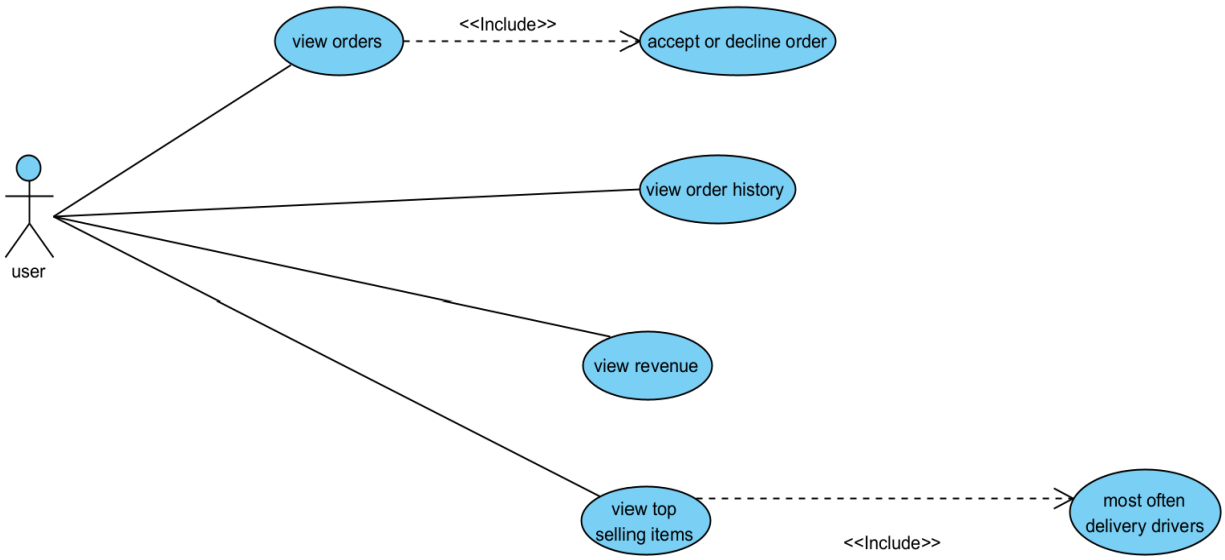


Figure 19

Figure 19 is the use case diagram for the business owner they can view the incoming order from the customer and choose whether they accept the order or decline it, which will send the message to the driver. They all be able to view all their previous orders which can then be viewed on a separate page where they can view the revenue they have made during the week and then separately see their best-selling items and who has been their most frequent delivery driver.

Design

This section includes the design of all the pages I want for my application. It will feature a screenshot of the page with a description afterwards. To create the design, I used balsamic wireframes as it was a UI framework that I have previously had experience with. Due to my making a mobile application UI need a lot of screenshots for my design so I chose Balsamiq as it is a low fidelity design and is very easy and quick to use. If I had used a high-fidelity design for example adobe xd this would have taken time for me to learn and then a lot more extra time to design each page. However, with balsamiqs design, it gives an idea of what each page should look like once completed roughly.

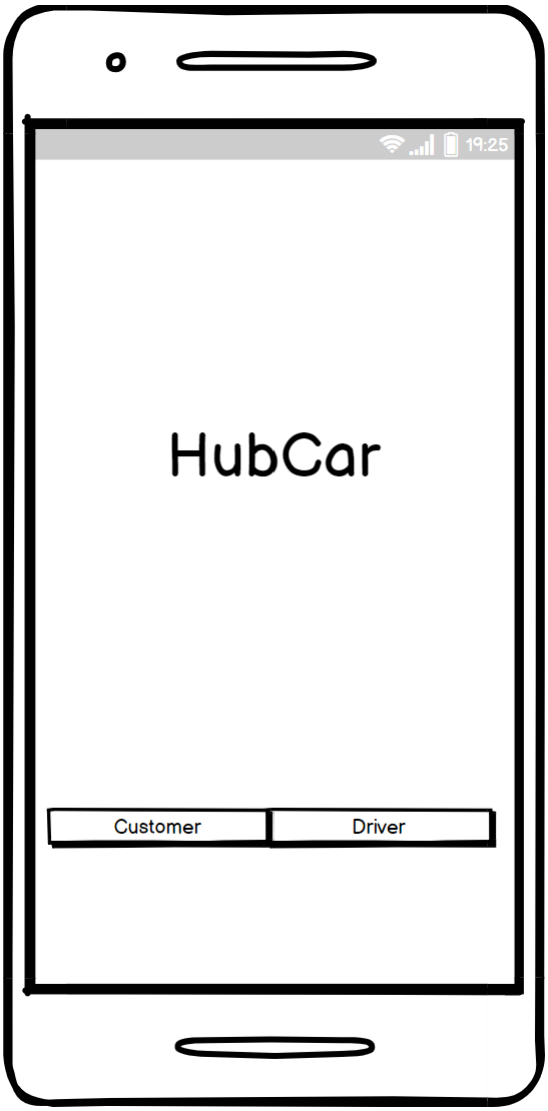


Figure 20

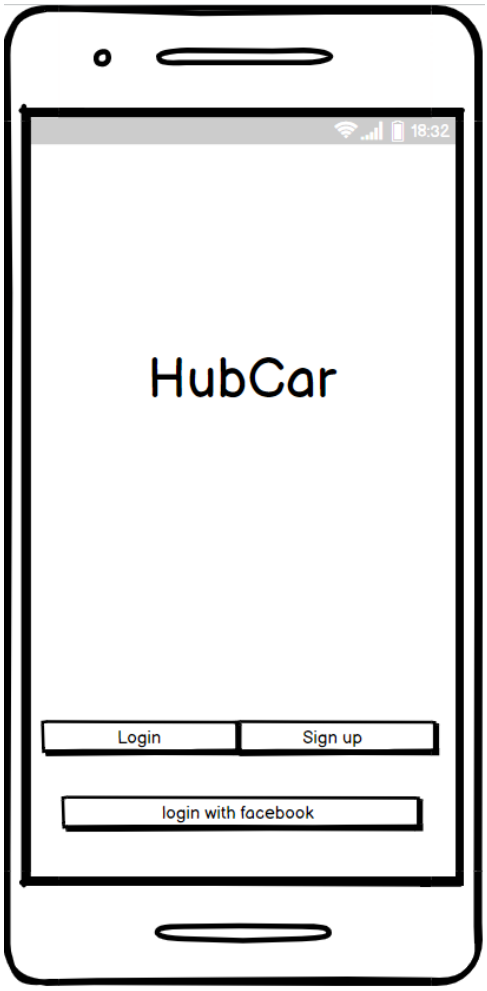


Figure 21

Figures 20 and 21 show the opening activity design and will first present the user with the button to choose either customer or driver. Once they have selected, they will be sent to the login/ signup page where they even have the option to log in using their Facebook account.

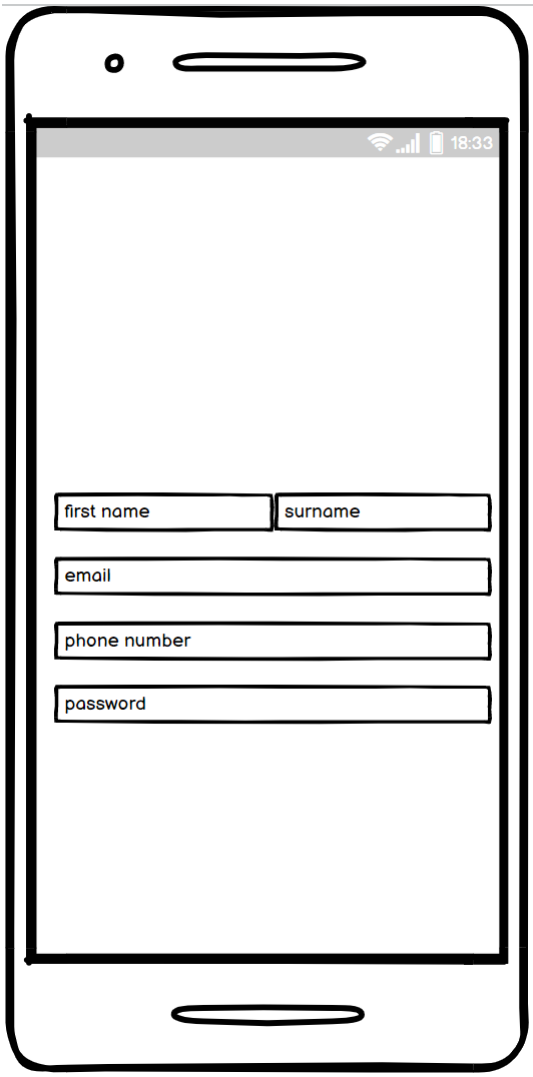


Figure 22

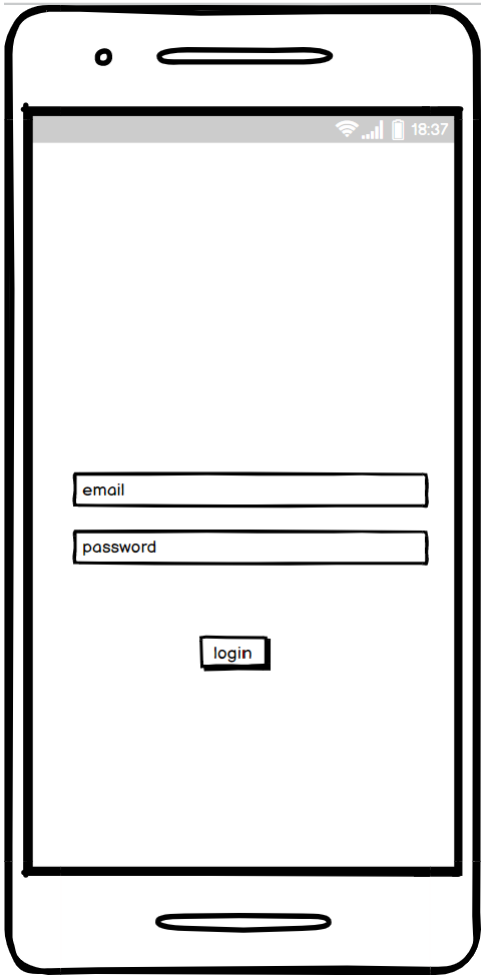


Figure 23

Figures 22 and 23 show the sign-up page and the figure shows the login page although these are basic designs these are what both pages will look like as they don't need much complexity.

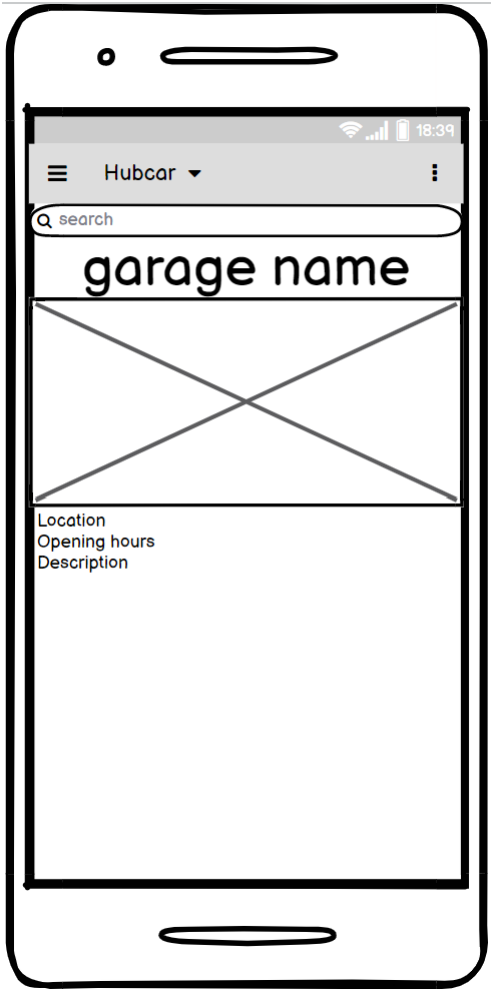


Figure 24

Figure 24 shows Once the user is a registered user and is signed in this will be their home page it will automatically bring them to the e-commerce page. Here it will show all the garages on the application, and they will be able to scroll through. Each garage will have its name at the top of its section along with a picture of its establishment and below a description with the location and opening hours. The user will be able to scroll through the various garages or be able to use the search bar. Users can also use the menu button top left to navigate to other pages.

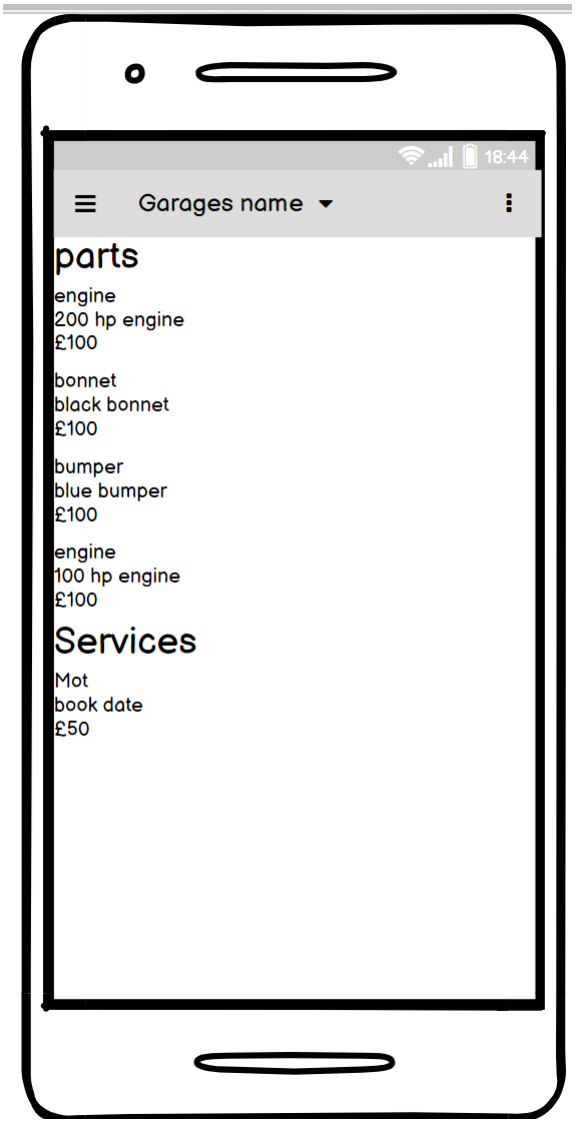


Figure 25

Figure 25 shows the item lists and services once a garage has been selected. The page will be split into 2 parts in the first half and services in the second once the user selects an item, they will be brought to another page.

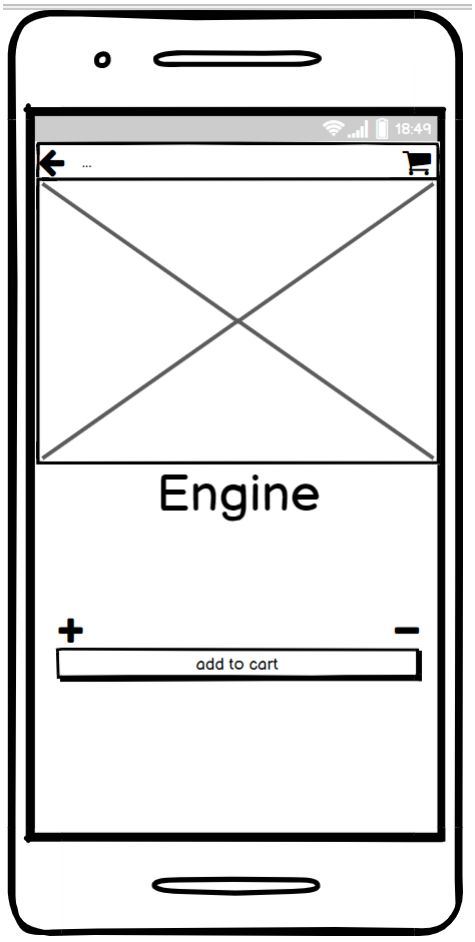


Figure 26

Figure 26 is a follow-up form figure once the user has selected the item, they will be brought to this page which will show the user the item with a picture and description. They will be able to select the quantity they want, and press adds to cart. Top left they can press the button to go back or press the cart on the top right to view their cart.

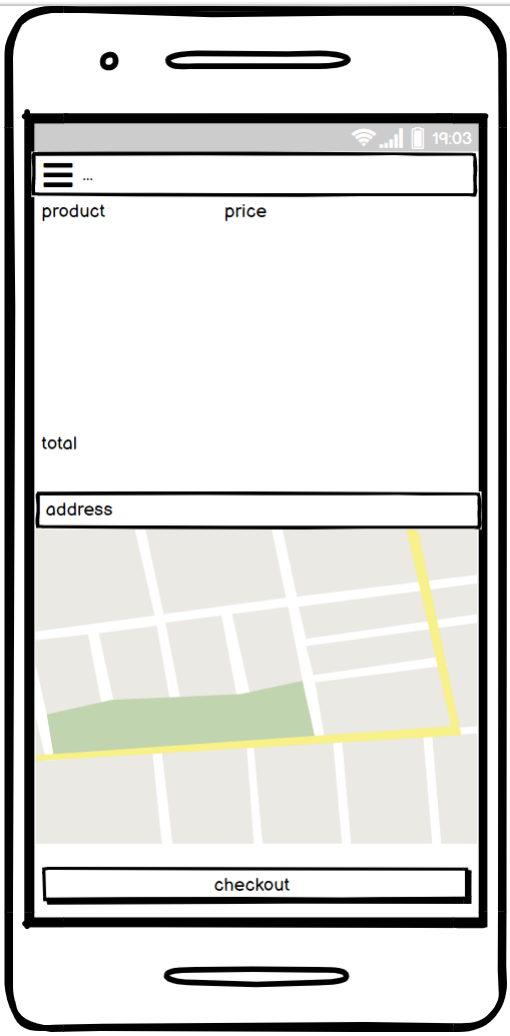


Figure 27

Figure 27 shows the page once the user has selected add to cart it will have the product with the amount the price along with the total. It will then ask for the address which will be located through google maps and show the user's location.

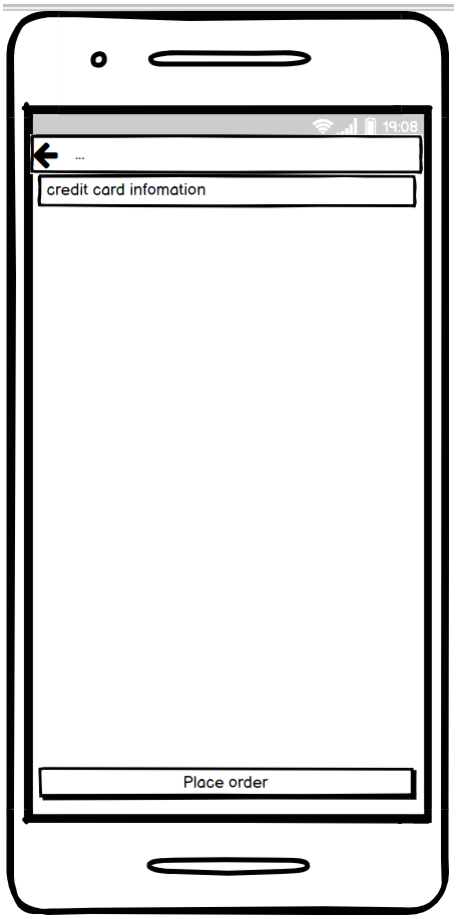


Figure 28

Figure 28 shows the payment page after checkout the user will have to enter their card number and CVV number and place an order.

Figure 29

Figure 29 shows the delivery process it will have google maps showing the distance from the garage to the user's location and where the driver is. At the bottom will have the details of the purchase and a delivery message.

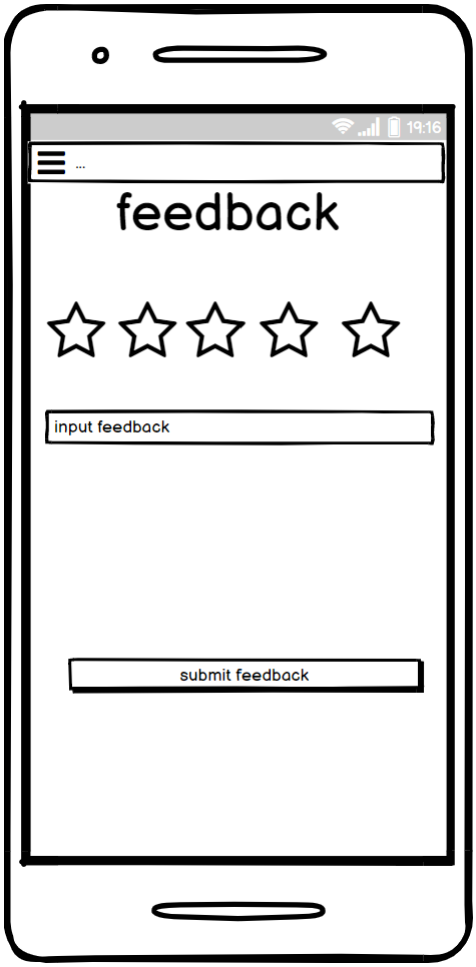


Figure 30

Figure 30 shows the completion of the purchase and delivery and will prompt the user with a feedback page where they can leave a rating out of 5 stars along with a message and send the feedback

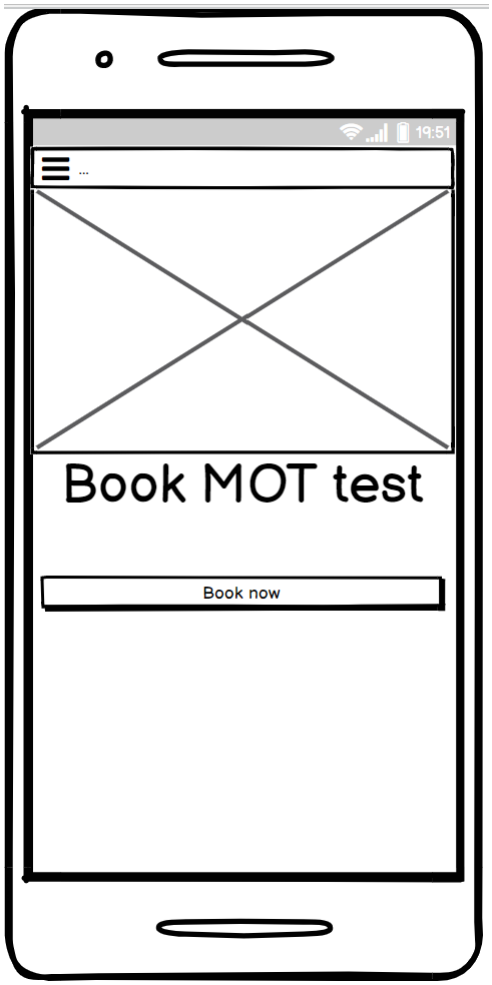


Figure 31

Figure 31 shows the booking feature/services when selecting this function this is the first page and is very similar to the items page when purchasing.

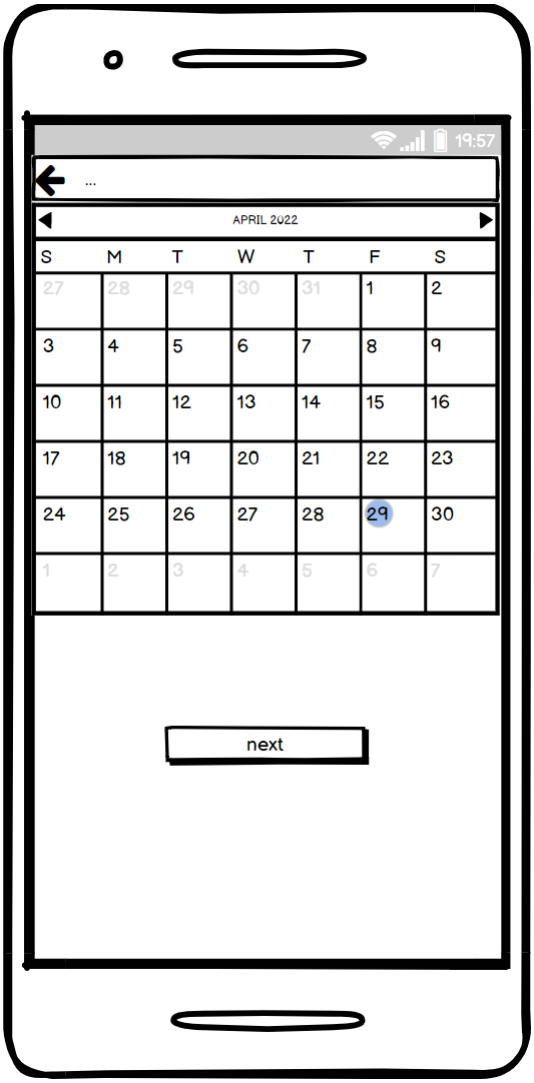

19:57
...
APRIL 2022
S M T W T F S
27 28 29 30 31 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
1 2 3 4 5 6 7
next

Figure 32

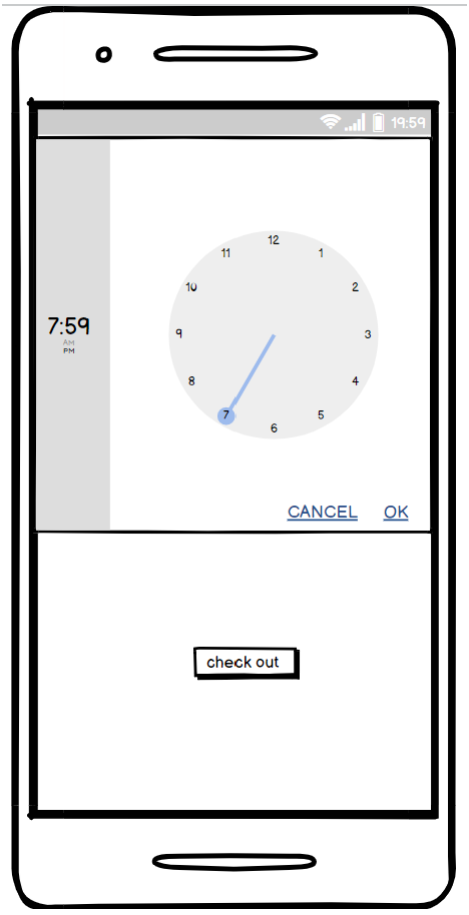


Figure 33

Figures 32 and 33 show the booking process first the user will be presented with a date to select and once the date has been selected, they will need to select a time. Once both steps have been completed and have selected checkout, they will be brought to the same payment page as the figure and once payment has gone through booking is confirmed.

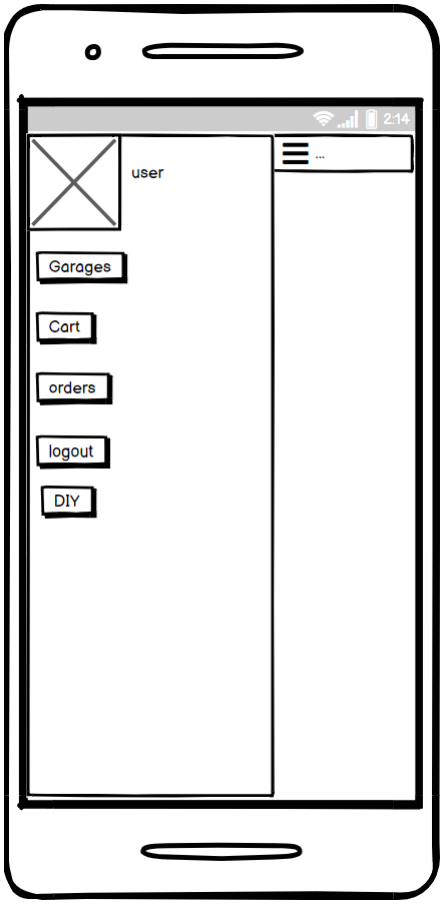


Figure 34

Figure 34 shows the menu bar for a registered user at the top will be their image along with their user info. Below they will have 5 pages to choose from. Garages will take them to the eCommerce section. The cart will show them items they have in their cart. Orders will show any current deliveries and their status of them. The log out option. And DIY which will take them to the DIY page.

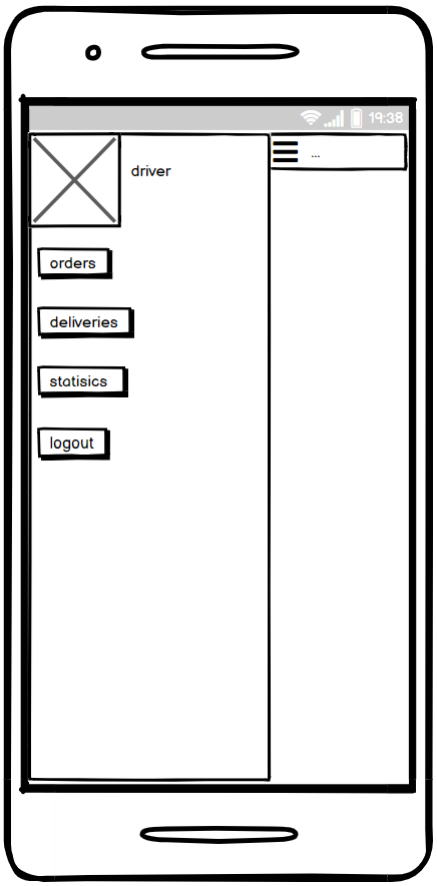


Figure 35

Figure 35 shows the menu bar for a driver they have a similar layout as a figure, but with different buttons in order, they can view what orders are currently going and choose to accept a job. Deliveries will be able to view past deliveries they have done. Statistics are how many deliveries they have done in a week and the logout button.

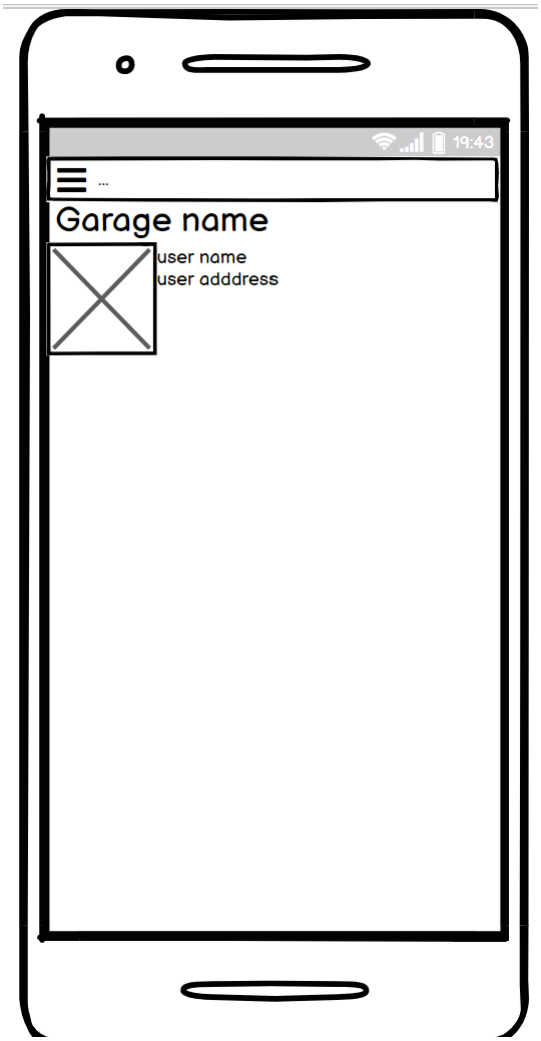


Figure 36

Figure 36 shows the order page for a delivery driver it will show them the garage's name with the user's details below once they click on the profile, they can choose to accept the order.

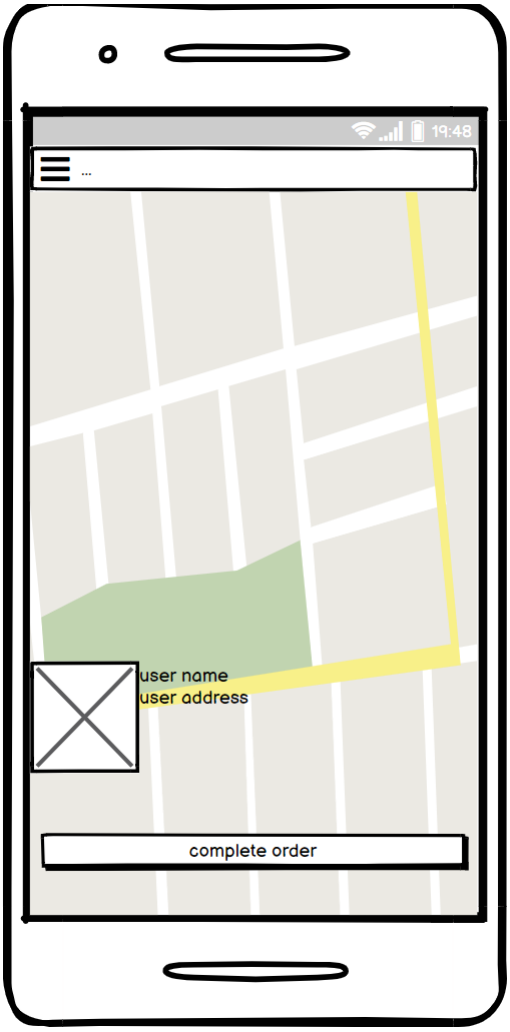


Figure 37

Figure 37 shows the page once the driver has confirmed the order, they will be given the direction to the user's location through google maps and will have the user's name and address. Upon arrival, the driver will select the complete order to finish the delivery.

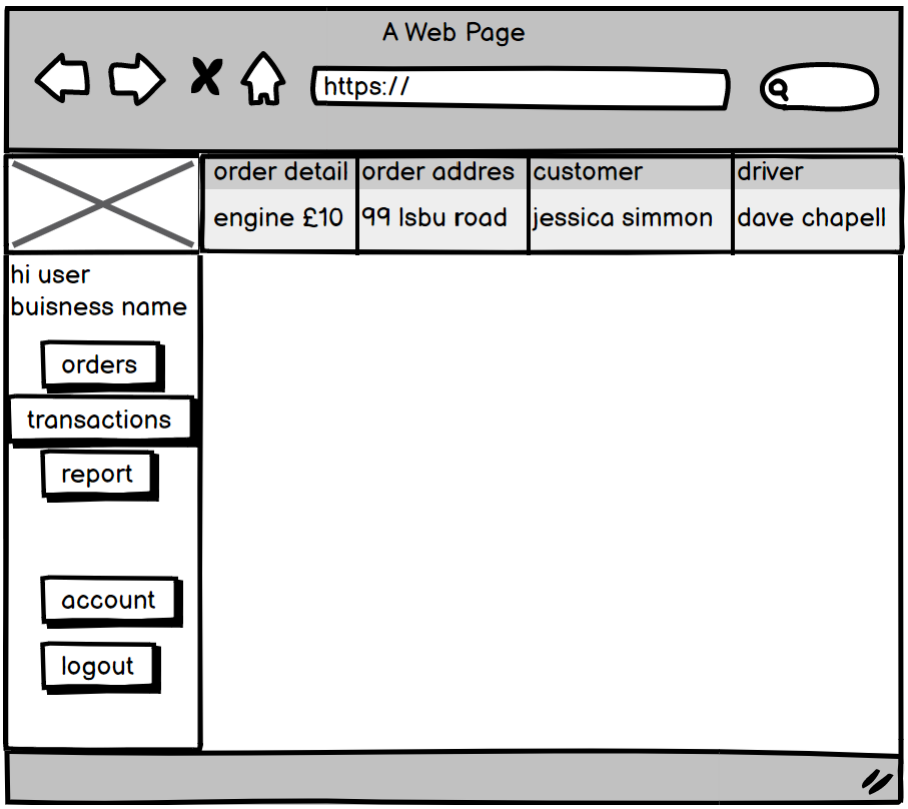


Figure 38

Figure 38 shows the order page for the business when an order comes through it will show the details of the order and they can choose whether to accept or decline an order once they have done this then a message will be sent to the driver.

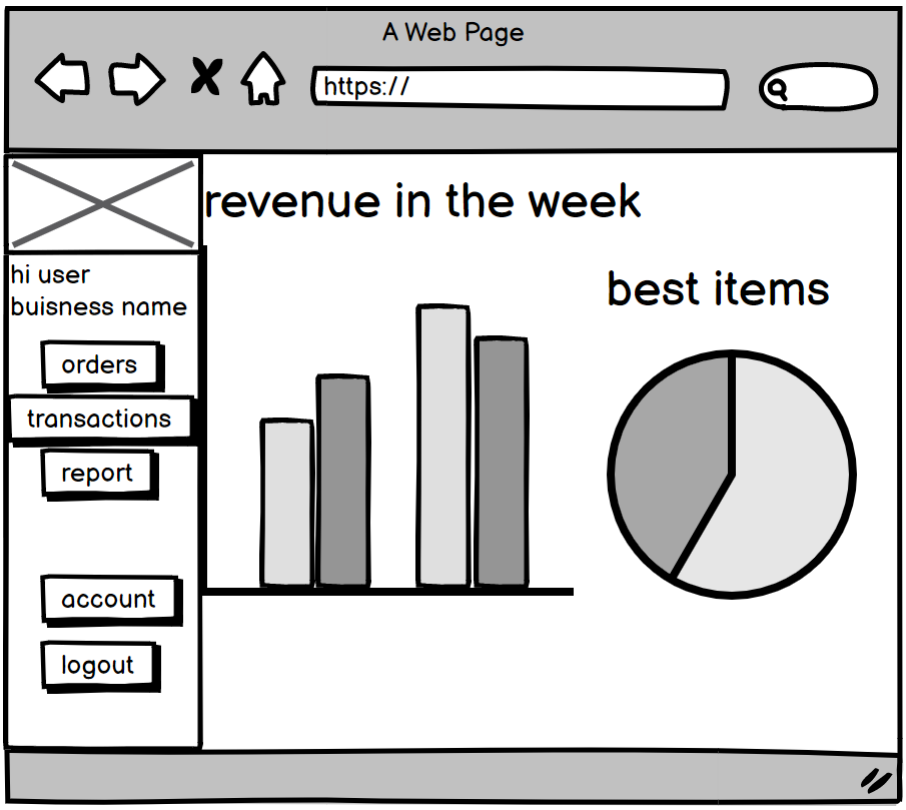


Figure 39

Figure 39 shows the report page for the business it will show them a chart of their best-selling items and how much they have made through orders each day of the week.

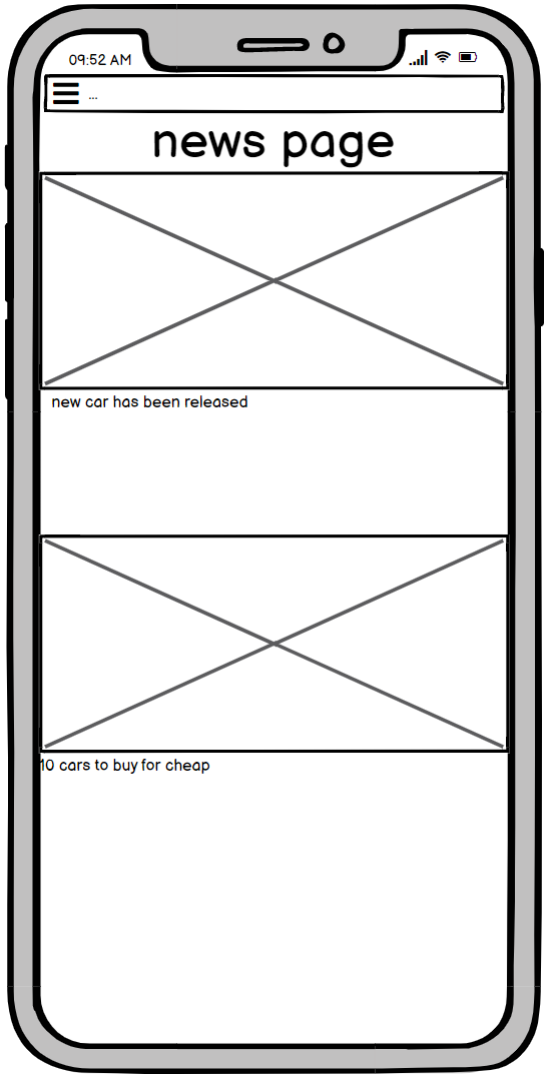


Figure 40

Figure 40 is a concept of a page I would like to add if I complete all other requirements this wasn't featured in my requirements section but is a news section that I would like to add that would look like this.

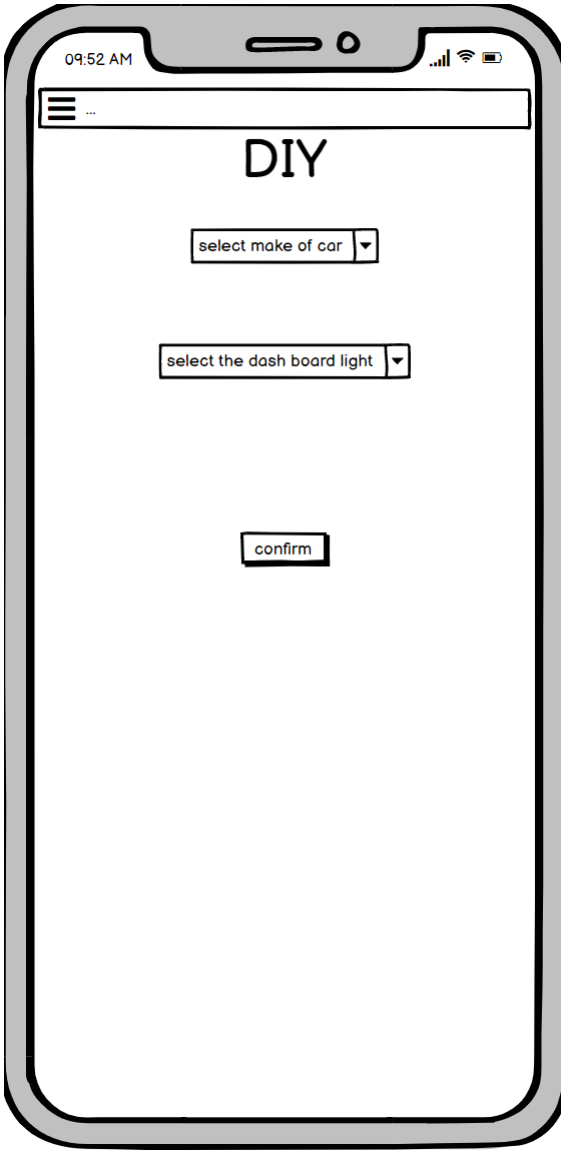

09:52 AM
...
DIY
select make of car
select the dash board light
confirm

Figure 41

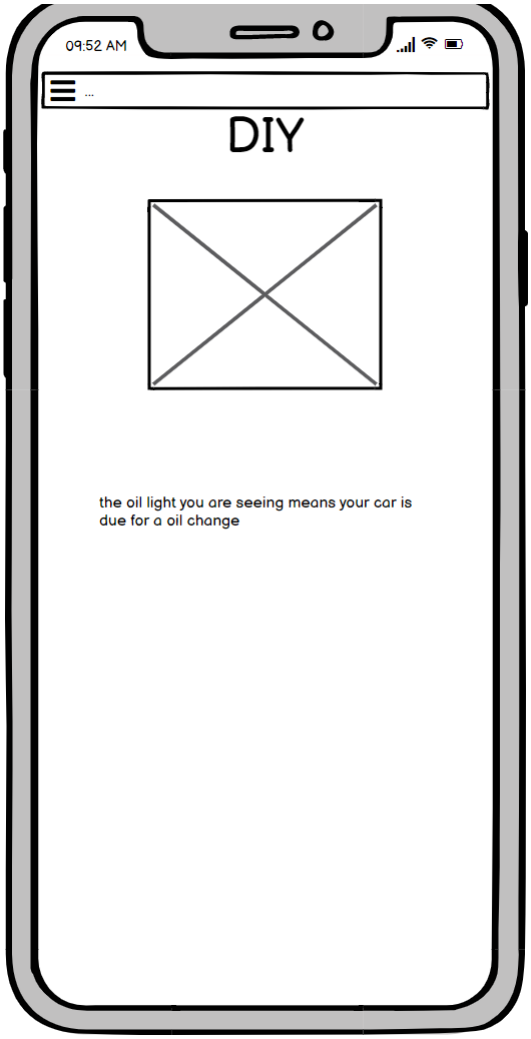


Figure 42

Figures 41 and 42 show the DIY function of the application figure shows the user has two options to select the make of their car and select dashboard light this will then send the user to a figure which will show them a picture of what needs to be done along with a description.

Implementation

this section will explain the most important parts of the implementation process to build the HubCar application. The coding part of the functionality will be explained here along with which tools were used, the methods and data interactions which will be explained in more detail.

Set up

Python

For the backend, the first step for the set up is installing python which just requires a download from the python website and checking the correct version is installed.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\haroo> python
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 43

Heroku

To be able to host my application I used Heroku to do this all that is needed is to create an account and needed to install the Heroku command line.

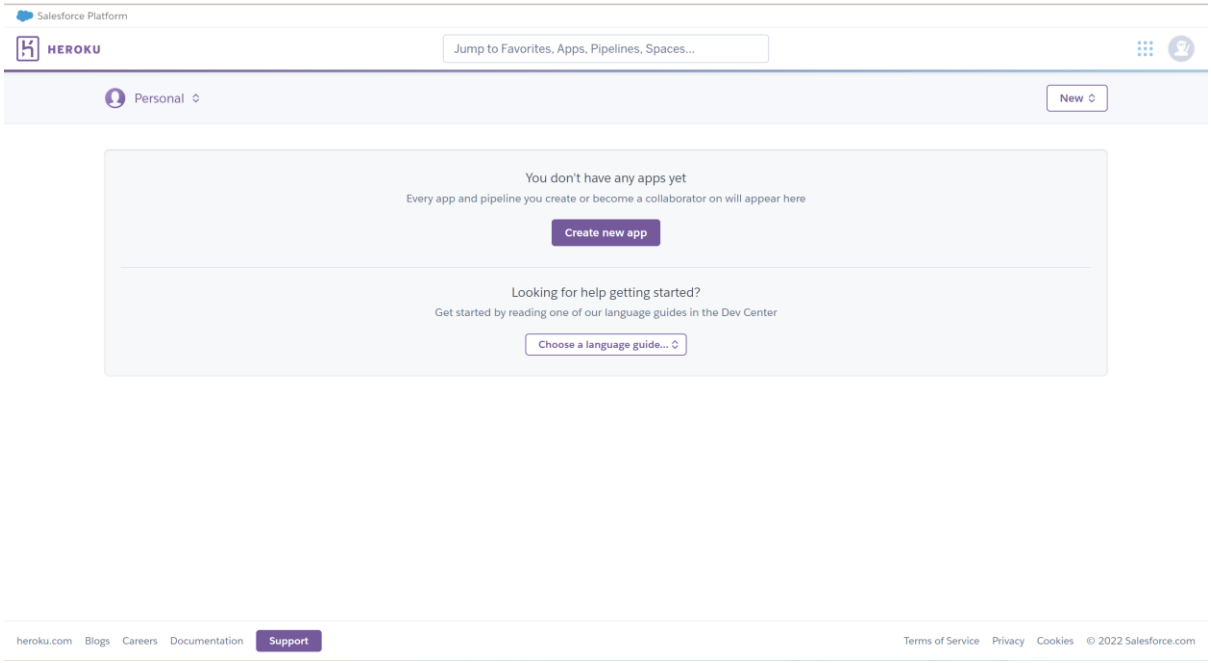


Figure 44

Postman APIs

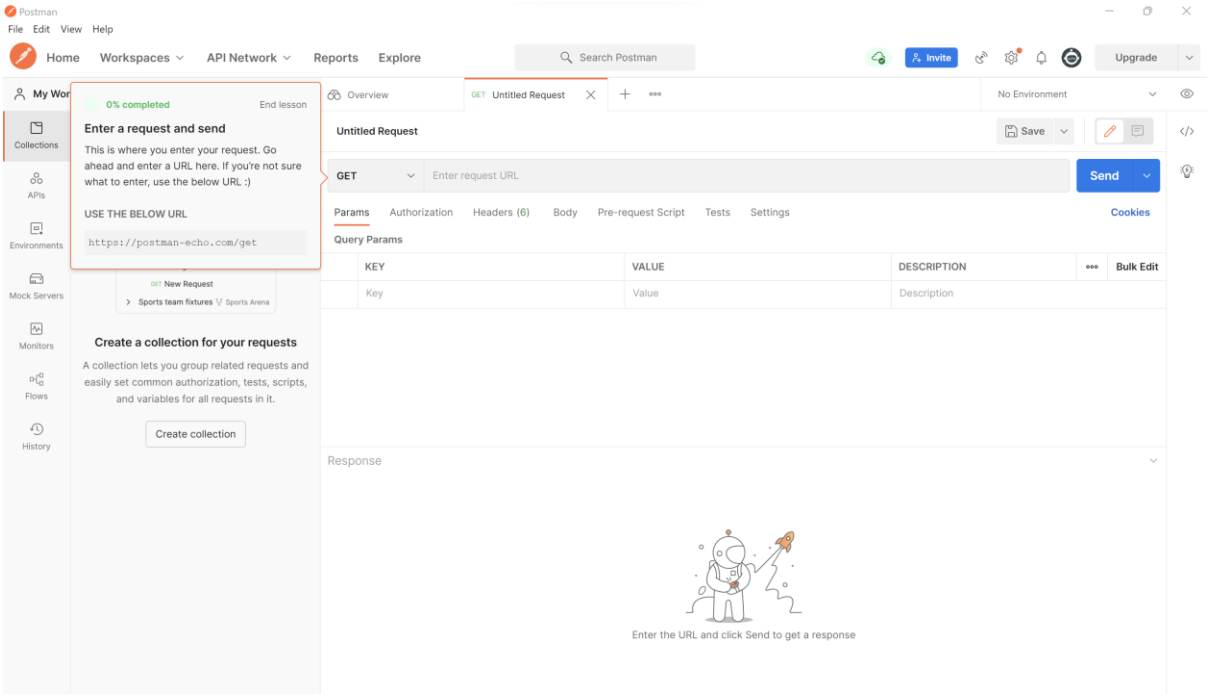


Figure 45

This is the dashboard for the postman and will allow to me add a URL that I want to add for my application just by adding the URL to the request bar and selecting the method needed.

Android studio

To be able to create an android app the environment for android deployments needs to be set up. When an android studio has been downloading the set-up for my project, I selected a custom setup so that was able to customize installation settings and components could be installed. The next step was to install an android virtual device so that I could test on the emulator.

Facebook login

```
'oauth2_provider',
'social.apps.django_app.default',
'rest_framework_social_oauth2',
'bootstrap3'
```

Figure 46

```
 'social.apps.django_app.context_processors.backends',
 'social.apps.django_app.context_processors.login_redirect',
```

Figure 47

```
AUTHENTICATION_BACKENDS = (
    'social.backends.facebook.FacebookOAuth2',
    'rest_framework_social_oauth2.backends.DjangoOAuth2',
    'django.contrib.auth.backends.ModelBackend',
)

# Facebook configuration
SOCIAL_AUTH_FACEBOOK_KEY = '595892800571371'
SOCIAL_AUTH_FACEBOOK_SECRET = '08dba096cd8f0dba7ae27b9f434c55c6'

# Define SOCIAL_AUTH_FACEBOOK_SCOPE to get extra permissions from
SOCIAL_AUTH_FACEBOOK_SCOPE = ['email']
SOCIAL_AUTH_FACEBOOK_PROFILE_EXTRA_PARAMS = {
    'fields': 'id,name,email'
}
```

Figure 48

Figures 46, 47 and 48 show connecting the application to Facebook to this the app secret and key need to both be implemented to link the application. FacebookOAuth2 is needed to authenticate with Facebook.

```
def create_user_by_type(backend, user, request, response, *args, **kwargs):
    if backend.name == 'facebook':
        avatar = 'https://graph.facebook.com/%s/picture?type=large' % response['id']

    if request['user_type'] == "driver" and not Driver.objects.filter(user_id=user.id):
        Driver.objects.create(user_id=user.id, avatar = avatar)
    elif not Customer.objects.filter(user_id=user.id):
        Customer.objects.create(user_id=user.id, avatar = avatar)
```

Figure 49

Figure 49 shows the authentication from Facebook. When this function is run, it will first check to see if the backend is named Facebook, which means it will look throufacebookook and create avatars from this link

```
https://graph.facebook.com/%s/picture?type=large
```

this string will return the avatar images. Then it will check if the user type is the driver and if the driver object exists if it does not it creates a new driver object in data base which equato the id from create_user and the avatar from the Facebook link. Otherwise, if the user type is not the driver, then a new customer and the id will be checked through user-id user id just like the driver and will be entered into the database.

On the application, the user will have the choice when signing up to eitherlog inn through the facebook account on the signup page if they choose facebook they will be taken to the Facebook and sign in which will then store their details in the database allowing for the application to make their user profile.

The driver picking up an order

```python
@csrf_exempt
# POST
# params: access_token, order_id
def driver_pick_order(request):

    if request.method == "POST":
        # Get token
        access_token = AccessToken.objects.get(token = request.POST.get("access_token"),
            expires__gt = timezone.now())

        # Get Driver
        driver = access_token.user.driver

        # Check if driver can only pick up one order at the same time
        if Order.objects.filter(driver = driver).exclude(status = Order.ONTHEWAY):
            return JsonResponse({"status": "failed", "error": "You can only pick one order at the same time."})

        try:
            order = Order.objects.get(
                id = request.POST["order_id"],
                driver = None,
                status = Order.READY
            )
            order.driver = driver
            order.status = Order.ONTHEWAY
            order.picked_at = timezone.now()
            order.save()

            return JsonResponse({"status": "success"})

        except Order.DoesNotExist:
            return JsonResponse({"status": "failed", "error": "This order has been picked up by another."})

    return JsonResponse({})
```

Figure 50

Figure 50 shows the driver picking up an order ready for delivery. First, @csrf_exempt is needed because this is a post request. This is needed as it is a post request from the business side of the system, so I need to exempt the csrf checking process to allow the driver to pick up the order. The access token is needed so the application can make API request on behalf of the user. The access token is the authorization of an application to access users' data. Once the token is acquired the driver needs to be obtained based on that token. Once the driver has been obtained, they need to check how many orders the driver can collect at once. The all the orders will be obtained from the database and satisfy the condition of driver = driver and exclude all orders that have been stated on the way by checking this condition it will see if a driver is currently doing an order. Next will present the order for drivers the order will be submitted satisfy the order id and have the status as ready. To assign an order to a driver the order.driver is used and the order status is updated to on the way and the time is set for pickup and order 2 different drivers are picking up the same order at the same time the order needs to be removed for one driver so except Order.DoesNotExist is used, and they will be presented with an error message.

Getting the latest order

```
class OrderCustomerSerializer(serializers.ModelSerializer):
    name = serializers.ReadOnlyField(source="user.get_full_name")

    class Meta:
        model = Customer
        fields = ("id", "name", "avatar", "phone", "address")

class OrderDriverSerializer(serializers.ModelSerializer):
    name = serializers.ReadOnlyField(source="user.get_full_name")

    class Meta:
        model = Customer
        fields = ("id", "name", "avatar", "phone", "address")

class OrderGarageSerializer(serializers.ModelSerializer):
    class Meta:
        model = Garage
        fields = ("id", "name", "phone", "address")

class OrderItemSerializer(serializers.ModelSerializer):
    class Meta:
        model = Item
        fields = ("id", "name", "price")

class OrderDetailsSerializer(serializers.ModelSerializer):
    item = OrderItemSerializer()

    class Meta:
        model = OrderDetails
        fields = ("id", "item", "quantity", "sub_total")

class OrderSerializer(serializers.ModelSerializer):
    customer = OrderCustomerSerializer()
    driver = OrderDriverSerializer()
    garage = OrderGarageSerializer()
    order_details = OrderDetailsSerializer(many = True)
    status = serializers.ReadOnlyField(source = "get_status_display")
```

Figure 51

```
class Meta:
    model = Order
    fields = ("id", "customer", "garage", "driver", "order_details", "total", "status", "address")
```

Figure 52

Figures 51 and 52 show the getting an order. There are serializers in the Django REST Framework that convert data types that can be understood by Javascript and front-end frameworks into those

that can be understood by Django. Data can be deserialized, allowing complex types to be returned to their original form, after validation of the incoming data. REST framework's serializers are extremely like Django's Form and ModelForm classes in terms of functionality. The reason for so many serializers to make it easier to convert customer data from the database into JSON format and pass it back to clients. metaclassss is a container for the model's choices (metadata). Singular and plural forms of names, permissions, associated database table names, and whether the model is abstract are all defined.

Updating drivers' location

```
def customer_get_latest_order(request):
    access_token = AccessToken.objects.get(token = request.GET.get("access_token"),
        expires__gt = timezone.now())

    customer = access_token.user.customer
    order = OrderSerializer(Order.objects.filter(customer = customer).last()).data

    return JsonResponse({"order": order})

def customer_driver_location(request):
    access_token = AccessToken.objects.get(token = request.GET.get("access_token"),
        expires__gt = timezone.now())

    customer = access_token.user.customer

    # Get driver's location related to this customer's current order.
    current_order = Order.objects.filter(customer = customer, status = Order.DELIVERED).last()
    location = current_order.driver.location

    return JsonResponse({"location": location})
```

Figure 53

Figure 53 shows the updating of the driver's location first the access token needs to be passed from the access token a customer is found once they have been found this condition needs to be satisfied customer = customer and the status is on the way. Once the current order has been obtained the driver is taken from this then the location of the driver can be found. once we have the location it can be returned in a JSON response. This is used to monitor the location of the driver so that alerts can be sent to the user on the updates of the delivery and where the driver is they willl be able to follow the drivers icon through the order page to see how far they are.

Getting items

```
def customer_get_items(request, garage_id):
    items = ItemSerializer(
        Item.objects.filter(garage_id = garage_id).order_by("-id"),
        many = True,
        context = {"request": request}
    ).data

    return JsonResponse({"items": items})
```

Figure 54

Figure 54 shows getting items the garages id belongs to the items object and the restaurant = restaurant is the parameter passed in the def customer_get_items. The next step is many = true which tells the def that the set contains many items so the def needs to serialize each item with a serializer class. Context = is needed so the correct URL can be obtained for the image. This is so each item in the user's order can be viewed and they will be able to see what items have been purchased through their history.

Order model

```
class Order(models.Model):
    PREPARING = 1
    READY = 2
    ONTHEWAY = 3
    DELIVERED = 4

    STATUS_CHOICES = (
        (PREPARING, "Preparing"),
        (READY, "Ready"),
        (ONTHEWAY, "On the way"),
        (DELIVERED, "Delivered"),
    )

    customer = models.ForeignKey(Customer)
    garage = models.ForeignKey(Garage)
    driver = models.ForeignKey(Driver, blank = True, null = True)
    address = models.CharField(max_length=500)
    total = models.IntegerField()
    status = models.IntegerField(choices = STATUS_CHOICES)
    created_at = models.DateTimeField(default = timezone.now)
    picked_at = models.DateTimeField(blank = True, null = True)

    def __str__(self):
        return str(self.id)


class OrderDetails(models.Model):
    order = models.ForeignKey(Order, related_name='order_details')
    item = models.ForeignKey(Item)
    quantity = models.IntegerField()
    sub_total = models.IntegerField()

    def __str__(self):
        return str(self.id)
```

Figure 55

Figure 55 shows the order model first thing is to create a new class for orders. First, the order needs four statuses. First the order of the messages is set then the messages sent to the user is addThe nextNext step is to add all the information about the order is the driver, the customer etc. for the time zone it needs to be imported by importing the Django function from the utils library. Whenever a new model is crea it the chain needs to be moved into a database so through the terminal the command python manage.py makemigrations function is used followed by python manage.py migrate and finally python manage.pyrun server. The next step is to create the object for order details to record every item that is in the order. This is where the class order details are used, inside the OrderDetails the item needs to be a foreign key because the item needs to be linked and the string function is created for all of these. The final step is to add the order and order details objects into the admin dashboard which. This will be how the user will be updated on the status of their

order and will follow the multiple stages giving the user the notification through each step and storing the data for their order history.

garage revenue

```python
@login_required(login_url='/garage/sign-in/')
def garage_report(request):
    # Calculate revenue and number of order by current week
    from datetime import datetime, timedelta

    revenue = []
    orders = []

    # Calculate weekdays
    today = datetime.now()
    current_weekdays = [today + timedelta(days = i) for i in range(0 - today.weekday(), 7 - today.weekday())]

    for day in current_weekdays:
        delivered_orders = Order.objects.filter(
            garage = request.user.garage,
            status = Order.DELIVERED,
            created_at__year = day.year,
            created_at__month = day.month,
            created_at__day = day.day
        )
        revenue.append(sum(order.total for order in delivered_orders))
        orders.append(delivered_orders.count())
```

Figure 56

Figure 56 shows the revenue section for the garage. The first thing is adding the function to calculate revenue by week which require the import of the DateTime and timedelta functions. The two arrays of revenue and order were created to contain the data. Then the function to calculate the weekdays is created which takes every single day in the week and will get the delivered_orders once these have been obtained revenue_append calculates all the total amounts of the orders and sums up everything which will then be held in revenue for one day. Just like revenue. Append orders. The append will count how many orders were delivered during that day which will be held in order. append for one day which is the reason for the count. The data stored will be sent to the garages dashboard where they'll be able to view their revenue through charts each week they will also be presented 2 other charts showing their best-sold item and their most frequent driver.

Testing

This section of the report will go through the testing of functionalities for this project. Had this been an actual product I was released to the public a testing team would've been required to complete this task and had it been made for a client they would have also been used to test the product. But as there is only me doing the work, I am responsible for the testing of the project as well.

Testing is needed so that we can see if the product can do what it was intended to do and see whether it has any issues or bugs that need to be identified and solved before release.

If I had an actual scrum team for this project a testing team and the development team would have worked together to develop and test the software in a sprint. While testing the team would test the product as if they were a customer to see how the product would look from their point of view. But as I am doing the project by myself scrum will be followed but not exactly as it is intended to be.

After each functionality for my project, I tested it using an emulator, as can be expected with most projects the functionalities did not do the task they were intended to and changes had to be made, even resulting in some functionalities being left out. During the testing, I used my user stories as references to see if the functionalities met these requirements. I did not do the documentation for these tests I conducted as I needed to save time and just focus on developing the application.

Below I will be talking about the two different approaches to testing and how they have been implemented the first being unit testing and the second device testing

Unit testing

Unit testing is a type of software testing that focuses on checking the functionality of individual programme units or components. The goal of this test is to ensure that every piece of software works as expected. A unit is the tiniest piece of software that can be tested. It typically has one or a few inputs and a single output, with a few exceptions. App testing strategies are built around unit tests, which are the most fundamental. Unit tests allow you to quickly check that the functionality of single components in your code is valid by generating and running them against it. Unit testing is also critical since it aids in the rapid detection and correction of software defects.

Unit tests can't be used to test complicated UI interaction events. Instead, you should use the UI testing frameworks, like the ones in Automating UI Tests, to run the tests. For this project, this isn't going to work because the UI is standard and doesn't have a lot of complicated interaction.

Device testing

When testing a mobile application, the main thing being looked at is how the application will perform on different devices. While developing and testing a good software to use to test are emulators but a downside to these is they will not show how the application will function when using the mobile's hardware and screen. But with the use of actual mobile devices, it will show how the application will function in real-time on that device and will show how the application will react on different makes and software versions. another thing to take into account and is a benefit is testing on an actual device will take real-life variables into account for example to test I used an old Samsung s7 which is quite outdated and had several problems due to age-related issues, example, its battery was in bad condition and due to its age was quite slow it can show how the application will react if another application was also open on the device such as Spotify which can run in the background.

Test plan

This section will include how the test was done and what device was used.

Test environment

For the testing environment, I used two different processes an emulator and actual devices. First, I used an emulator with a google pixel phone.

For the actual devices, I used 2 different devices the Samsung galaxy s20 and Samsung galaxy s7 I used these as they were the only devices, I had available.

Testing configuration

Since the virtual emulator is part of the Android Studio IDE, it doesn't require a lot of setups. Downloading the appropriate files for the emulator selected in this example, Pixel 3a, is all that is required for setup.

To conduct device testing a mobile phone is needed, a physical connection, developer mode enabled and a Wi-Fi connection.

Test case creation

Below is a table showing the tests completed and inserted into a template. I have created 7 different tests which I will provide a short description. The test date is not added however as tests were done multiple times and recording each test would just be too time-consuming.

Test Case ID	0001	**Test Case Description**	Test the Login Functionality		
Created By	Haroon	**Reviewed By**	N/A	**Version**	1.0

Tester's Name	Haroon	**Date Tested**		**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Emulator
2	
3	
4	

S #	Test Data
1	Username = haroonahmad
2	Password = haroona
3	
4	

Test Scenario	Verify on entering valid username and password, the customer can login

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run Application	Open Activity Opens	As Expected	Pass
2	Click Login Button	Login screen opens	As Expected	Pass
3	Enter username, password	Details can be entered	As Expected	Pass
4	Click Login	User is logged in	As Expected	Pass

Figure 57

Figure 57 shows test 1 which will just test if the user can log in

Test Case ID	0002	Test Case Description	Test the Login Functionality with wrong details		
Created By	Haroon	Reviewed By	N/A	Version	1.0

Tester's Name	Haroon	Date Tested		Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Emulator
2	
3	
4	

S #	Test Data
1	Username = haroonahmad123
2	Password = haroona23
3	
4	

Test Scenario	User enters wrong details they cannot enter the application

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run Application	Open Activity	As Expected	Pass
2	Click Login Button	Login screen	As Expected	Pass
3	Enter username, password	Details can be entered	As Expected	Pass
4	Click Login	User cannot access the application due to wrong details	As Expected	Pass

Figure 58

Figure 58 will test to see if the user will be able to log in even if they enter the wrong credentials.

Test Case ID	0003	Test Case Description	Browsing and search function		
Created By	Haroon	Reviewed By	N/A	Version	1.0

Tester's Name	Haroon	Date Tested		Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Emulator
2	
3	
4	

S #	Test Data
1	Username = haroonahmad
2	Password = haroona
3	
4	

Test Scenario	The user has logged in and can scroll and search for a garage

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run Application	Open Activity	As Expected	Pass
2	Click Login Button	Login screen	As Expected	Pass
3	Enter username, password	Details can be entered	As Expected	Pass
4	Click Login	User is logged in	As Expected	Pass
5	Scroll on home page	User can scroll through the different garages on the home page	As expected	Pass
6	Use search bar to search for a garage	User will use search bar to search for a specific garage and it will appear	As expected	Pass

Figure 59

Figure 59 shows the third test that the user can successfully log in and browse the homepage and use the search function.

Test Case ID	0004	Test Case Description	Purchasing an item		
Created By	Haroon	Reviewed By	N/A	Version	1.0

Tester's Name	Haroon	Date Tested		Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Emulator
2	
3	
4	

S #	Test Data
1	Username = haroonahmad
2	Password = haroona
3	Card number = 1234 5678 1234 5678
4	Ccv = 123
5	City = London
6	County = Enfield
7	Address = 99 merryhills drive
8	Post code = en27pq

Test Scenario	User navigates through the application to purchase an item

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run Application	Open Activity	As Expected	Pass
2	Click Login Button	Login screen	As Expected	Pass
3	Enter username, password	Details can be entered	As Expected	Pass
4	Click Login	User is logged in	As Expected	Pass
5	Click garage	user selects garages page	As expected	Pass

Figure 60

6	User selects item and adds to car	User selects the item to purchase, and it's added to cart	As expected	Pass
7	User selects proceed to checkout	Once items have been added to cart user proceeds to checkout	As expected	Pass
8	User enters card details	User enters card details to confirm purchase	As expected	pass
9	User views tracking	User views tracking on order	As expected	Pass

Figure 61

Figure 60 and 61 shows the testing of purchasing an item it tests the whole process from logging in to the completion of the transaction.

Test Case ID	0005	Test Case Description	Test the Login Functionality for driver		
Created By	Haroon	Reviewed By	N/A	Version	1.0

Tester's Name	Haroon	Date Tested		Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Emulator
2	
3	
4	

S #	Test Data
1	Username = haroondriver
2	Password = driver
3	
4	

Test Scenario	Driver logs into his account

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run Application	Open Activity	As Expected	Pass
2	Click Login Button	Login screen	As Expected	Pass
3	Enter username, password	details can be entered	As Expected	Pass
4	Click Login	Driver logs in	As Expected	Pass

Figure 62

Figure 62 test the login function for drivers

Test Case ID	0006	Test Case Description	Test the Login Functionality for driver		
Created By	Haroon	Reviewed By	N/A	Version	1.0

Tester's Name	Haroon	Date Tested		Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	Emulator
2	
3	
4	

S #	Test Data
1	Username = haroondriver
2	Password = driver
3	
4	

Test Scenario	driver does an order

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Run Application	Open Activity	As Expected	Pass
2	Click Login Button	Login screen	As Expected	Pass
3	Enter username, password	details can be entered	As Expected	Pass
4	Click Login	Driver logs in	As Expected	Pass
5	Driver views order	Driver views available order through menu	As expected	Pass
6	Driver selects confirm order	Driver confirms the order from garage through the orders section	As expected	pass

Figure 63

7	Driver has a google map of the garage location and user address	When the driver selects confirm, a map should appear of location of garage and direction to users location	As expected	Pass
8	Driver completes order	Driver presses complete order upon arrival of the users location	As expected	pass

Figure 64

Figure 63 and 64 show the test for the delivery driver receiving an order confirming it and completing it.

Devices used

Samsung galaxy s20

Samsung galaxy s7

I was only able to use these 2 devices as I did not have access to others to test my application, but these devices should be enough as of now as the application is not in its final form but more of a beta stage. When it comes to the final deployment more devices would be used but this would take more time and work.

With the device testing, I had different findings with the two phones which will be stated but these are finding I had in general.

- The tracking function sometimes appears and sometimes doesn't the map might appear but with no direction
- The Samsung galaxy s7 did have some struggles when launching the application but this was due to the age of the phone and the phone is very slow in general
- The tab bar lags occasionally and may appear blank
- The Facebook login button does not work and will just appear with a blank page or crash the app

Future development and self-evaluation

Future work

This section will go into depth about the work needed to complete the application and get it ready for its final form. The application as it is now a good beginning but does require more work for actual deployment. While discussing and developing the project I had discussions with a lecturer about my project who seemed to like the proposal and gave me tips on what to add to the application and if features were to be added to come back to him and he would provide me with connections who can discuss with getting more resources to be able to deploy the product. for my project, I would like to have a dedicated team who could help me perfect the features and add ones that I wasn't able to due to complications and not being able to. They would be able to help me fix any faults currently with the application.

Future actions:

- Add a news page
- Add a quick fix page
- Finish all the requirements in the requirement section
- Fix all errors and bugs
- Improve the design of the product currently it is quite basic working with a UI designer would help make the application look more appealing
- Launch the application in the google play store
- Have an identification page where users can search for the problem they are facing and have resources such as YouTube videos or blogs to help solve such issues.

Self-evaluation

As the project is coming to an end evaluating the project has been done as well. Although the evaluation focuses on the final product, I will also be discussing other aspects such as the process and breaking it down into strengths, weaknesses and lessons learned.

Strengths

The first strength is the knowledge and experience gained from this project. This project is beyond anything I have created previously it gave me my first experience in mobile application development and a new experience working with APIs.

Weaknesses

A weakness I had during the project was mainly with testing the development of my application took longer than anticipated which meant when it came to testing it wasn't as thorough as I would have liked and was quite rushed. I also didn't have much experience with testing on mobile applications so with the time I was left with I worked with what I could.

Another weakness was documentation in general from the first draft report I hadn't understood what was required so it had to be redone and had to respond and take the feedback as much as I could. I also faced the same issue as testing due to me focusing so much on the development I found myself losing focus on documentation and often brushing it aside.

The third weakness was my planning/management even though I had to create a Gantt chart to allocate time and a Jira to state every task I often found myself allocating time to a task but then taking longer on the task than it should have or leaving it for a later time complete another one. As stated above testing and documentation should have had more time allocated to i. #

The final weakness would be the design to design my product I used Balsamiq which is a low fidelity UI/UX design I would have preferred to use a higher fidelity one but would have required more time and effort to learn.

All the features I would have liked to have added to the application I was not able to due to difficulties and the deadline.

Lessons learned

I learned a lot of lessons throughout the development of my application. My first lesson came from the initial progress report draft where I learned where I was going wrong in my documentation and valuable information that would be needed for a good grade. The second lesson I learned was every day and every section I had to do was as important as the next and would need my undivided attention.

Through the development process, I learned that it is very important to make sure you have done thorough research on the products you use for the development of your project. Because you may choose a certain software for example in my case, I trailed a few APIs before I chose postman APIs which by then already has lost time and had to relearn an API which cost me time. Some functionalities you add to your application may also need to be scraped and have to restart with a new idea which may change the direction of API, so research and choices were a big lesson.

Time allocation was a big lesson that I had learnt I spent too much time with the development of the application which left me under pressure when it came to documentation and testing had I managed my time properly I would have been under a lot less stress and could complete other tasks to a higher standard.

As stated in the point below the importance of testing is another lesson, I learned had I taken testing more seriously and emphasised it more I would have been able to detect bugs in my application earlier and resolved them. This will stick with me further on in life in my career.

Conclusion

In conclusion, this project has been the biggest and toughest I have faced however in some ways was also quite enjoyable and am pleased with the application I have created as it was my first time creating a mobile application. Along the journey, I came across many challenges which I have overcome so that I was able to complete the project. The project would have gone a lot smoother had I done things differently however these are mistakes I can make now and learn from. This project has enhanced my knowledge and helped me become a better programmer and mobile application developer. As stated, before I am pleased with my final product as stated before even though I could have done better. However, considering I have never made a mobile application before and never had experience with APIs when I started the project, I have gained a lot of experience in these areas, now I just need to develop this application and hopefully be able to get it deployed.

Reference list

agilie.com. (n.d.). *11 Best Payment Gateways to Choose for Your Android App*. [online] Available at: https://agilie.com/blog/11-best-payment-gateways-to-choose-for-your-android-app [Accessed 13 May 2022].

Android Developers. (n.d.). *Kotlin and Android*. [online] Available at: https://developer.android.com/kotlin.

Anon, (n.d.). *What Is Research Methodology In Computer Science? – TheSassWay.com*. [online] Available at: https://thesassway.com/what-is-research-methodology-in-computer-science/ [Accessed 13 May 2022].

archive.ph. (2020a). *Is it safe – and ethical – to order meals online during the coronavir…*. [online] Available at: https://archive.ph/gw6G6 [Accessed 13 May 2022].

archive.ph. (2020b). *Uber Eats Competitors, Revenue and Employees - Owler Company Profile*. [online] Available at: https://archive.ph/iP8Ss [Accessed 13 May 2022].

Coursehero.com. (2022). [online] Available at: https://www.coursehero.com/file/p41es7mk/developers-It-raised-more-than-80-Million-since-2007-in-various-investment/ [Accessed 13 May 2022].

GitHub. (n.d.). *Build software better, together*. [online] Available at: https://github.com/search?q=google+maps [Accessed 13 May 2022].

Hamilton, T. (2020). *Download Sample Test Case Template: Example Excel, Word Formats*. [online] www.guru99.com. Available at: https://www.guru99.com/download-sample-test-case-template-with-explanation-of-important-fields.html.

hanet (2020). *What is a dark kitchen?* [online] archive.ph. Available at: https://archive.ph/ap9FB [Accessed 13 May 2022].

Heroku (2020). *Cloud Application Platform | Heroku*. [online] Heroku.com. Available at: https://www.heroku.com/.

Linn, T. (2020). *How food delivery apps have changed the game for restaurants - Los An…*.

[online] archive.ph. Available at: https://archive.ph/nzjpJ.

Neumark-Sztainer, D., Larson, N.I., Fulkerson, J.A., Eisenberg, M.E. and Story, M. (2010). Family meals and adolescents: what have we learned from Project EAT (Eating Among Teens)? *Public Health Nutrition*, 13(7), pp.1113–1121. doi:10.1017/s1368980010000169.

PCMAG. (n.d.). *Definition of native application*. [online] Available at: https://www.pcmag.com/encyclopedia/term/native-application#:~:text=Native%20applications%20are%20compiled%20into [Accessed 13 May 2022].

Postmates.com. (2019). *Postmates*. [online] Available at: https://postmates.com/.

Python (2019). *Welcome to Python.org*. [online] Python.org. Available at: https://www.python.org/.

sachitanand (2020). *Why are food aggregators leveraging the delivery-only model?* [online] archive.ph. Available at: https://archive.ph/DhVHr [Accessed 13 May 2022].

Stack Overflow (n.d.). *Stack Overflow - Where Developers Learn, Share, & Build Careers*. [online] Stack Overflow. Available at: https://stackoverflow.com/.

StatCounter (2021). *Mobile Operating System Market Share Worldwide*. [online] StatCounter Global Stats. Available at: https://gs.statcounter.com/os-market-share/mobile/worldwide.

W3Schools (2019). *W3Schools Online Web Tutorials*. [online] W3schools.com. Available at: https://www.w3schools.com/.

www.postman.com. (n.d.). *Postman*. [online] Available at: https://www.postman.com/postman/workspace/postman-public-workspace/documentation/12959542-c8142d51-e97c-46b6-bd77-52bb66712c9a [Accessed 13 May 2022].

Appendices

Project Title: HubCar	
Project Supervisor: Muhammad Alam	**Contact Details:** alamm52@lsbu.ac.uk

Project Objectives (3-5 lines)

This project aims to create a mobile application that would provide users assistance with the maintenance and issues they are having with their vehicle. the aim is for the application to provide multiple solutions such as a support system where the user can input a problem and receive links to a solution and feature a browser section in which users can look at garages nearby.

Project Description (2-3 paragraphs)

The main aim of the project is to produce a mobile application that can assist users who are trying to learn about the problems they are having with their cars or to provide them with locations of garages nearby with ratings of the garage and contact information to book appointments. The application will also allow users to buy car parts and view when the part will arrive,

For the front end of the application, I will be using HTML and Bootstrap for the web application, and for the back end, python will be used. To create the mobile app for iOS swift will be needed and for android, android is needed.

The use of technology has risen, and day-to-day activities are being made simpler, HubCar will be able to assist users by providing the user with solutions all from one location rather than having to search through multiple websites and garages. If the user inputs that they need a battery for example the application will filter the garages that have batteries and will provide the user with a fixed price from a garage near to their location. As a result, this will benefit the user as they can get a reliable price, from a trusted garage through the rating system and has saved them the time of searching through multiple garages.

Project Output

- System Design and evaluation
- ☐ Comparative overview or study and evaluation framework
- System Analysis and Modelling
- Theoretical analysis, Algorithm Design, and Development

Application screenshots

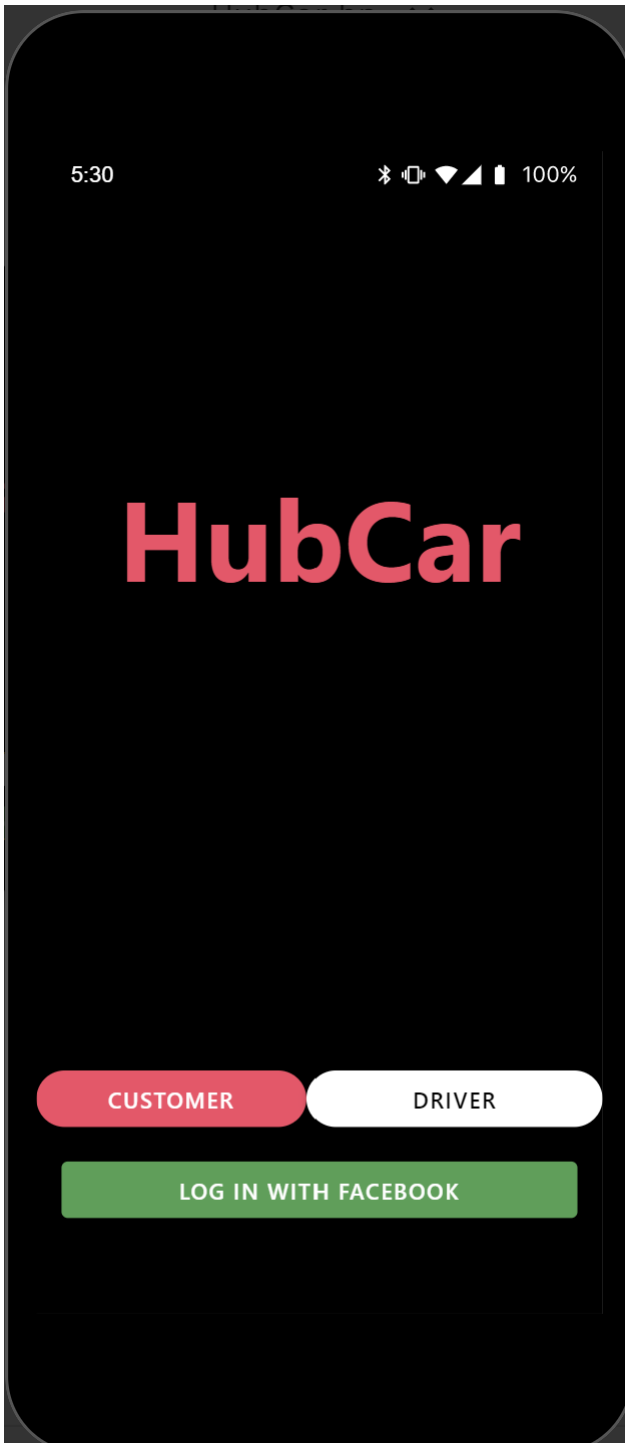


Figure 65

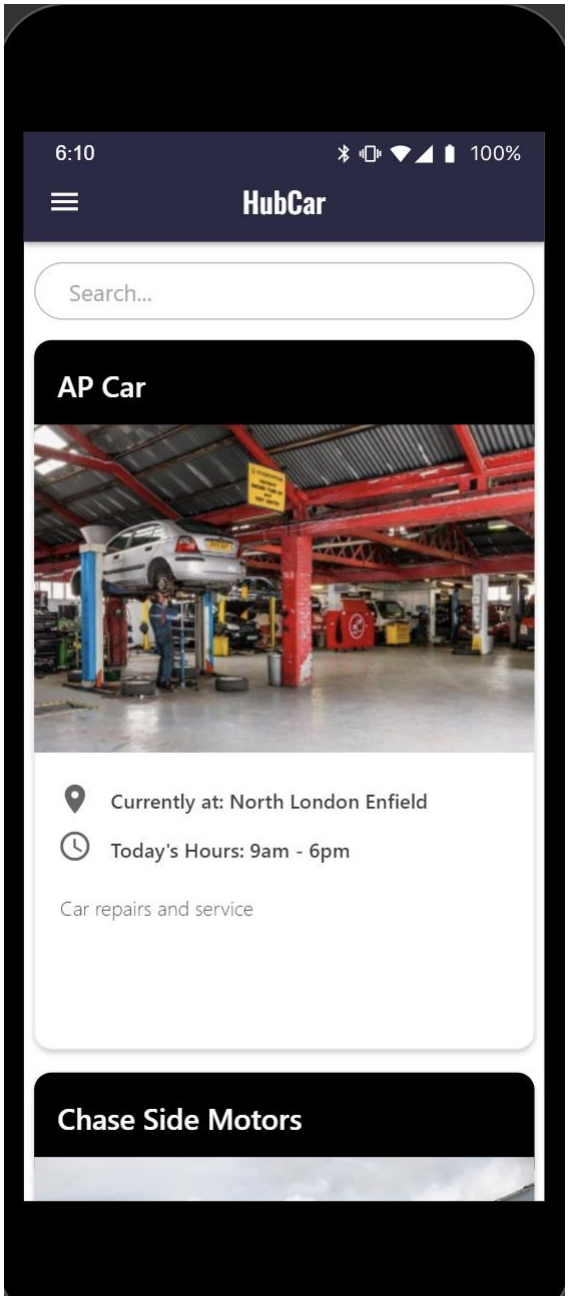

6:10
100%
HubCar
Search...
AP Car
Currently at: North London Enfield
Today's Hours: 9am - 6pm
Car repairs and service
Chase Side Motors

Figure 66

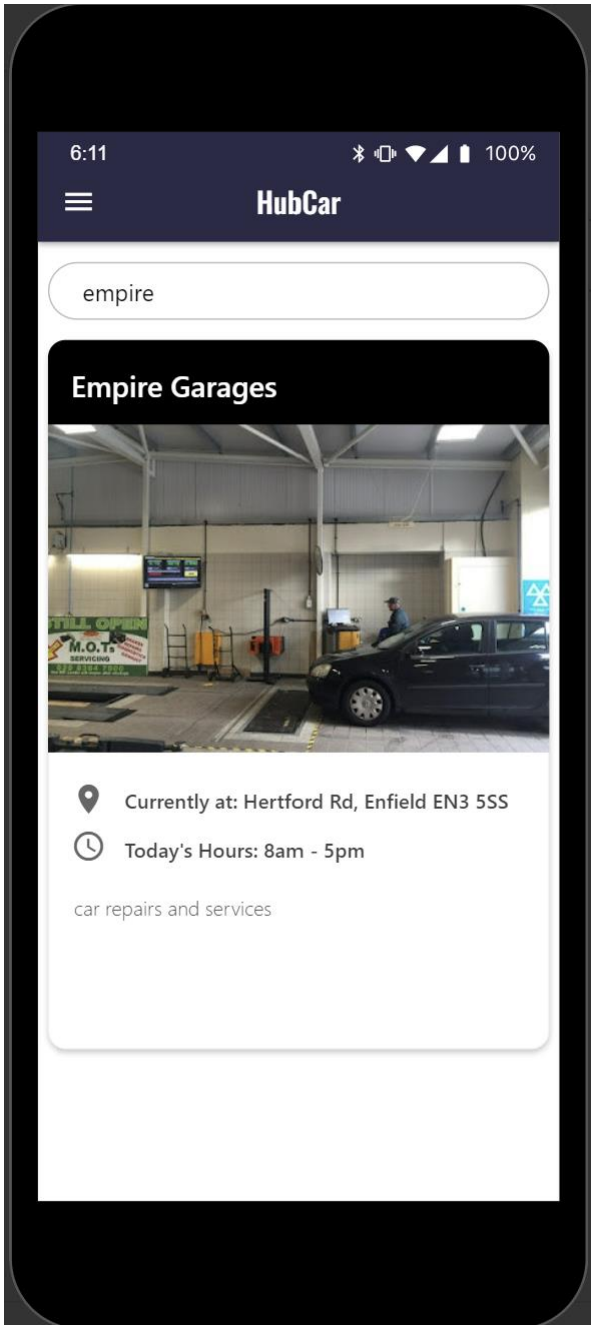

6:11
100%
HubCar
empire
Empire Garages
Currently at: Hertford Rd, Enfield EN3 5SS
Today's Hours: 8am - 5pm
car repairs and services

Figure 67

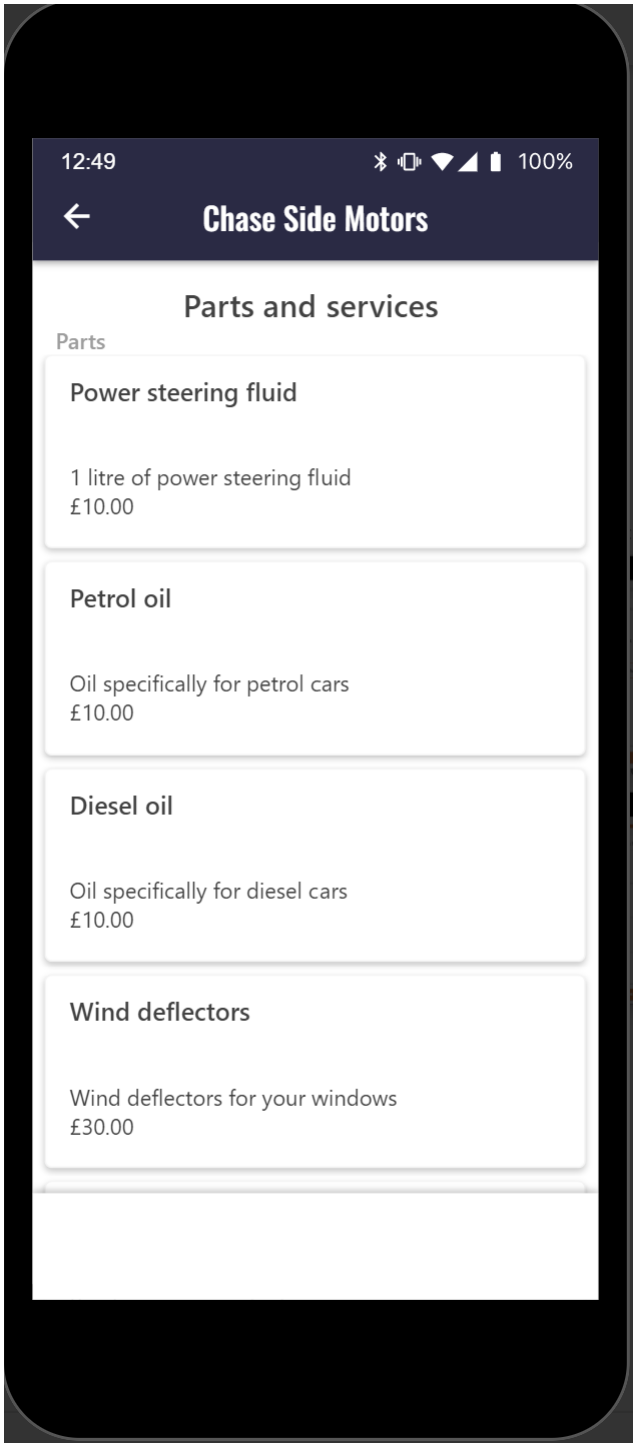

12:49
100%
Chase Side Motors
Parts and services
Parts
Power steering fluid
1 litre of power steering fluid
£10.00
Petrol oil
Oil specifically for petrol cars
£10.00
Diesel oil
Oil specifically for diesel cars
£10.00
Wind deflectors
Wind deflectors for your windows
£30.00

Figure 68


12:20
100%
RAVENOL
HYDRAULIK SERVOLENKUNG FLUID
PSF
Synthetic
Power Steering fluid
1 litre of power steering fluid
1
ADD TO CART
£10.00

Figure 69

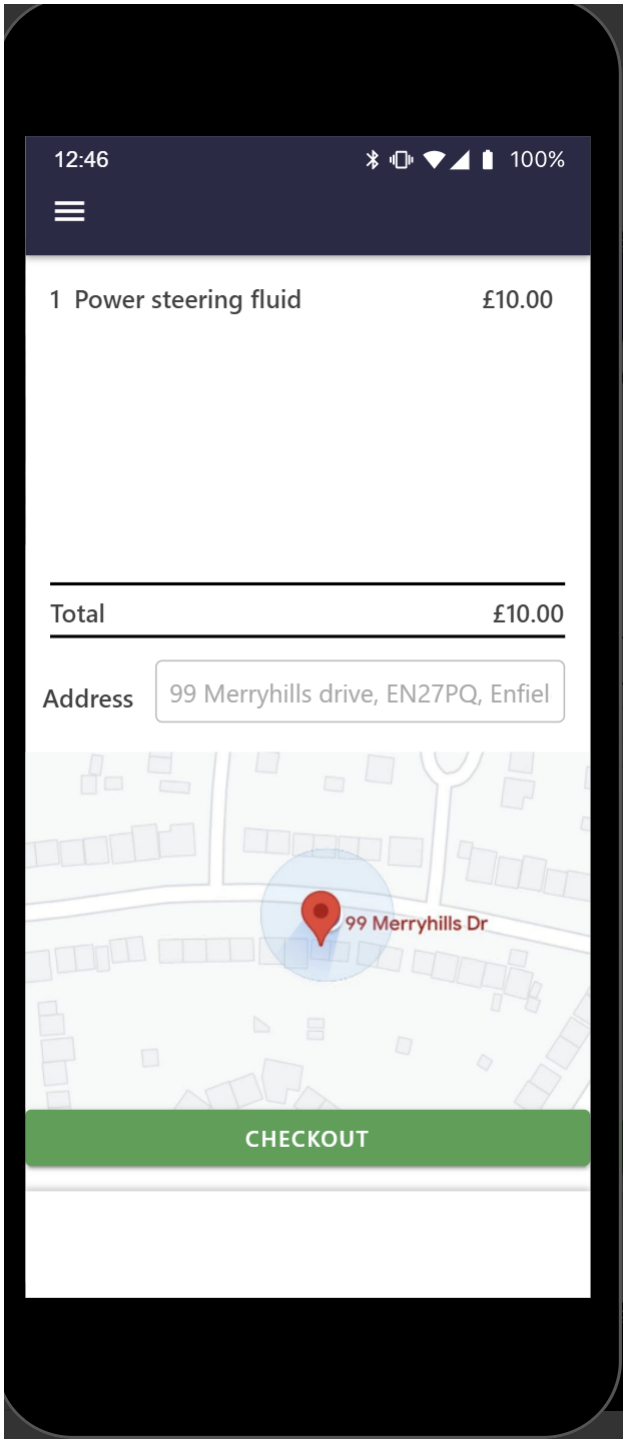

12:46
100%
1 Power steering fluid
£10.00
Total
£10.00
Address
99 Merryhills drive, EN27PQ, Enfiel
99 Merryhills Dr
CHECKOUT

Figure 70

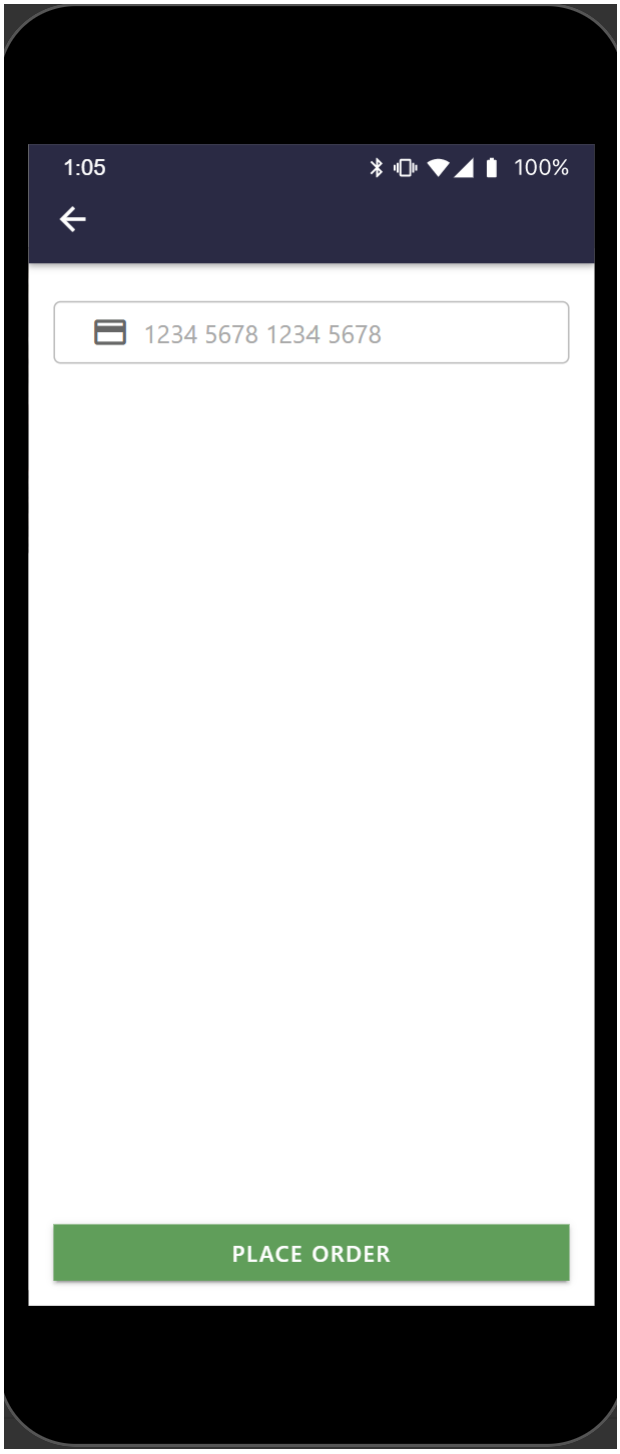


Figure 71

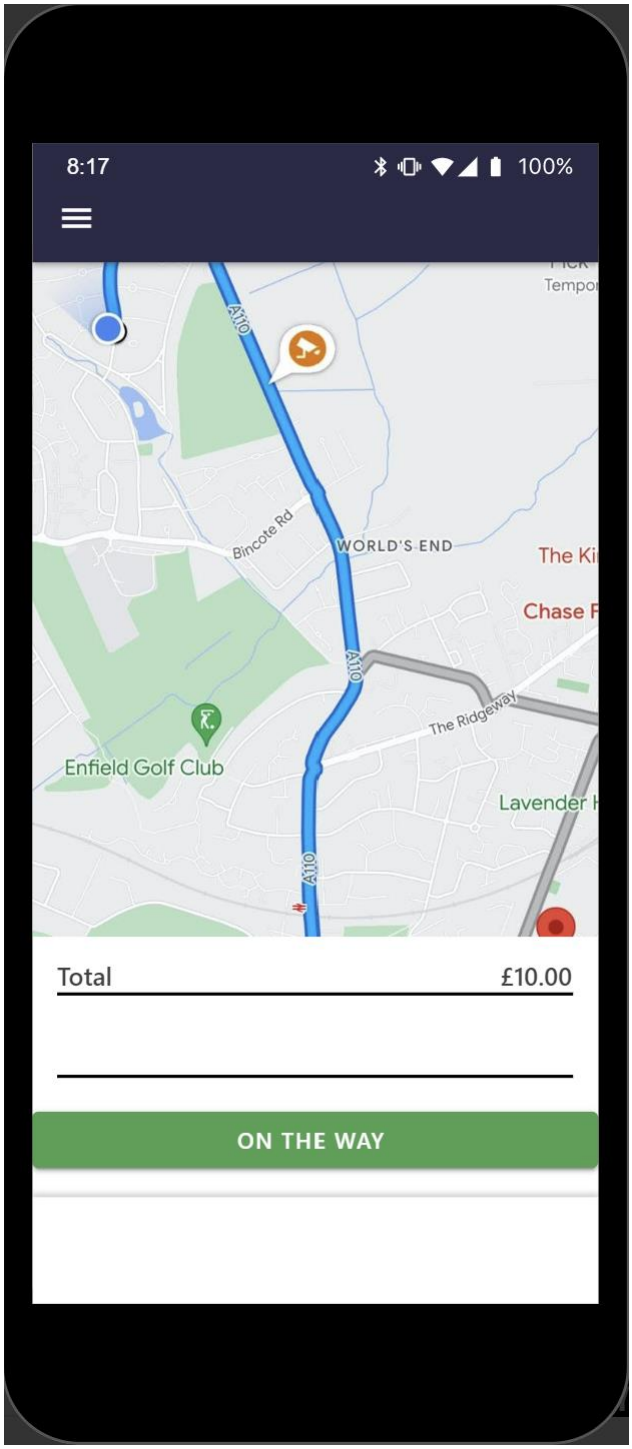

8:17
100%
A110
Bincote Rd
WORLD'S END
The Ki
Chase F
A110
The Ridgeway
Enfield Golf Club
Lavender H
A110
Total
£10.00
ON THE WAY

Figure 72

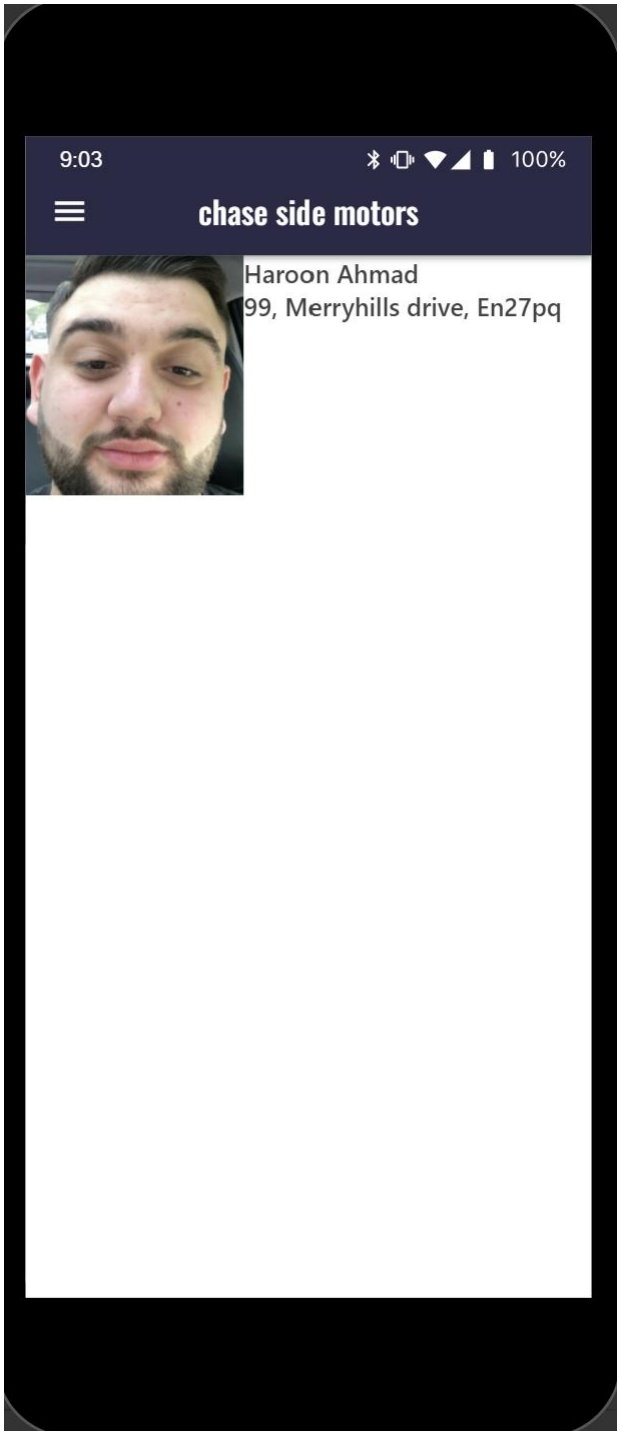

9:03
100%
chase side motors
Haroon Ahmad
99, Merryhills drive, En27pq

Figure 73

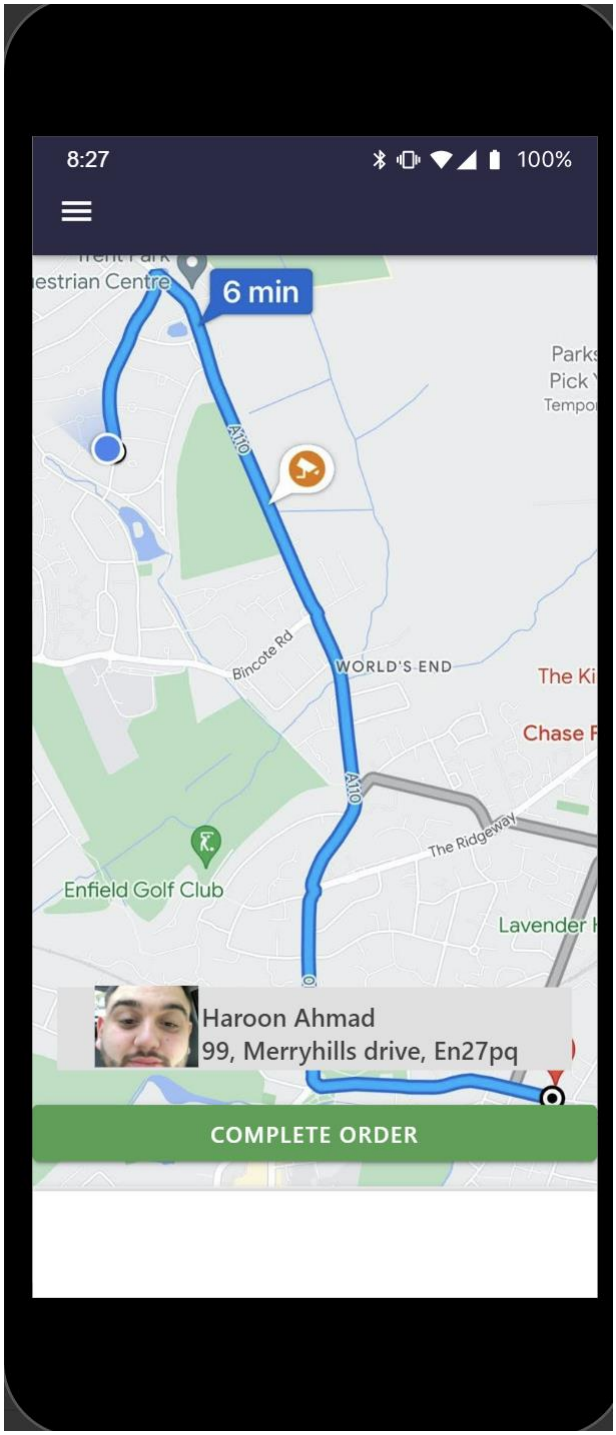

8:27
100%
6 min
A110
Bincote Rd
WORLD'S END
A110
The Ridgeway
Enfield Golf Club
Haroon Ahmad
99, Merryhills drive, En27pq
COMPLETE ORDER

Figure 74

Technical review tables

Table 5

	Kotlin	Swift
Features	• A general-purpose, class-based, object-oriented programming language, Java has been developed to have fewer implementation requirements than other languages of its kind. Application development can be done on this platform. It follows that Java is a highly secure and dependable platform since it is quick. It is widely used in laptops, data centres, game consoles, scientific supercomputers, cell phones, etc. to produce Java applications.	• Swift, backed by one of the world's most powerful tech firms, Apple, is destined to lead iOS programming. • Swift is open source • Swift pushes clean, consistent code. Swift protects against errors and improves readability. • Swift was designed for speed. Swift is up to 2x faster than Objective-C and 8x faster than Python.
Advantage	• Java's syntax is simple, making it easy to develop, learn, maintain, and understand. Many of the more complex elements of languages like Java's predecessors, such as explicit pointers and storage classes, have been eliminated from Java. This makes Java less difficult to learn than languages like C and C++. • Java lowers security risks by not using explicit pointers. A pointer stores another value's memory address, allowing unauthorized memory access. The problem is handled by eliminating pointers. Also, in Java, each application has a Security manager where we may set class access rules. • The development and maintenance of Java programs are inexpensive since these programs are dependent on a certain hardware infrastructure to execute. We can quickly execute them on any system, which eliminates the additional	• Swift is a clean, expressive language with streamlined syntax and grammar. It is simpler than Objective-C, requiring less code to complete the same operation. Automatic Reference Counting (ARC) tracks and manages memory usage for apps, saving developers time and effort. Swift makes iOS app development faster. • You receive a product that is future-proof and can be upgraded with additional features as needed. The result is easy growth. • The language was made to beat its predecessor in terms of performance and speed. • A lot of third-party code is used in app development – reusable and typically open-source frameworks or libraries. They might be static or dynamic. Static libraries, once compiled, form part of your executable file, increasing its size and loading time. As a result, they can't be automatically updated. Dynamic libraries, on the other hand, are only loaded when needed. In contrast, dynamic libraries require only one

	costs associated with maintaining them. • The Write Once Run Anywhere feature of Java is a great benefit to its users. Java's byte code is platform neutral and can run on any machine regardless of the operating system. This code can be run on any machine that supports the Java Virtual Machine. • Java is a high-level programming language that is also human-readable, making it ideal for web development. In many ways, it resembles human language, and it has a very simple and easy-to-maintain syntax that is like the syntax of the C++ programming language, but more straightforwardly. • As a result of its platform independence, Java may be used on a variety of devices. To run on any platform, Java code may be taken to any platform and can be executed on it. As a result, Java has the benefit of portability. • Java is a multithreaded language, meaning multiple threads can run simultaneously. A thread is a process's smallest unit. Multithreading maximizes CPU utilization. Multiple threads share a memory, increasing application efficiency and performance. These threads are unrelated and unaffected by each other. • When compared to programs written in other languages, Java programs are more stable. Furthermore, a new version of Java is published in a short period with more advanced capabilities, which increases the stability of the software. • Java is a distributed language that allows numerous	copy in all application files. Swift started with dynamic libraries in iOS. • Swift utilizes Automatic Reference Counting (ARC), a technology that adds a garbage collector to iOS. Garbage collectors are used in Java, C#, and Go to eliminate unused class instances. They help reduce memory footprint but add up to 20% to CPU. Swift's ARC detects unused instances and disposes of them on your behalf. It allows you to speed up your app without affecting your memory or CPU. • Server-side Swift works with the most popular backend frameworks. Using Swift for both the backend and front end of your app enables considerable code sharing and reuse, decreasing development time and costs.

	computers to share data and programs, improving system performance and efficiency. Java's distributed processing is supported by the Remote Method Invocation. For shared objects in a distributed context, Java also offers CORBA and Socket Programming. • There are two major aspects to the Java memory allocation strategy: Heap Area and Stack Area. Memory for any variable might come from the heap or the stack area of the JVM. Stack or heap space is used whenever a variable is declared by JVM.	
Disadvantage	• Java uses more memory and is slower than native languages like C or C++. It is also slower than C and C++ since each code must be interpreted as machine code. The JVM's extra level of compilation and abstraction causes poor performance. Moreover, the trash collector might slow down Java by consuming extra CPU time. • Even though there are numerous GUI builders in Java for designing graphical interfaces, it is not ideal for creating sophisticated user interfaces. While using them, there are numerous discrepancies to be aware of. • Most of Java's work is focused on data storage, not data backup. This is a major flaw that deters users from giving it a high rating and attention.	

Table 6

	Flutter	React Native	Xamarin
Features	• Flutter is an open-source mobile UI framework • Flutter apps are built with dart	• React native is an open-source framework • allows cross-platform mobile app development using JavaScript and React, a popular open-	• Can run both is and android applications • Is open-source so code can be shared

	• Flutter can communicate with other platforms without the use of a JavaScript bridge • Flutter is compiled with just-in-time	source JavaScript toolkit used by Facebook, Instagram, and others. • React Native uses JSX to build native Android and iOS apps.	
Advantages	• Quick development saves time, effort, and money. With cross-platform technology, you can utilize the same code base to construct iOS and Android apps. This speeds up the development process and eliminates the need for two teams to work on a single platform. • With Flutter's "hot reload" feature, you can make changes to the code and see the results immediately in the app preview. This way, you may simply repair errors and try out new UI elements and functionality. • With Flutter's layered architecture, you can customize everything. • Flutter also works for the web and has sufficient documentation, so you can test native controls. • Detaching UI from native controllers reduces many subtleties and faults committed by smartphone makers. Although rare, native development makes it difficult to	• As an open-source framework, React Native enables the whole developer community to access and contribute to all documentation related to the technology. With React Native, you can always reach out to other developers for advice, search for relevant information, or even aid other developers in dealing with something. • The platform's usefulness consists in boosting performance using native modules and controls. It works by interacting with native Android and iOS components and then generates code to native APIs without interference. The framework uses a separate thread from the native APIs and UIs, which improves performance. • Despite their differences, React Native actively supports both Live and Hot Reloading. Live Reloading can compile and read a coder's changes. It also sends a new file to the simulator, which starts it up immediately. After the first reload, Hot Reloading based on Hot Module Replacement (HMR) was developed. Hot Reloading has a further benefit: after saving changes in	• A sophisticated and modern IDE can be built using the framework's Visual Studio. It provides the.NET/C# infrastructure required to create high-performance native apps. • The platform also has shared code logic that can be created once and deployed across multiple platforms. These systems share components like business objects, app logic, and data access layers. • The developer can customize the User Interface layout, which is useful for building programs for certain users. • All your apps can be tested rapidly in the Xamarin Test Cloud. The cloud allows you to test any app on thousands of devices, and its reporting system lets you spot bottlenecks and fix them quickly. The test

	disregard these. With a separate UI, you can easily view all system versions.	the file, an HMR intermediary keeps the new files in the relevant places while the program runs in the background. The main benefit of using Hot Reloading is that it allows developers to inspect changes in the source code without having to recompile the program. So, if a developer has many windows open for code and app screen, he may view the effects instantaneously after making code changes. • React Native's code reuse feature helps reduce app development costs. This framework eliminates the need to develop separate iOS and Android code. Because of the React Native community's expertise, all app development organizations need a smaller team of native developers. • Developers have access to third-party plugins like JavaScript and native modules. • React Native also helps to simplify data binding in a way that protects the parent data from being changed by the child component, making apps more robust and reliable. Before making any modifications to the object, developers must first change its state. This ensures that only authorized components can be changed.	cloud offers flexibility. • The framework is great, with UI controls, styles, charts, graphs, cloud services, and other sophisticated tools to assist boost your app's functionality. • With Xamarin, a single code works on numerous platforms. You can go online in weeks because the developer simply needs C# language skills.
Disadvantages	• Flutter is a new framework. thus, it's	• Learning React Native can be difficult, especially for	• The built-in app overhead is

	not stable. A lot of concerns persist, including a lack of advanced functionality that uses operating system capabilities. Many of these features aren't supported yet, and many libraries are in pre-alpha, with limitations compared to native versions. • Dart is also new. Compared to Swift and Kotlin, it lacks functionality and the ones it has aren't well-defined. • Flutter apps are big. They take up a lot of space and time to download. • The look & feel is not identical to native solutions. Flutter doesn't make native components. With its Cupertino collection, it mimics Android's Material Design and iOS-specific components. It's noticeable with different system versions where text fields or buttons differ, but Flutter stays the same. • There are no guidelines for creating Flutter apps, which can cause issues when creating more complicated software.	new app developers who are unfamiliar with the JSX syntax extension. App developers must also know how to code native apps. Fixing discrepancies in both iOS and Android can be tough for developers who don't know both platforms. • Because React Native is a JavaScript library and an open-source framework, developers find it harder to keep their apps secure. Making apps that require more security, like banking or finance apps, requires extra caution. • React Native isn't suitable for apps that require complicated movements, screen transitions, animations, or user interaction. Because iOS and Android touch subsystems are so diverse, using a single API may be difficult. • React Native takes a long to initialize the runtime. Even on fast systems, this issue occurs because JavaScript threads take longer to begin. • React native uses JavaScript so it is not ideal for computationally heavy apps. React Native slows down these apps, as inefficient float computations make memory management and use challenging.	significant. This may impact app download time and device storage capacity. • Android, iOS, and NET developers have extensive support forums, organizations, and communities. Xamarin is newer than any of the others, hence the community must grow. A problem that arises during app development is unlikely to be solved online. • To work on this framework, you may have to import and bind Java & Objective C using C#. • Write once, code anywhere isn't accurate because some code will need to be developed for each platform. The manual UI development for each platform will take time.

	Android Studio	Visual Studio Code	XCode

Features	• Android studio is an Android IDE and is made specifically for android to provide high-quality apps and to build them efficiently. • Apply changes will allow a user to move code around and make changes while the app is running without having to restart it. • intelligent code editor helps users improve their code and will help them develop code faster as it will present to the user code completion, code analysis, and refactoring. • Android studio also partakes in the android open-source project and allows other users to upload their code.	• Multiple programming language support Web-Support used to be a separate editor for each language, but now it's built-in. Determining any errors or cross-language references is easy. • Intelli-Sense: It can detect missing code snippets. Common variable syntaxes and declarations are also automated. The application will automatically declare variables that the user has forgotten to declare. • Traditionally, editors supported either Windows, Linux, or Mac systems. But VSC is cross-platform. So it can function on all three platforms. • To utilize a language that is not supported, the user/programmer can download the extension and use it. The extension does not slow down the editor because it runs as a separate process. • Integrated support for Web-based apps is included in the package. As a result, web applications may be developed and maintained in VSC.	• It includes an Assistant Editor that displays helpful codes on the side pane, allowing you to finish your work more quickly. • Apple's developer website can be communicated using the Xcode IDE, which includes the appropriate tools.

Advantages	• Android Studio enables developers to quickly integrate changes by sending code without having to restart the app. This allows for quick app updates while the app is active. • A quicker emulator than the real device is included with Android Studio. The emulator can replicate various distinct hardware features including GPS, multiple touch inputs, motion, acceleration sensors, etc. • Android Studio provides multiple functional UI testing tools and frameworks. The features of Android Studio include powerful testing tools and frameworks for different purposes. These tools may test on real devices, emulators, or sophisticated integration environments, as well as the Firebase Test Lab. • Android Studio includes a Firebase Assistant that lets you connect any app to a Firebase server and add features like app analytics, authentication, and notification messages. Android Studio also helps connect the app to Google Cloud.	• It's very accessible because it's compatible with all systems and environments, it's free to download, and it opens quickly. • Strong native support for many languages, as well as the ability to add sophisticated language features through extensibility. • The debugging tools are extremely developed, and they assist in the placement of breakpoints, conditional breakpoints, local variables view, class variables view, immediate resolution of expressions, and the call stack for function calls, among other things. It also has the capability of supporting multi-threaded debugging. • It is small and lightweight, and it maintains large workspaces efficiently even when used with PCs with lower configurations. • There are a variety of Extensions available for various programming languages, allowing programs to be even more versatile.	• Xcode IDE helps users create apps for Apple devices by providing an Assistant Editor. This tool presents resources that it deems beneficial in the current coding process without disturbing the primary window's development and editing. This allows developers to quickly access data that can help them finish projects faster. • The Assistant Editor can help developers find information about their projects. The Jump bar presents the search result where users can immediately examine the code, so the development workflow is not disrupted. • The Xcode IDE allows developers to test apps on any device. This allows developers to check their app's quality: no more bugs, easy to use design, and overall good performance. • Also, the Xcode IDE's Interface builder allows developers to assess their interface's usability. The program creates a complete UI prototype that mimics the target environment and then tests the app in it. Users can edit their interface without quitting the graphical prototype. • The Xcode IDE provides multiple working schemes that can be

	• Android Studio has a visual drag-and-drop XML editor. This helps develop a fresh app layout quickly. The Android Studio layout editor with Constraint Layout API enables you to develop layouts for various screen sizes. This provides efficient design methods depending on device needs. • The benefits of Android Studio include powerful tools that assist automate processes, managing dependencies, and customizing the setup. The app projects may use both local and hosted libraries and are very customizable. • Android Studio provides a powerful and unified environment for developing applications for Android devices. A device-optimized experience is also possible with the tool. Android has the largest device ecosystem and benefits from device-specific programming tools like Android Studio. • Android Studio includes various example projects and code templates to help developers create new apps using tried-and-true design	• Text Editor, Keyboard shortcuts, Syntax highlighting, and the Workspace view may all be customized to your liking in great detail.	launched depending on the task at hand. Its interface starts to debug mode by default but can be altered to perform different tasks.

	principles. You may use the example code to search the web. Moreover, Android Studio enables you to import fully working applications from GitHub. • Android Studio has a strong static analysis tool. With a single click, the tool gives rapid fixes for app security, performance, and other issues. Android Studio provides applications extra capability by specifying data points in depth.		
Disadvantages	• Performance issues on older computers indicate that you should only develop on the most recent hardware. • An intense indexing and warm-up procedure lead to the opening of a project, which means that a fast look at an old project may need a few minutes of waiting time. • It is possible that initial build times may be quite lengthy, however, this has been gradually improving in subsequent versions of Android Studio.	• Customization of key combinations should be more accessible and less difficult to implement and modify. • Because VS Code does not have native integrations with legacy version control systems out of the box, you will not be able to use it. • It requires a significant amount of allocated RAM. • Some extensions are extremely buggy, and they may cause the application to crash or restart.	• Xcode feels quite outdated because it still uses objective c which is seen as outdated and slow • Unable to use multiple windows at once due to there being no feature to do so • It is very complicated to export the program once it has been made • Works with apple operating systems only

Methodology	Agile	Scrum
What is it	Agile software development is incremental and iterative. Instead of	Scrum is a popular process framework for implementing Agile. Sprints of one

	detailed planning at the start of the project, Agile methodologies encourage ongoing user feedback. Over time, cross-functional teams work on iterations of a product, which are prioritized based on business or customer value. Each iteration should result in a working product.	to two weeks allow the team to release software regularly. Towards the end of each sprint, stakeholders and team members gather. Scrum has fixed roles, responsibilities, and meetings. To structure each sprint, Scrum specifies four ceremonies: planning, daily stand-up, demo, and retrospective. At the end of each sprint, feedback is received
Advantages	• Change is welcomed, With shorter planning cycles, changes can be accommodated and accepted at any time. The backlog can always be refined and reprioritized, allowing teams to make changes in a matter of weeks. • User stories are frequently used to define product features about criteria. Each feature delivers real value by focusing on the user's needs. It allows for more beta testing of the software after each Sprint. This early feedback allows for changes to be made as needed. • Agile is great for projects where the end goal is unclear. As the project progresses, the goals will become clearer, and development can easily adapt. • Delivering in iterations allows for quality development, testing, and collaboration. Testing during each iteration helps find and fix bugs faster. And with consistent iterations, high-quality software can be delivered faster. • Agile projects encourage user feedback throughout the project, allowing lessons learned to improve future iterations.	• Daily stand-up meetings increase transparency and project visibility, removing many misunderstandings and confusion. Issues are identified ahead of time, allowing the team to resolve them. • No project manager telling the Scrum Team what to do and when. Instead, the team decides what work can be done in each sprint. • Changes are easier to cope with and accommodate with short sprints and constant feedback. New features can easily be added. • Constant communication keeps the team informed of issues and changes, lowering costs and improving quality. Smaller features allow for more frequent feedback and early correction of errors before they become costly.
Disadvantages	• It's not always easy to plan a delivery date. Some items may not be completed on time due to Agile's time-boxed delivery and project managers frequently reprioritizing tasks. Additional sprints can be added to the project at any time, adding to the timeline. • Agile works best when developers devote 100% of their time to a	• Due to the lack of a defined end date, some Scrum projects may experience scope creep. With no completion date, stakeholders may keep requesting new features. • To succeed, the team must be familiar with Scrum principles and have defined roles and responsibilities. Because Scrum Teams have no defined roles, they

	project. Agile requires more active participation and collaboration than a traditional approach. It also means that developers must commit to the entire project.	require technical expertise. The team must also commit to daily Scrum meetings and stay together throughout the project. • Tasks poorly defined can lead to errors: Without clear tasks, project costs and timelines will be off. Uncertain goals make planning difficult and sprints take longer than expected.

Table 7

Full backend source code

settings.py

hubcar > settings.py > ...

```
"""
Django settings for hubcar project.

Generated by 'django-admin startproject' using Django 1.10.

For more information on this file, see
https://docs.djangoproject.com/en/1.10/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.10/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.10/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '-jg2ngl7$$ejo!&cr7v^#yxcyggdo#bm4!op3x3m72nnf5#1l0'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
```

Figure 75

hubcar > settings.py > ...

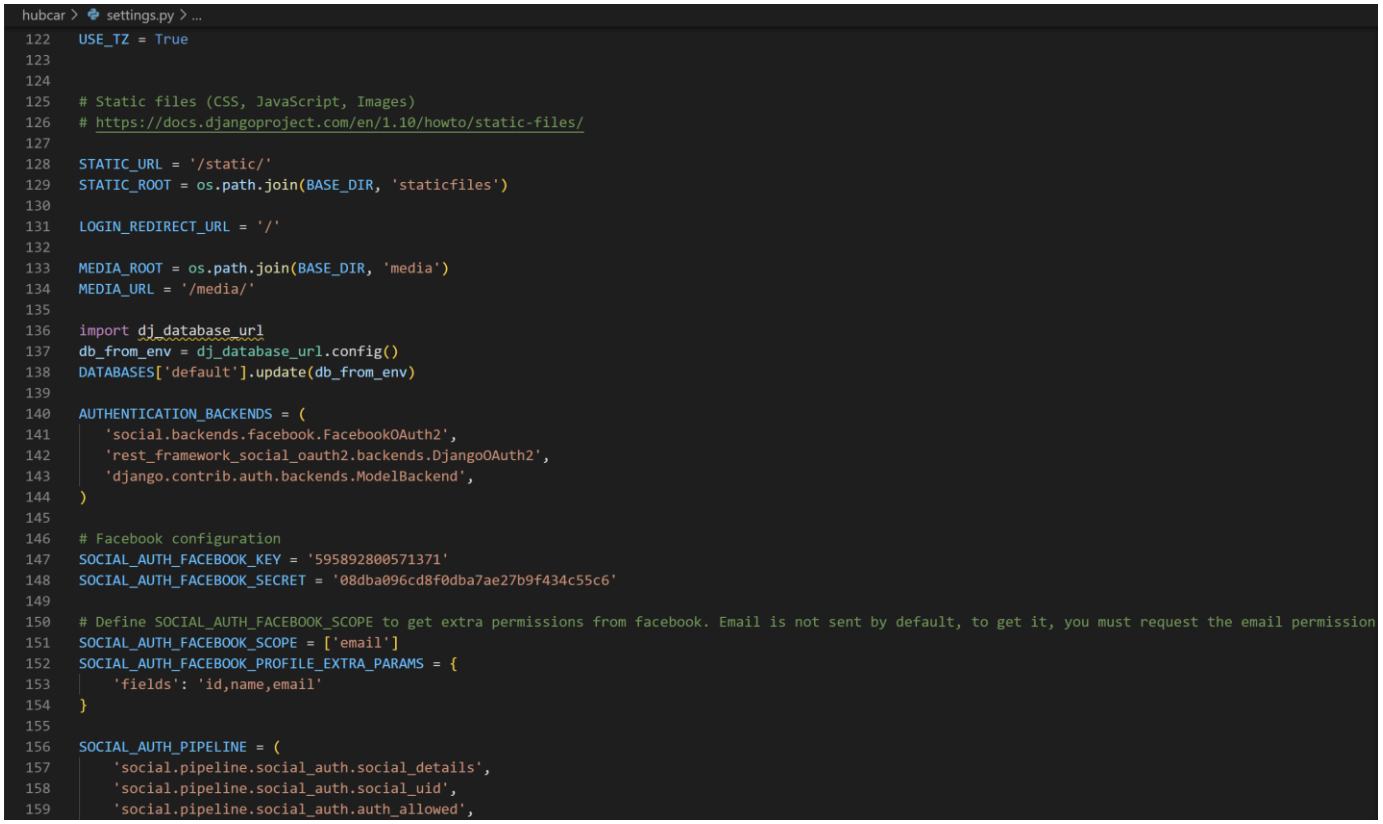
```
    'hubcarapp',
    'oauth2_provider',
    'social.apps.django_app.default',
    'rest_framework_social_oauth2',
    'bootstrap3'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'hubcar.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'django.template.context_processors.media',
                'social.apps.django_app.context_processors.backends',
                'social.apps.django_app.context_processors.login_redirect',
            ],
        },
    },
]

WSGI_APPLICATION = 'hubcar.wsgi.application'
```

Figure 76

hubcar > settings.py > ...

```
82  # https://docs.djangoproject.com/en/1.10/ref/settings/#databases
83
84  DATABASES = {
85      'default': {
86          'ENGINE': 'django.db.backends.sqlite3',
87          'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
88      }
89  }
90
91
92  # Password validation
93  # https://docs.djangoproject.com/en/1.10/ref/settings/#auth-password-validators
94
95  AUTH_PASSWORD_VALIDATORS = [
96      {
97          'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
98      },
99      {
100         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
101     },
102     {
103         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
104     },
105     {
106         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
107     },
108 ]
109
110
111 # Internationalization
112 # https://docs.djangoproject.com/en/1.10/topics/i18n/
113
114 LANGUAGE_CODE = 'en-us'
115
116 TIME_ZONE = 'UTC'
117
118 USE_I18N = True
119
120 USE_L10N = True
```

Figure 77



```
hubcar > settings.py > ...
122  USE_TZ = True
123
124
125  # Static files (CSS, JavaScript, Images)
126  # https://docs.djangoproject.com/en/1.10/howto/static-files/
127
128  STATIC_URL = '/static/'
129  STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
130
131  LOGIN_REDIRECT_URL = '/'
132
133  MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
134  MEDIA_URL = '/media/'
135
136  import dj_database_url
137  db_from_env = dj_database_url.config()
138  DATABASES['default'].update(db_from_env)
139
140  AUTHENTICATION_BACKENDS = (
141      'social.backends.facebook.FacebookOAuth2',
142      'rest_framework_social_oauth2.backends.DjangoOAuth2',
143      'django.contrib.auth.backends.ModelBackend',
144  )
145
146  # Facebook configuration
147  SOCIAL_AUTH_FACEBOOK_KEY = '595892800571371'
148  SOCIAL_AUTH_FACEBOOK_SECRET = '08dba096cd8f0dba7ae27b9f434c55c6'
149
150  # Define SOCIAL_AUTH_FACEBOOK_SCOPE to get extra permissions from facebook. Email is not sent by default, to get it, you must request the email permission
151  SOCIAL_AUTH_FACEBOOK_SCOPE = ['email']
152  SOCIAL_AUTH_FACEBOOK_PROFILE_EXTRA_PARAMS = {
153      'fields': 'id,name,email'
154  }
155
156  SOCIAL_AUTH_PIPELINE = (
157      'social.pipeline.social_auth.social_details',
158      'social.pipeline.social_auth.social_uid',
159      'social.pipeline.social_auth.auth_allowed',
```

Figure 78

```
160      'social.pipeline.social_auth.social_user',
161      'social.pipeline.user.get_username',
162      'social.pipeline.user.create_user',
163      'hubcarapp.social_auth_pipeline.create_user_by_type',  # <--- set the path to the function
164      'social.pipeline.social_auth.associate_user',
165      'social.pipeline.social_auth.load_extra_data',
166      'social.pipeline.user.user_details',
167  )
168
169  STRIPE_API_KEY = 'sk_test_vBLAUomv7lb32iEaxkP4fiQk'
```

Figure 79

Urls.py

hubcar > urls.py > ...

```python
from django.conf.urls import url, include
from django.contrib import admin
from django.contrib.auth import views as auth_views
from django.conf.urls.static import static
from django.conf import settings

from hubcarapp import views, apis

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', views.home, name='home'),

    # Garage
    url(r'^garage/sign-in/$', auth_views.login,
        {'template_name': 'garage/sign_in.html'},
        name = 'garage-sign-in'),
    url(r'^garage/sign-out', auth_views.logout,
        {'next_page': '/'},
        name = 'garage-sign-out'),
    url(r'^garage/sign-up', views.garage_sign_up,
        name = 'garage-sign-up'),
    url(r'^garage/$', views.garage_home, name = 'garage-home'),

    url(r'^garage/account/$', views.garage_account, name = 'garage-account'),
    url(r'^garage/item/$', views.garage_item, name = 'garage-item'),
    url(r'^garage/item/add/$', views.garage_add_item, name = 'garage-add-item'),
    url(r'^garage/item/edit/(?P<item_id>\d+)/$', views.garage_edit_item, name = 'garage-edit-item'),
    url(r'^garage/order/$', views.garage_order, name = 'garage-order'),
    url(r'^garage/report/$', views.garage_report, name = 'garage-report'),

    # Sign In/ Sign Up/ Sign Out
    url(r'^api/social/', include('rest_framework_social_oauth2.urls')),
    # /convert-token (sign in/ sign up)
    # /revoke-token (sign out)
    url(r'^api/garage/order/notification/(?P<last_request_time>.+)/$', apis.garage_order_notification),
```

Figure 80

hubcar > urls.py > ...

```python
    # APIs for CUSTOMERS
    url(r'^api/customer/garages/$', apis.customer_get_garages),
    url(r'^api/customer/items/(?P<garage_id>\d+)/$', apis.customer_get_items),
    url(r'^api/customer/order/add/$', apis.customer_add_order),
    url(r'^api/customer/order/latest/$', apis.customer_get_latest_order),
    url(r'^api/customer/driver/location/$', apis.customer_driver_location),


    # APIs for DRIVERS
    url(r'^api/driver/orders/ready/$', apis.driver_get_ready_orders),
    url(r'^api/driver/order/pick/$', apis.driver_pick_order),
    url(r'^api/driver/order/latest/$', apis.driver_get_latest_order),
    url(r'^api/driver/order/complete/$', apis.driver_complete_order),
    url(r'^api/driver/revenue/$', apis.driver_get_revenue),
    url(r'^api/driver/location/update/$', apis.driver_update_location),

] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Figure 81

Wsgi.py

hubcar > wsgi.py > ...

```python
"""
WSGI config for hubcar project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/1.10/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "hubcar.settings")

application = get_wsgi_application()

# Use whitenoise package to serve static files on Heroku
from whitenoise.django import DjangoWhiteNoise
application = DjangoWhiteNoise(application)

```

Figure 82

Initial.py

hubcarapp > migrations > 0001_initial.py > ...

```python
# -*- coding: utf-8 -*-
# Generated by Django 1.10 on 2016-09-12 05:07
from __future__ import unicode_literals

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    initial = True

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]

    operations = [
        migrations.CreateModel(
            name='Garage',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=500)),
                ('phone', models.CharField(max_length=500)),
                ('address', models.CharField(max_length=500)),
                ('logo', models.ImageField(upload_to='garage_logo/')),
                ('user', models.OneToOneField(on_delete=django.db.models.deletion.CASCADE, related_name='garage', to=settings.AUTH_USER_MODEL)),
            ],
        ),
    ]

```

Figure 83

Customer_driver.py

hubcarapp > migrations > 0002_customer_driver.py > ...

```python
# -*- coding: utf-8 -*-
# Generated by Django 1.10 on 2016-09-13 02:45
from __future__ import unicode_literals

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
        ('hubcarapp', '0001_initial'),
    ]

    operations = [
        migrations.CreateModel(
            name='Customer',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('avatar', models.CharField(max_length=500)),
                ('phone', models.CharField(blank=True, max_length=500)),
                ('address', models.CharField(blank=True, max_length=500)),
                ('user', models.OneToOneField(on_delete=django.db.models.deletion.CASCADE, related_name='customer', to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='Driver',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('avatar', models.CharField(max_length=500)),
                ('phone', models.CharField(blank=True, max_length=500)),
                ('address', models.CharField(blank=True, max_length=500)),
                ('user', models.OneToOneField(on_delete=django.db.models.deletion.CASCADE, related_name='driver', to=settings.AUTH_USER_MODEL)),
            ],
        ),
    ]

```

Figure 84

Item.py

hubcarapp > migrations > 0003_item.py > ...

```python
# -*- coding: utf-8 -*-
# Generated by Django 1.10 on 2016-09-13 11:54
from __future__ import unicode_literals

from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        ('hubcarapp', '0002_customer_driver'),
    ]

    operations = [
        migrations.CreateModel(
            name='Item',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=500)),
                ('short_description', models.CharField(max_length=500)),
                ('image', models.ImageField(upload_to='item_images/')),
                ('price', models.IntegerField(default=0)),
                ('garage', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='hubcarapp.Garage')),
            ],
        ),
    ]

```

Figure 85

Order.py

hubcarapp > migrations > 0004_order.py > ...

```python
# -*- coding: utf-8 -*-
# Generated by Django 1.10 on 2016-09-14 00:14
from __future__ import unicode_literals

from django.db import migrations, models
import django.db.models.deletion
import django.utils.timezone


class Migration(migrations.Migration):

    dependencies = [
        ('hubcarapp', '0003_item'),
    ]

    operations = [
        migrations.CreateModel(
            name='Order',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('address', models.CharField(max_length=500)),
                ('total', models.IntegerField()),
                ('status', models.IntegerField(choices=[(1, 'Preparing'), (2, 'Ready'), (3, 'On the way'), (4, 'Delivered')])),
                ('created_at', models.DateTimeField(default=django.utils.timezone.now)),
                ('picked_at', models.DateTimeField(blank=True, null=True)),
                ('customer', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='hubcarapp.Customer')),
                ('driver', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='hubcarapp.Driver')),
                ('garage', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='hubcarapp.Garage')),
            ],
        ),
    ]

```

Figure 86

Orderdetails.py

hubcarapp > migrations > 0005_orderdetails.py > ...

```python
# -*- coding: utf-8 -*-
# Generated by Django 1.10 on 2016-09-14 00:19
from __future__ import unicode_literals

from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        ('hubcarapp', '0004_order'),
    ]

    operations = [
        migrations.CreateModel(
            name='OrderDetails',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('quantity', models.IntegerField()),
                ('sub_total', models.IntegerField()),
                ('item', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='hubcarapp.Item')),
                ('order', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, related_name='order_details', to='hubcarapp.Order')),
            ],
        ),
    ]

```

Figure 87

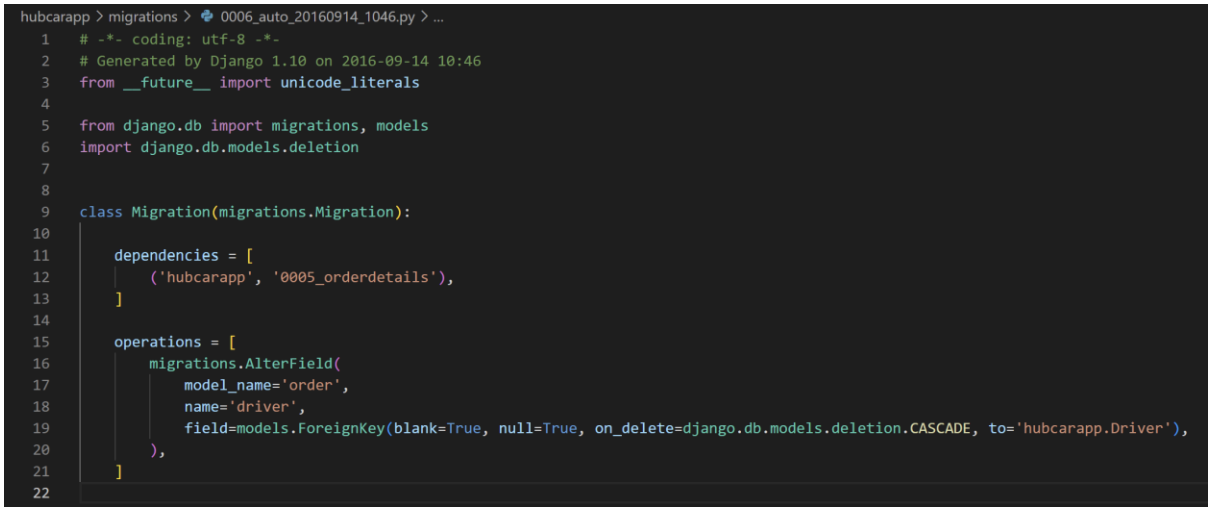


Figure 88

driver_location.py

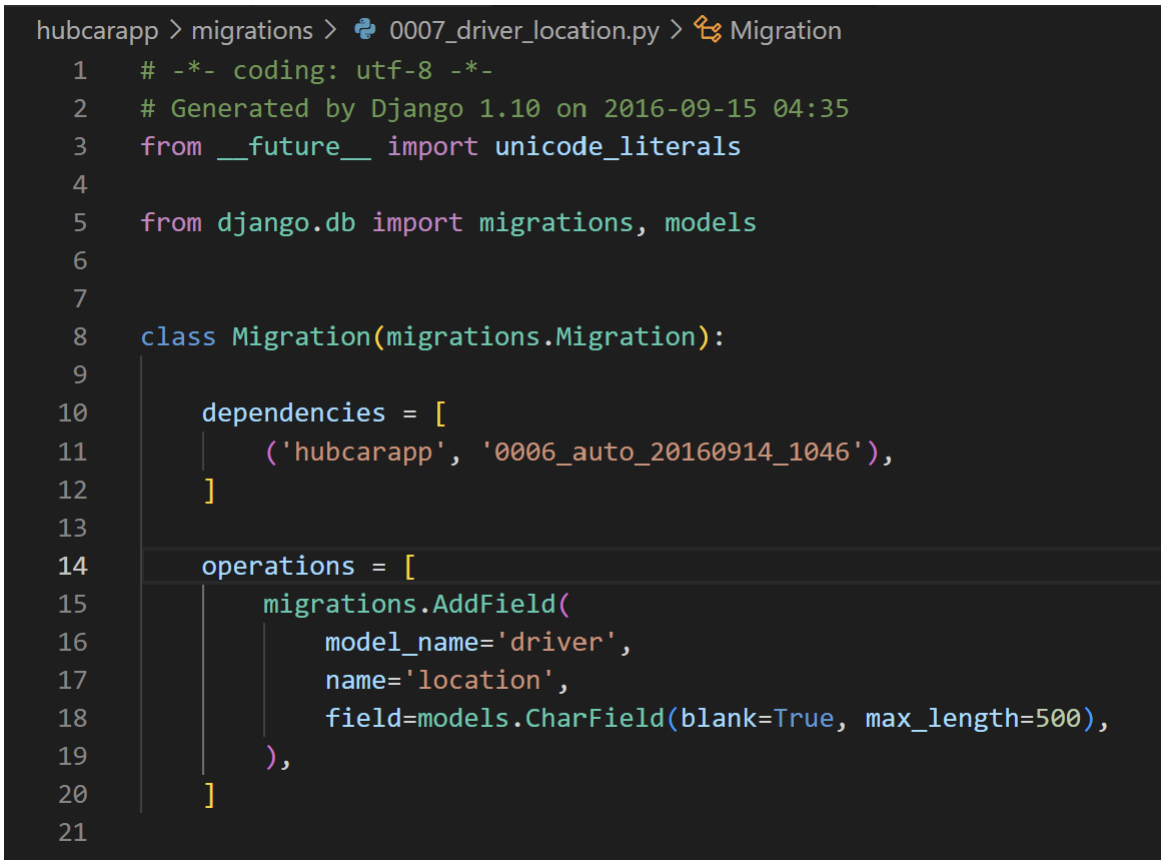


Figure 89

Style.css

hubcarapp > static > css > # style.css > .btn-pink:hover

```css
body {
  padding-top: 50px;
  background-color: #EDF2F5;
  overflow: auto;
}

.bg-blue {
  background-color: #3F3F63;
}

.bg-gray {
  background-color: #88A1B2;
}

.text-white {
  color: white;
}

.btn-pink {
  color: white;
  background-color: #EE5462;
}

.btn-pink:hover,
.btn-pink:focus {
  color: white;
  background-color: #DA4F5D;
}

/* SIDE BAR */
.sidebar {
  height: 100vh;
  position: fixed;
  top: 0;
  left: 0;
  padding: 0;
  background-color: #3F3F63;
  overflow: auto;
}
```

Figure 90

hubcarapp > static > css > # style.css > .btn-pink:hover

```
40
41  /* SIDE BAR - HEADER */
42  .sidebar > div:first-child {
43    padding: 30px 10px;
44    font-weight: bold;
45    font-size: 15px;
46    background-color: #383957;
47  }
48
49  .sidebar > div:first-child > h4 {
50    color: #e0deeb;
51  }
52
53  .sidebar > div:first-child > h5 {
54    color: #b6b4c1;
55  }
56
57  /* SIDE BAR - LIST */
58  .sidebar .list-group {
59    padding: 0;
60    background-color: #3F3F63;
61  }
62
63  .sidebar .list-group > span {
64    color: #b6b4c1;
65    display: block;
66    margin-top: 20px;
67    padding: 20px 30px;
68    font-size: 16px;
69    font-weight: 300;
70  }
71
72  .sidebar .list-group .list-group-item,
73  .sidebar .list-group .list-group-item:focus {
74    background-color: #3F3F63;
75    color: #b6b4c1;
76    border: 0;
77    padding-left: 60px;
78    padding-right: 30px;
```

Figure 91

hubcarapp > static > css > # style.css > .btn-pink:hover

```
}

.sidebar .list-group .list-group-item.active,
.sidebar .list-group .list-group-item:hover {
  background-color: #383957;
  color: #e0deeb;
}

.sidebar .list-group .list-group-item:first-child {
  border-top-left-radius: 0;
  border-top-right-radius: 0;
}

.sidebar .list-group .list-group-item .badge,
.sidebar .list-group .list-group-item.active .badge {
  background-color: #EE5462;
  color: #e0deeb;
}

/* SIDE BAR - LOGOUT */
.sidebar > div:last-child {
  padding: 30px;
}

.sidebar > div:last-child .btn {
  width: 100%;
}

/* CONTENT */
.content {
  padding: 0 40px;
}

.content .panel {
  border: none;
}

.table > thead > tr > th,
.table > thead > tr > td {
```

Figure 92

```
118     vertical-align: middle;
119   }
120
121   .table,
122   .table > thead > tr > th {
123     border-color: #88A1B2
124   }
125
```

Figure 93

Account.html

hubcarapp > templates > garage > account.html > div.col-lg-offset-2.col-lg-8 > div.panel

```
1   {% extends 'garage/base.html' %}
2   {% load bootstrap3 %}
3
4   {% block page %}
5       <div class="col-lg-offset-2 col-lg-8">
6           <div class="panel">
7               <div class="panel-heading bg-blue">
8                   <h4 class="panel-title text-center text-white">
9                       Account
10                  </h4>
11              </div>
12              <div class="panel-body">
13                  <form method="POST" enctype="multipart/form-data">
14                      {% csrf_token %}
15                      {% bootstrap_form user_form %}
16                      {% bootstrap_form garage_form %}
17                      <button type="submit" class="btn btn-pink">Update</button>
18                  </form>
19              </div>
20          </div>
21      </div>
22  {% endblock %}
23
```

Figure 94

Add_item.html

manage.py | apis.py 2 | add_item.html

hubcarapp > templates > garage > add_item.html > ...

```
1  {% extends 'garage/base.html' %}
2  {% load bootstrap3 %}
3
4  {% block page %}
5      <div class="col-lg-offset-2 col-lg-8">
6          <div class="panel">
7              <div class="panel-heading bg-blue">
8                  <h4 class="panel-title text-center text-white">
9                      Add Item
10                 </h4>
11             </div>
12             <div class="panel-body">
13                 <form method="POST" enctype="multipart/form-data">
14                     {% csrf_token %}
15                     {% bootstrap_form form %}
16                     <button type="submit" class="btn btn-pink">Create</button>
17                 </form>
18             </div>
19         </div>
20     </div>
21 {% endblock %}
22
```

Figure 95

Base.html

hubcarapp > templates > garage > base.html > script

```
{% extends 'base.html' %}

{% block title %} Garage {% endblock %}

{% block script %}
    <script>
        $(document).ready(function() {
            var now = new Date();
            setInterval(function() {
                $.ajax({
                    url: '/api/garage/order/notification/' + now.toISOString() + '/',
                    method: 'GET',
                    success: function(data) {
                        if (data['notification'] === 0) {
                            $('.badge').text('');
                        } else {
                            $('.badge').text(data['notification']);
                        }
                    }
                })
            }, 3000)
        })

    </script>
{% endblock %}

{% block sidebar %}

    {% url 'garage-order' as garage_order_url %}
    {% url 'garage-item' as garage_item_url %}
    {% url 'garage-report' as garage_report_url %}
    {% url 'garage-account' as garage_account_url %}

    <div class="text-center">
      <img class="img-circle" src="{{ MEDIA_URL }}{{ request.user.garage.logo }}"
          width="70" height="70" />
      <br/>
      <br/>
      <h4>Hi, {{ request.user.get_full_name }}</h4>
```

Figure 96

```
40        <h5>{{ request.user.garage.name }}</h5>
41      </div>
42
43      <div class="list-group">
44        <span class="text-uppercase">Dashboard</span>
45
46        <a href="{% url 'garage-order' %}"
47           class="list-group-item {% if request.path == garage_order_url %} active {% endif %} ">
48           Orders
49           <span class="badge"></span>
50        </a>
51        <a href="{% url 'garage-item' %}"
52           class="list-group-item {% if request.path == garage_item_url %} active {% endif %} ">Items</a>
53        <a href="{% url 'garage-report' %}"
54           class="list-group-item {% if request.path == garage_report_url %} active {% endif %} ">Reports</a>
55
56        <span class="text-uppercase">Profile</span>
57        <a href="{% url 'garage-account' %}"
58           class="list-group-item {% if request.path == garage_account_url %} active {% endif %} ">Account</a>
59      </div>
60
61      <div class="text-center">
62        <a href="{% url 'garage-sign-out' %}?next={{ request.path }}"
63           class="btn btn-pink">Logout</a>
64      </div>
65
66  {% endblock %}
67
```

Figure 97

Edit_item.html

manage.py | apis.py 2 | edit_item.html

hubcarapp > templates > garage > edit_item.html > ...

```
1   {% extends 'garage/base.html' %}
2   {% load bootstrap3 %}
3
4   {% block page %}
5       <div class="col-lg-offset-2 col-lg-8">
6           <div class="panel">
7               <div class="panel-heading bg-blue">
8                   <h4 class="panel-title text-center text-white">
9                       Edit Item
10                  </h4>
11              </div>
12              <div class="panel-body">
13                  <form method="POST" enctype="multipart/form-data">
14                      {% csrf_token %}
15                      {% bootstrap_form form %}
16                      <button type="submit" class="btn btn-pink">Update</button>
17                  </form>
18              </div>
19          </div>
20      </div>
21  {% endblock %}
22
```

Figure 98

Item.html

hubcarapp > templates > garage > item.html > div.panel > div.panel-body

```
{% extends 'garage/base.html' %}

{% block page %}

    <div class="panel">
        <div class="panel-heading bg-blue">
            <h4 class="panel-title text-center text-white">Items</h4>
        </div>
        <div class="panel-body">
            <div class="text-right">
                <a href="{% url 'garage-add-item' %}" class="btn btn-pink">Add Item</a>
                <br/>
                <br/>
            </div>

            <table class="table table-bordered table-hover table-striped">
                <thead>
                  <tr class="bg-gray text-white">
                    <th>Id</th>
                    <th>Name</th>
                    <th>Short Description</th>
                    <th>Price</th>
                    <th class="text-center">Image</th>
                  </tr>
                </thead>
                <tbody>
                    {% for item in items %}
                        <tr>
                          <td scope="row">{{ item.id }}</td>
                          <td><a href="{% url 'garage-edit-item' item.id %}">{{ item.name }}</a></td>
                          <td>{{ item.short_description }}</td>
                          <td>{{ item.price }}</td>
                          <td class="text-center"><img class="img-rounded"
                              src="{{ MEDIA_URL }}{{ item.image }}" width="50" height="50"></td>
                        </tr>

                    {% endfor %}
                </tbody>
            </table>
```

Figure 99

```
        </div>
    </div>



{% endblock %}

```

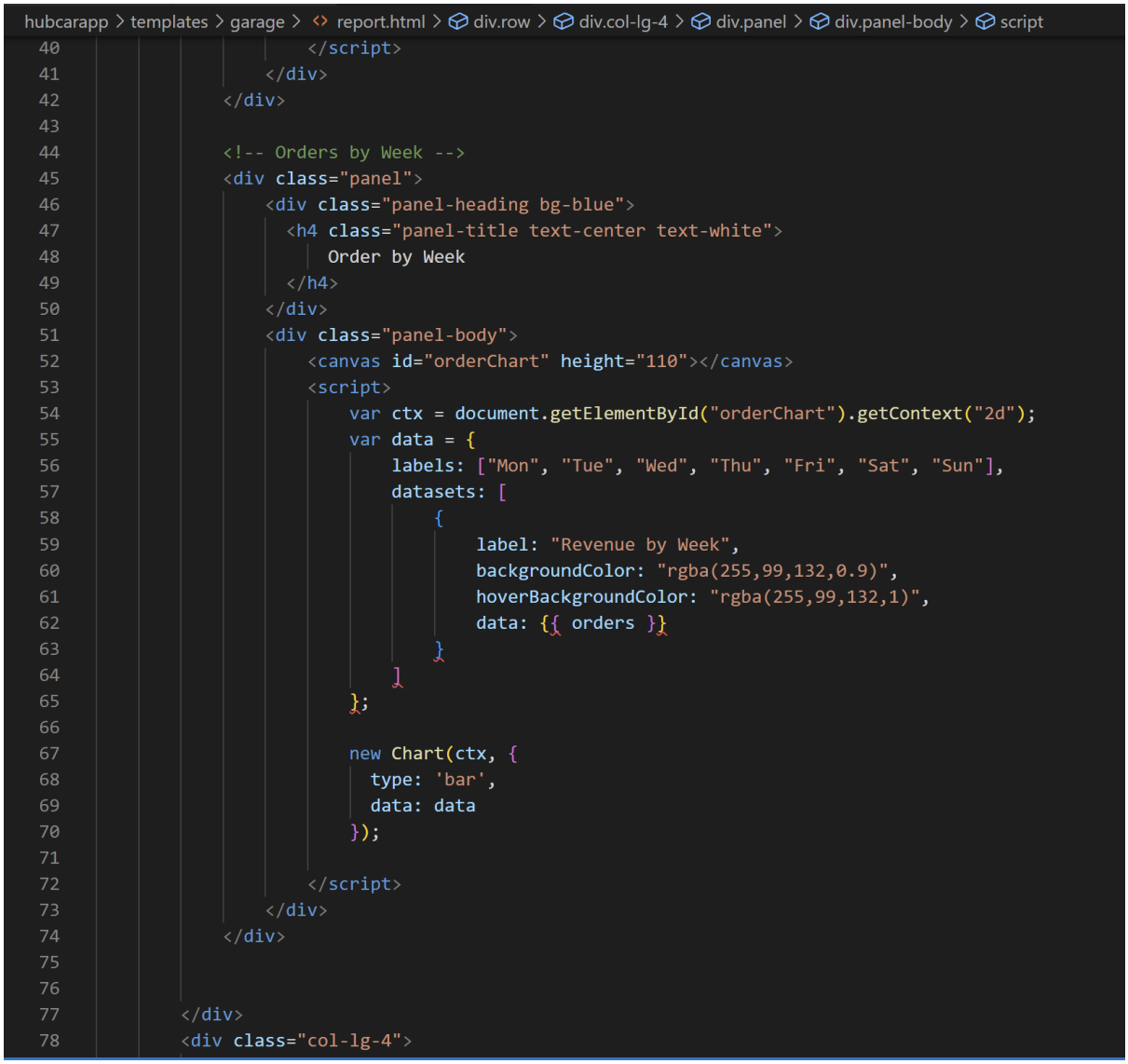
Figure 100

Report.html

hubcarapp > templates > garage > report.html > div.row > div.col-lg-4 > div.panel > div.panel-body > script

```
{% extends 'garage/base.html' %}

{% block script %}
    <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.2.2/Chart.min.js"></script>
{% endblock %}

{% block page %}

    <div class="row">
        <div class="col-lg-8">

            <!-- Revenue by Week -->
            <div class="panel">
                <div class="panel-heading bg-blue">
                  <h4 class="panel-title text-center text-white">
                      Revenue by Week
                  </h4>
                </div>
                <div class="panel-body">
                    <canvas id="revenueChart" height="110"></canvas>
                    <script>
                        var ctx = document.getElementById("revenueChart").getContext("2d");
                        var data = {
                            labels: ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
                            datasets: [
                                {
                                    label: "Revenue by Week",
                                    backgroundColor: "rgba(54,162,235,0.9)",
                                    hoverBackgroundColor: "rgba(54,162,235,1)",
                                    data: {{ revenue }}
                                }
                            ]
                        };

                        new Chart(ctx, {
                          type: 'bar',
                          data: data
                        });

```

Figure 101


hubcarapp > templates > garage > report.html > div.row > div.col-lg-4 > div.panel > div.panel-body > script
</script>

<!-- Orders by Week -->

<h4 class="panel-title text-center text-white">
Order by Week
</h4>

<canvas id="orderChart" height="110"></canvas>
<script>
var ctx = document.getElementById("orderChart").getContext("2d");
var data = {
labels: ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
datasets: [
{
label: "Revenue by Week",
backgroundColor: "rgba(255,99,132,0.9)",
hoverBackgroundColor: "rgba(255,99,132,1)",
data: {{ orders }}
}
]
};
new Chart(ctx, {
type: 'bar',
data: data
});
</script>

Figure 102


hubcarapp > templates > garage > report.html > div.row > div.col-lg-4 > div.panel > div.panel-body > scr
79
80 <!-- TOP 3 Items -->
81
82
83 <h4 class="panel-title text-center text-white">
84 Top 3 Items
85 </h4>
86
87
88 <canvas id="itemChart" height="242"></canvas>
89 <script>
90 var ctx = document.getElementById("itemChart").getContext("2d");
91 var data = {
92 labels: {{ item.labels|safe }},
93 datasets: [
94 {
95 backgroundColor: [
96 "#36A2EB", "#FFCE56", "#FF6384"
97],
98 data: {{ item.data }}
99 }
100]
101 };
102
103 new Chart(ctx, {
104 type: 'pie',
105 data: data
106 });
107
108 </script>
109
110
111
112 <!-- TOP 3 Driver ALL TIME -->
113
114
115 <h4 class="panel-title text-center text-white">
116 Top 3 Drivers
117 </h4>

Figure 103

```
                </div>
                <div class="panel-body">
                    <canvas id="driverChart" height="242"></canvas>
                    <script>
                        var ctx = document.getElementById("driverChart").getContext("2d");
                        var data = {
                            labels: {{ driver.labels|safe }},
                            datasets: [
                                {
                                    backgroundColor: [
                                      "#36A2EB", "#FFCE56", "#FF6384"
                                    ],
                                    data: {{ driver.data }}
                                }
                            ]
                        };

                        new Chart(ctx, {
                          type: 'pie',
                          data: data
                        });

                    </script>
                </div>
            </div>
        </div>
    </div>

{% endblock %}

```

Figure 104

Sign_in.html

hubcarapp > templates > garage > sign_in.html > ...

```
{% extends 'base_signup.html' %}
{% load bootstrap3 %}

{% block title %}Sign In{% endblock %}
{% block heading %}Garage - Sign In{% endblock %}

{% block content %}
    <form method="POST">
      {% csrf_token %}
      {% bootstrap_form form %}
      <button type="submit" class="btn btn-pink pull-right">Sign In</button>
    </form>
    <a href="{% url 'garage-sign-up' %}">Become a Garage</a>
{% endblock %}

```

Figure 105

Sign_up.html

hubcarapp > templates > garage > sign_up.html > ...

```
{% extends 'base_signup.html' %}
{% load bootstrap3 %}

{% block title %}Sign In{% endblock %}
{% block heading %}Garage - Sign In{% endblock %}

{% block content %}
    <form method="POST" enctype="multipart/form-data">
      {% csrf_token %}
      {% bootstrap_form user_form %}
      {% bootstrap_form garage_form %}
      <button type="submit" class="btn btn-pink pull-right">Sign Up</button>
    </form>
    <a href="{% url 'garage-sign-in' %}">Already have an account?</a>
{% endblock %}

```

Figure 106

Base_signup.html

hubcarapp > templates > base_signup.html > ...

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>{% block title %}{% endblock %}</title>

  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
  <link rel="stylesheet" href="{% static 'css/style.css' %}">

  <script src="{% static 'js/bootstrap.min.js' %}"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
<body class="bg-blue">

    <div class="container">
      <div class="row">
        <div class="col-lg-4 col-lg-offset-4">
          <br/>
          <br/>
          <br/>
          <div class="panel panel-body">
            <div class="panel-body">
              <h3 class="text-center text-uppercase"><b>{% block heading %}{% endblock %}</b></h3>
              <br/>
              {% block content %}{% endblock %}
            </div>
          </div>
        </div>
      </div>
    </div>

</body>
</html>

```

Figure 107

Base.html

hubcarapp > templates > base.html > ...

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>{% block title %}{% endblock %}</title>

  <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
  <link rel="stylesheet" href="{% static 'css/style.css' %}">

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <script src="{% static 'js/bootstrap.min.js' %}"></script>

  {% block script %}{% endblock %}
</head>
<body>

    <div class="container-fluid">
      <div class="row">
        <div class="col-lg-2 col-md-2 col-sm-2 sidebar">
            {% block sidebar %}{% endblock %}
        </div>
        <div class="col-lg-10 col-lg-offset-2 col-md-10 col-md-offset-2 col-sm-10 col-sm-offset-2 content">
            {% block page %}{% endblock %}
        </div>
      </div>
    </div>

</body>
</html>

```

Figure 108

Admin.py

hubcarapp > admin.py

```
from django.contrib import admin

# Register your models here.
from hubcarapp.models import Garage, Customer, Driver, Item, Order, OrderDetails

admin.site.register(Garage)
admin.site.register(Customer)
admin.site.register(Driver)
admin.site.register(Item)
admin.site.register(Order)
admin.site.register(OrderDetails)

```

Figure 109

Apis.py

hubcarapp > apis.py > customer_get_items

```python
import json

from django.utils import timezone
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt

from oauth2_provider.models import AccessToken

from hubcarapp.models import Garage, Item, Order, OrderDetails, Driver
from hubcarapp.serializers import GarageSerializer, \
    ItemSerializer, \
    OrderSerializer

import stripe
from hubcar.settings import STRIPE_API_KEY

stripe.api_key = STRIPE_API_KEY

###############
# CUSTOMERS
###############

def customer_get_garages(request):
    garages = GarageSerializer(
        Garage.objects.all().order_by("-id"),
        many = True,
        context = {"request": request}
    ).data

    return JsonResponse({"garages": garages})

def customer_get_items(request, garage_id):
    items = ItemSerializer(
        Item.objects.filter(garage_id = garage_id).order_by("-id"),
        many = True,
        context = {"request": request}
    ).data

    return JsonResponse({"items": items})
```

Figure 110

hubcarapp > apis.py > customer_get_items

```python

@csrf_exempt
def customer_add_order(request):
    """
        params:
            access_token
            garage_id
            address
            order_details (json format), example:
                [{"item_id": 1, "quantity": 2},{"item_id": 2, "quantity": 3}]
            stripe_token

        return:
            {"status": "success"}
    """

    if request.method == "POST":
        # Get token
        access_token = AccessToken.objects.get(token = request.POST.get("access_token"),
            expires__gt = timezone.now())

        # Get profile
        customer = access_token.user.customer

        # Get Stripe token
        stripe_token = request.POST["stripe_token"]

        # Check whether customer has any order that is not delivered
        if Order.objects.filter(customer = customer).exclude(status = Order.DELIVERED):
            return JsonResponse({"status": "failed", "error": "Your last order must be completed."})

        # Check Address
        if not request.POST["address"]:
            return JsonResponse({"status": "failed", "error": "Address is required."})

        # Get Order Details
        order_details = json.loads(request.POST["order_details"])

        order_total = 0
```

Figure 111

Customer_get_items

hubcarapp > apis.py > customer_get_items

```
79          for item in order_details:
80              order_total += Item.objects.get(id = item["item_id"]).price * item["quantity"]
81
82          if len(order_details) > 0:
83
84              # Step 1: Create a charge: this will charge customer's card
85              charge = stripe.Charge.create(
86                  amount = order_total * 100, # Amount in pounds
87                  currency = "gbp",
88                  source = stripe_token,
89                  description = "Hubcar Order"
90              )
91
92              if charge.status != "failed":
93                  # Step 2 - Create an Order
94                  order = Order.objects.create(
95                      customer = customer,
96                      garage_id = request.POST["garage_id"],
97                      total = order_total,
98                      status = Order.PREPARING,
99                      address = request.POST["address"]
100                 )
101
102                 # Step 3 - Create Order details
103                 for item in order_details:
104                     OrderDetails.objects.create(
105                         order = order,
106                         item_id = item["item_id"],
107                         quantity = item["quantity"],
108                         sub_total = Item.objects.get(id = item["item_id"]).price * item["quantity"]
109                     )
110
111                 return JsonResponse({"status": "success"})
112             else:
113                 return JsonResponse({"status": "failed", "error": "Failed connect to Stripe."})
114
115
```

Figure 112

hubcarapp > apps.py > ...

```
1   from django.apps import AppConfig
2
3
4   class HubcarappConfig(AppConfig):
5       name = 'hubcarapp'
6
```

Figure 113

Forms.py

hubcarapp > forms.py > ...

```python
from django import forms

from django.contrib.auth.models import User
from hubcarapp.models import Garage, Item

class UserForm(forms.ModelForm):
    email = forms.CharField(max_length=100, required=True)
    password = forms.CharField(widget=forms.PasswordInput())

    class Meta:
        model = User
        fields = ("username", "password", "first_name", "last_name", "email")

class UserFormForEdit(forms.ModelForm):
    email = forms.CharField(max_length=100, required=True)

    class Meta:
        model = User
        fields = ("first_name", "last_name", "email")

class GarageForm(forms.ModelForm):
    class Meta:
        model = Garage
        fields = ("name", "phone", "address", "logo")

class ItemForm(forms.ModelForm):
    class Meta:
        model = Item
        exclude = ("garage",)

```

Figure 114

Models.py

hubcarapp > models.py > Customer

```python
from django.db import models
from django.contrib.auth.models import User
from django.utils import timezone

# Create your models here.
class Garage(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='garage')
    name = models.CharField(max_length=500)
    phone = models.CharField(max_length=500)
    address = models.CharField(max_length=500)
    logo = models.ImageField(upload_to='garage_logo/', blank=False)

    def __str__(self):
        return self.name

class Customer(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='customer')
    avatar = models.CharField(max_length=500)
    phone = models.CharField(max_length=500, blank=True)
    address = models.CharField(max_length=500, blank=True)

    def __str__(self):
        return self.user.get_full_name()

class Driver(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='driver')
    avatar = models.CharField(max_length=500)
    phone = models.CharField(max_length=500, blank=True)
    address = models.CharField(max_length=500, blank=True)
    location = models.CharField(max_length=500, blank=True)

    def __str__(self):
        return self.user.get_full_name()

class Item(models.Model):
    garage = models.ForeignKey(Garage)
    name = models.CharField(max_length=500)
    short_description = models.CharField(max_length=500)
    image = models.ImageField(upload_to='item_images/', blank=False)
```

Figure 115

```
hubcarapp > models.py > Customer
40      price = models.IntegerField(default=0)
41
42      def __str__(self):
43          return self.name
44
45  class Order(models.Model):
46      PREPARING = 1
47      READY = 2
48      ONTHEWAY = 3
49      DELIVERED = 4
50
51      STATUS_CHOICES = (
52          (PREPARING, "Preparing"),
53          (READY, "Ready"),
54          (ONTHEWAY, "On the way"),
55          (DELIVERED, "Delivered"),
56      )
57
58      customer = models.ForeignKey(Customer)
59      garage = models.ForeignKey(Garage)
60      driver = models.ForeignKey(Driver, blank = True, null = True)
61      address = models.CharField(max_length=500)
62      total = models.IntegerField()
63      status = models.IntegerField(choices = STATUS_CHOICES)
64      created_at = models.DateTimeField(default = timezone.now)
65      picked_at = models.DateTimeField(blank = True, null = True)
66
67      def __str__(self):
68          return str(self.id)
69
70
71  class OrderDetails(models.Model):
72      order = models.ForeignKey(Order, related_name='order_details')
73      item = models.ForeignKey(Item)
74      quantity = models.IntegerField()
75      sub_total = models.IntegerField()
76
77      def __str__(self):
78          return str(self.id)
```

Figure 116

Serializers.py

hubcarapp > serializers.py > ...

```python
from rest_framework import serializers

from hubcarapp.models import Garage, \
    Item, \
    Customer, \
    Driver, \
    Order, \
    OrderDetails

class GarageSerializer(serializers.ModelSerializer):
    logo = serializers.SerializerMethodField()

    def get_logo(self, garage):
        request = self.context.get('request')
        logo_url = garage.logo.url
        return request.build_absolute_uri(logo_url)

    class Meta:
        model = Garage
        fields = ("id", "name", "phone", "address", "logo")

class ItemSerializer(serializers.ModelSerializer):
    image = serializers.SerializerMethodField()

    def get_image(self, item):
        request = self.context.get('request')
        image_url = item.image.url
        return request.build_absolute_uri(image_url)

    class Meta:
        model = Item
        fields = ("id", "name", "short_description", "image", "price")

# ORDER SERIALIZER
class OrderCustomerSerializer(serializers.ModelSerializer):
    name = serializers.ReadOnlyField(source="user.get_full_name")

    class Meta:
        model = Customer
```

Figure 117

```
hubcarapp > serializers.py > ...
40          fields = ("id", "name", "avatar", "phone", "address")
41
42  class OrderDriverSerializer(serializers.ModelSerializer):
43      name = serializers.ReadOnlyField(source="user.get_full_name")
44
45      class Meta:
46          model = Customer
47          fields = ("id", "name", "avatar", "phone", "address")
48
49  class OrderGarageSerializer(serializers.ModelSerializer):
50      class Meta:
51          model = Garage
52          fields = ("id", "name", "phone", "address")
53
54  class OrderItemSerializer(serializers.ModelSerializer):
55      class Meta:
56          model = Item
57          fields = ("id", "name", "price")
58
59  class OrderDetailsSerializer(serializers.ModelSerializer):
60      item = OrderItemSerializer()
61
62      class Meta:
63          model = OrderDetails
64          fields = ("id", "item", "quantity", "sub_total")
65
66  class OrderSerializer(serializers.ModelSerializer):
67      customer = OrderCustomerSerializer()
68      driver = OrderDriverSerializer()
69      garage = OrderGarageSerializer()
70      order_details = OrderDetailsSerializer(many = True)
71      status = serializers.ReadOnlyField(source = "get_status_display")
72
73      class Meta:
74          model = Order
75          fields = ("id", "customer", "garage", "driver", "order_details", "total", "status", "address")
76
```

Figure 118

Social_auth_pipeline.py

```
hubcarapp > social_auth_pipeline.py > ...
1   from hubcarapp.models import Customer, Driver
2
3   def create_user_by_type(backend, user, request, response, *args, **kwargs):
4       if backend.name == 'facebook':
5           avatar = 'https://graph.facebook.com/%s/picture?type=large' % response['id']
6
7       if request['user_type'] == "driver" and not Driver.objects.filter(user_id=user.id):
8           Driver.objects.create(user_id=user.id, avatar = avatar)
9       elif not Customer.objects.filter(user_id=user.id):
10          Customer.objects.create(user_id=user.id, avatar = avatar)
11
```

Figure 119

Views.py

hubcarapp > views.py > ...

```python
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required

from hubcarapp.forms import UserForm, GarageForm, UserFormForEdit, ItemForm
from django.contrib.auth import authenticate, login

from django.contrib.auth.models import User
from hubcarapp.models import Item, Order, Driver

from django.db.models import Sum, Count, Case, When

# Create your views here.
def home(request):
    return redirect(garage_home)

@login_required(login_url='/garage/sign-in/')
def garage_home(request):
    return redirect(garage_order)

@login_required(login_url='/garage/sign-in/')
def garage_account(request):
    user_form = UserFormForEdit(instance = request.user)
    garage_form = GarageForm(instance = request.user.garage)

    if request.method == "POST":
        user_form = UserFormForEdit(request.POST, instance = request.user)
        garage_form = GarageForm(request.POST, request.FILES, instance = request.user.garage)

        if user_form.is_valid() and garage_form.is_valid():
            user_form.save()
            garage_form.save()

    return render(request, 'garage/account.html', {
        "user_form": user_form,
        "garage_form": garage_form
    })

@login_required(login_url='/garage/sign-in/')
def garage_item(request):
```

Figure 120

```
    items = Item.objects.filter(garage = request.user.garage).order_by("-id")
    return render(request, 'garage/item.html', {"items": items})

@login_required(login_url='/garage/sign-in/')
def garage_add_item(request):
    form = ItemForm()

    if request.method == "POST":
        form = ItemForm(request.POST, request.FILES)

        if form.is_valid():
            item = form.save(commit=False)
            item.garage = request.user.garage
            item.save()
            return redirect(garage_item)

    return render(request, 'garage/add_item.html', {
        "form": form
    })

@login_required(login_url='/garage/sign-in/')
def garage_edit_item(request, item_id):
    form = ItemForm(instance = Item.objects.get(id = item_id))

    if request.method == "POST":
        form = ItemForm(request.POST, request.FILES, instance = Item.objects.get(id = item_id))

        if form.is_valid():
            form.save()
            return redirect(garage_item)

    return render(request, 'garage/edit_item.html', {
        "form": form
    })


@login_required(login_url='/garage/sign-in/')
def garage_order(request):
    if request.method == "POST":
```

Figure 121

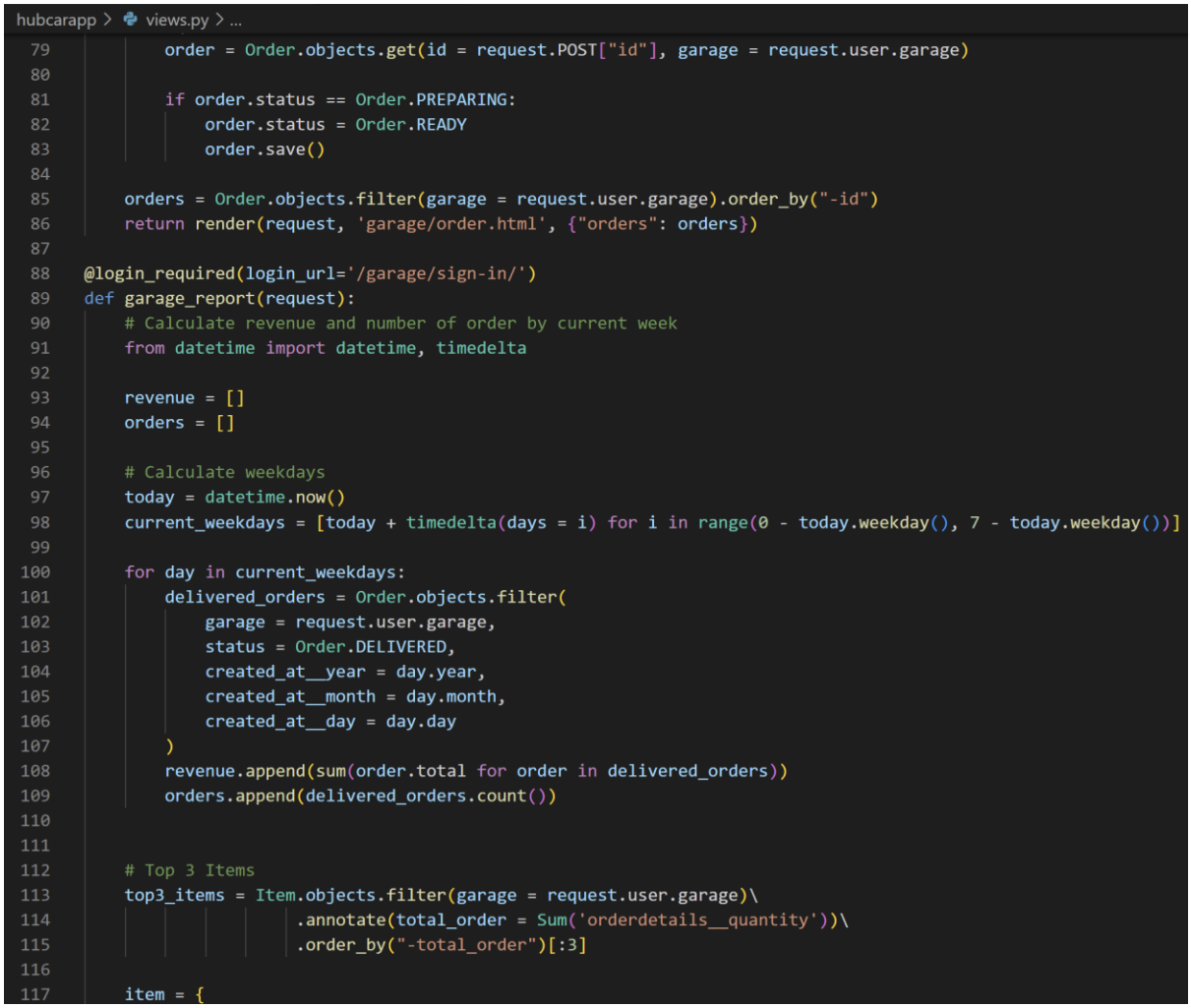

hubcarapp > views.py > ...
79 order = Order.objects.get(id = request.POST["id"], garage = request.user.garage)
80
81 if order.status == Order.PREPARING:
82 order.status = Order.READY
83 order.save()
84
85 orders = Order.objects.filter(garage = request.user.garage).order_by("-id")
86 return render(request, 'garage/order.html', {"orders": orders})
87
88 @login_required(login_url='/garage/sign-in/')
89 def garage_report(request):
90 # Calculate revenue and number of order by current week
91 from datetime import datetime, timedelta
92
93 revenue = []
94 orders = []
95
96 # Calculate weekdays
97 today = datetime.now()
98 current_weekdays = [today + timedelta(days = i) for i in range(0 - today.weekday(), 7 - today.weekday())]
99
100 for day in current_weekdays:
101 delivered_orders = Order.objects.filter(
102 garage = request.user.garage,
103 status = Order.DELIVERED,
104 created_at__year = day.year,
105 created_at__month = day.month,
106 created_at__day = day.day
107)
108 revenue.append(sum(order.total for order in delivered_orders))
109 orders.append(delivered_orders.count())
110
111
112 # Top 3 Items
113 top3_items = Item.objects.filter(garage = request.user.garage)\
114 .annotate(total_order = Sum('orderdetails__quantity'))\
115 .order_by("-total_order")[:3]
116
117 item = {

Figure 122


hubcarapp > views.py > ...
"labels": [item.name for item in top3_items],
"data": [item.total_order or 0 for item in top3_items]
}
# Top 3 Drivers
top3_drivers = Driver.objects.annotate(
total_order = Count(
Case (
When(order__garage = request.user.garage, then = 1)
)
)
).order_by("-total_order")[:3]
driver = {
"labels": [driver.user.get_full_name() for driver in top3_drivers],
"data": [driver.total_order for driver in top3_drivers]
}
return render(request, 'garage/report.html', {
"revenue": revenue,
"orders": orders,
"item": item,
"driver": driver
})
def garage_sign_up(request):
user_form = UserForm()
garage_form = GarageForm()
if request.method == "POST":
user_form = UserForm(request.POST)
garage_form = GarageForm(request.POST, request.FILES)
if user_form.is_valid() and garage_form.is_valid():
new_user = User.objects.create_user(**user_form.cleaned_data)
new_garage = garage_form.save(commit=False)
new_garage.user = new_user
new_garage.save()

Figure 123

```
hubcarapp > views.py > ...
157                 login(request, authenticate(
158                     username = user_form.cleaned_data["username"],
159                     password = user_form.cleaned_data["password"]
160                 ))
161
162                 return redirect(garage_home)
163
164         return render(request, "garage/sign_up.html", {
165             "user_form": user_form,
166             "garage_form": garage_form
167         })
168
```

Figure 124

Manage.py

```
manage.py
1   #!/usr/bin/env python
2   import os
3   import sys
4
5   if __name__ == "__main__":
6       os.environ.setdefault("DJANGO_SETTINGS_MODULE", "hubcar.settings")
7       try:
8           from django.core.management import execute_from_command_line
9       except ImportError:
10          # The above import may fail for some other reason. Ensure that the
11          # issue is really that Django is missing to avoid masking other
12          # exceptions on Python 2.
13          try:
14              import django
15          except ImportError:
16              raise ImportError(
17                  "Couldn't import Django. Are you sure it's installed and "
18                  "available on your PYTHONPATH environment variable? Did you "
19                  "forget to activate a virtual environment?"
20              )
21          raise
22      execute_from_command_line(sys.argv)
23
```

Figure 125

London South Bank University

EST 1892

School of Engineering

BSc FINAL YEAR PROJECT

ETHICAL CONSIDERATIONS

2021-22

YOUR DETAILS

Name of student: Haroon Ahmad_____

Supervisor: Alam Muhammad_____

Project title: HubCar_____

Main aim of project: Create an android application for new vehicle users that will have a ecommerce section for garages and mechanics.

_____

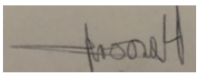
CONTACT WITH OTHERS

Will your project bring you into contact with other people (e.g. via an online survey)?

Yes ☐ No ☐

If you answered "No", sign the section below and submit this page only to your supervisor for countersigning, otherwise complete the whole form prior to submission. Also, if you answered "No" then only this page needs to be included as an appendix to your dissertation.

YOUR SIGNATURE

Signature:  Date: 13/05/2022

ETHICAL APROVAL

(To be completed by your supervisor)

I have checked the above for accuracy and I am satisfied that the information provided is an accurate reflection of the intended study.

There are no ethical issues causing my concern ☐

Signature: _____ Date: _____

Name (please print): _____

RESEARCH PARTICIPANTS

Estimated number of research participants: _____

Who will the research participants be? _____

On the basis of which criteria will they be selected? _____

Where and how will data collection take place (e.g. location, by phone/internet, by post)?

_____

Will any of the research participants be minors (under 18)?

Yes ☐ No ☐

If the answer is "Yes" please note the following:

You will need to obtain a disclosure at your own cost:
https://www.gov.uk/government/organisations/disclosure-and-barring-service

This ethics form, to which a copy of your disclosure must be attached, will need to be countersigned by a member of the School's ethics committee.

Please tick the box to indicate that you have understood these requirements ☐

METHODS OF DATA COLLECTION TO BE EMPLOYED

Paper-based questionnaire	Yes ☐	No ☐
Online questionnaire	Yes ☐	No ☐
Interview(s)	Yes ☐	No ☐
Focus group(s)	Yes ☐	No ☐
Other (please specify):		

CONFIDENTIALITY

Requests for confidentiality can only be accepted on the understanding that the confidential material will not be included in the written report/dissertation. They cannot be accepted on the understanding that the report/dissertation itself will be confidential. British universities are public bodies, and all research carried out at undergraduate/taught Masters level is by definition available for public scrutiny.

Please tick the box to indicate that you understand the limits to confidentiality in relation to your research ☐

POTENTIAL PATICIPANT DISTRESS

Is there any possibility that any of the procedures in your study will cause discomfort, anxiety, stress or embarrassment for your participants? Yes ☐ No ☐

If you have answered "Yes" please explain the details, and justify how you will seek to minimise any form of upset to your participants. Please use a separate sheet of paper, to be attached.

PLEASE INDICATE YOUR RESPONSE TO THE FOLLOWING QUESTIONS, AND DISCUSS YOUR RESPONSE WITH YOUR SUPERVISOR

Will you provide an oral explanation of the project to the participants? Yes ☐ No ☐

Will you provide a written explanation of the project to the participants? Yes ☐ No ☐

Will you ask the participants to complete a consent form? Yes ☐ No ☐

If your answer is "Yes" to any of the above please attach appropriate drafts.

If you are seeking ***verbal consent*** please provide a statement of how you will seek to gain this from participants: _____

Will you explain to the participants that you are a student undertaking degree level studies?

Yes ☐ No ☐

Will you explain to the participants that they may not benefit from your study?

Yes ☐ No ☐

Will you offer the participants the opportunity to decline to take part? Yes ☐ No ☐

Will you offer the participants the opportunity to withdraw at any stage? Yes ☐ No ☐

Will you offer anonymity? Yes ☐ No ☐

Will you adhere to the provisions of the Data Protection Act 1998? Yes ☐ No ☐

Please briefly explain (on a separate sheet of paper, to be attached) how you will address the following:

- Data may only be used for the specific purposes for which it was collected.
- Data must not be disclosed to other parties without the consent of the individual to whom it relates, unless there is legislation or other overriding legitimate reason to share the information (for example, the prevention or detection of crime). It is an offence for other parties to obtain this data without authorisation.
- There must be no identifiable personal data relating to individuals anywhere in the dissertation, either within the main body or the appendices.
- Individuals have a right of access to the information held about them, subject to certain exceptions (for example, information held for the prevention or detection of crime).
- Personal information may not be kept for any longer than is necessary, and it must be kept up to date.
- Subjects have the right to have factually incorrect information corrected (note: this does not extend to matters of opinion).

In short, all data regarding or provided by research participants must be stored securely on password-protected and encrypted computers, memory sticks and so on, must not be divulged to third parties without the research participants' consent. Furthermore, data relating to any individual must be made available to that individual on request and corrected where wrong, and it must be destroyed when no longer required.

YOUR OWN SAFETY

Are there any aspects of your project which might have implications for your own safety?

Yes ☐ No ☐

If the answer is "Yes" please indicate how you propose to minimise any risk to yourself:

CHECKLIST

Please indicate the supporting documents you are submitting along with this application:

Participants' explanation form	Yes ☐	No ☐	N/A ☐
Consent form (draft acceptable)	Yes ☐	No ☐	N/A ☐
Questionnaire/survey text (draft acceptable)	Yes ☐	No ☐	N/A ☐
Disclosure (if any participants are under 18)	Yes ☐	No ☐	N/A ☐

YOUR SIGNATURE

Signature: _____ Date: _____

Please write your name clearly in full here: _____

ETHICAL APPROVAL

(To be completed by supervisor)

I have checked the above for accuracy and I am satisfied that the information provided is an accurate reflection of the intended study. I can confirm that the form contains all the relevant information and that all the supporting documentation is attached.

There are no ethical issues causing me concern ☐

The ethical issues raised by this project require further consideration ☐

Signature: _____ Date: _____

Name (please print): _____

If you have ticked the second option please liaise with your representative on the School's ethics committee.

Appendix – further reading

Here are some good sources of information about ethical issues that may arise in the context of computing-related projects and careers (in no particular order):

- http://www.bcs.org/category/6030
- http://www.theiet.org/about/governance/rules-conduct/index.cfm
- http://www.acm.org/about/se-code
- http://www.ibm.com/developerworks/rational/library/may06/pollice/
- https://www.computer.org/cms/Computer.org/Publications/code-of-ethics.pdf