# Report

Coursework: Part 1

Car Dealership

U2026971

https://github.com/samwright000/ccse-cardealer/tree/main/cardealershippython

# Content

# High-level Business Requirements

1. Customers should be able to browse a selection of cars
2. Customers should be able to purchase car then display closest dealer
3. Customers should be able to apply for finance
4. Customers should be able to upload personal documents
5. Customers should only be able to access their account
6. For admin accounts multi factor authentication should be enabled.
7. Admin should be able to assess the internal administration module via a button on webpage
8. The finance team can keep track of inventory, number of cars sold, customer finance application and customer documents.

# Functional and Non-functional Requirements

## Functional

### Customer

1. Should be able to view cars and their details (model, series, price, picture, description.) which are in stock
2. Shouldn’t be able to view cars which aren’t in stock
3. Should be able to sign up for an account
4. Should be able to sign in to just their account
5. Should be able to add available cars to their baskets
6. Should be able to remove cars from their basket
7. Should be able to select a car from their basket for purchase
8. Once a car is selected for purchase can choose whether to purchase with one time payment or apply for a finance plan.
9. Should display the closest dealership when making a one time payment
10. Should be able to enter card number, CVV, card expiration date, delivery address including house number, road name, city, county and postcode when purchasing.
11. Should be able to view the price of the car when buying
12. Should be able to upload pdf and pictures to their account
13. Should be able to remove uploaded documents
14. Should be able to view all the order and its related information (order date, delivery address, price of the car, how much the car is paid off, the status of the order, car image) they have bought and are currently buying.
15. Should only be able to make payments to cars which aren’t yet paid off which have been approved.

### Administrator

16. Should be able to view car information such as id, series, model, description, price, image, stock, number of cars sold.
17. Should be able to add a car to the database
18. Should be able to edit the current cars
19. Should be able to remove cars

20. Should be able to view all users and their orders information
21. Should be able to edit a users details
22. Should be able to edit the status of an order from pending, ongoing and paid.
23. Admins should be able to navigate to an administration module which is only available to assigned users roles.

## Non-functional

1. Multiple users should be able to access the service at one
2. Customers should only be able to access their own data.
3. The system should be available at all times except when under maintenances
4. The system should store sensitive information securely
5. To access the admin account an email two factor authentication must occur
6. The system should be able to store many types of user data from text to pdf and images.

# How The High-level Business Requirements Were Mapped To Functional And Non-functional Requirements

Below is a table which shows how the business requirements map to the function and non-functional requirements.

| High-level business requirement | Functional requirements | How? |
|---|---|---|
| 1 | 1, 2 | Function requirement 1 allows users to view the available cars for purchase. Requirement 2 results in cars which aren't available and not possible for purchase. |
| 2 | 3, 4, 5, 6, 7, 8, 9, 10, 11 | These functional requirements all enable business requirement 2 to be carried out. A user needs an account and needs to be able to access that account to make purchases. A user needs to be able to add cars to their basket and then select them. The users need to be able to input their bank details and upload documents for the payment to be approved. |
| 3 | 8, 10, 11, 15 | These function requirements are also needed for when a user wants to apply for finances. They need to be able to upload their documents which then are reviewed by the financial department. |
| 4 | 12, 13 | As part of business requirements to apply for finance they also need to be able to upload relevant documents. This is a partial repeat of business requirements 2 and 3, it is explicitly mentioned to do this via uploading documents. |
| 5 | 14, 13 | To make sure the company remains GDPR (GDPR, 2018) the system needs to make sure user data is available for them to access. They also need to be able to remove data. |
| 7 | 16, 17, 18, 18, 20, 21 | The administration module is a large section of the project. The admins need to be able to manage the cars in ways outlined in function requirements. |
| 8 | 16, 20, 22 | The finance team needs to be able to access users data so they are able to make a judgement if finance payment is allowed. They also need to be able to keep track of the orders and stock. This is outlined in the function requirements |

Figure 1.1 - Mapping Business Requirements to Functional Requirements

| High-level business requirement | Non Functional requirements | How? |
|---|---|---|
| 1 | 1, 3, | We want many customers to be able to browse the website and use the services at once, therefore one of our functional requirements is that many users should be able to access the system.<br><br>As well as many people being able to access the website, we want people to be able to access it at any time. Not everyone looks at cars at the same time and if there is hope to grow the company internationally different countries have different time zones. |
| 2 | 1, 3 | *Same as above* |
| 3 | 1, 3 | *Same as above* |
| 4 | 1, 3, 4, 6 | *Same as above*. Additionally, the user needs to be able to enter text for form inputs but also needs to be able to upload documents relating to their finances. Therefore the system needs to be able to store this type of data. The data the user will be uploading is going to be sensitive, private financial data as well as passwords, therefore this needs to be held securely. One example of this is hashed passwords. |
| 5 | 2, 3, 4, 6 | The user should only be able to access their data. This is so that good security practices are followed but also so it complies with GDPR (GDPR, 2018) making sure users information is available when they need it. |
| 7 | 3, | Admins may need to access the website at all times of the day therefore it should always be available. |
| 8 | 3, 5 | To maintain security, the administrator users have to use two factor authentication via their email address. This email address is the one they signed up with and use to sign in with. |

Figure 1.2 - Mapping Business Requirements to Non-Functional Requirements

# Explain How These Requirements Were Delivered Using The Agile Software Development Methodology

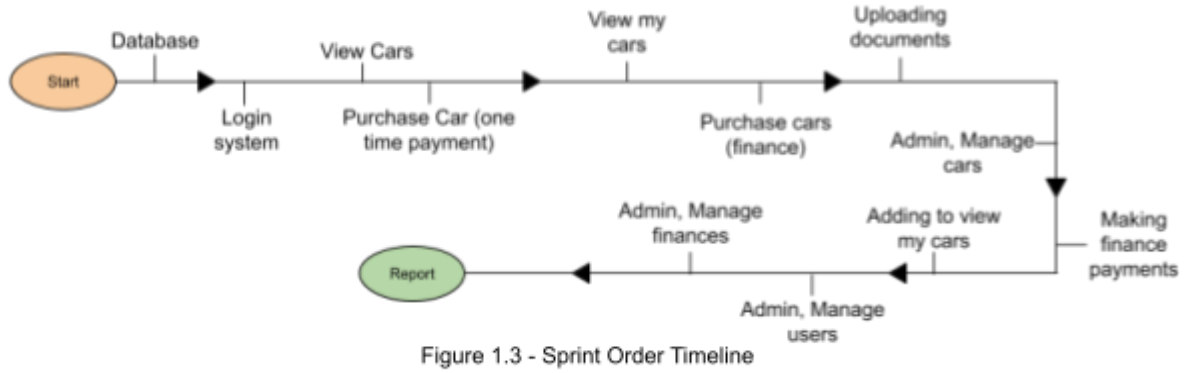


Figure 1.3 - Sprint Order Timeline

An agile software development is an iterative process that puts flexibility, collaboration and customer satisfaction as its priorities Codecademy (2019). Agile is both incremental and iterative and is made up of sprints. A sprint is a smaller, more manageable chunk of a problem (PaulPetrus, 2020). A sprint is usually a short period of time between one week and a month. For my project they were usually a few days long to a week. Each time a sprint starts it has planning, implementation then a review and testing (PaulPetrus, 2020). This is how I developed each of my sprints. Unlike a waterfall, it does this for each of its sprints rather than the whole project at large.

When I was developing my project, I broke it down into sprints. See figure 1.3 for the order in which the sprint was completed. Each sprint followed the processes outlined in figure 1.5 below. I also partially created user stories as seen in figure 1.4 below.

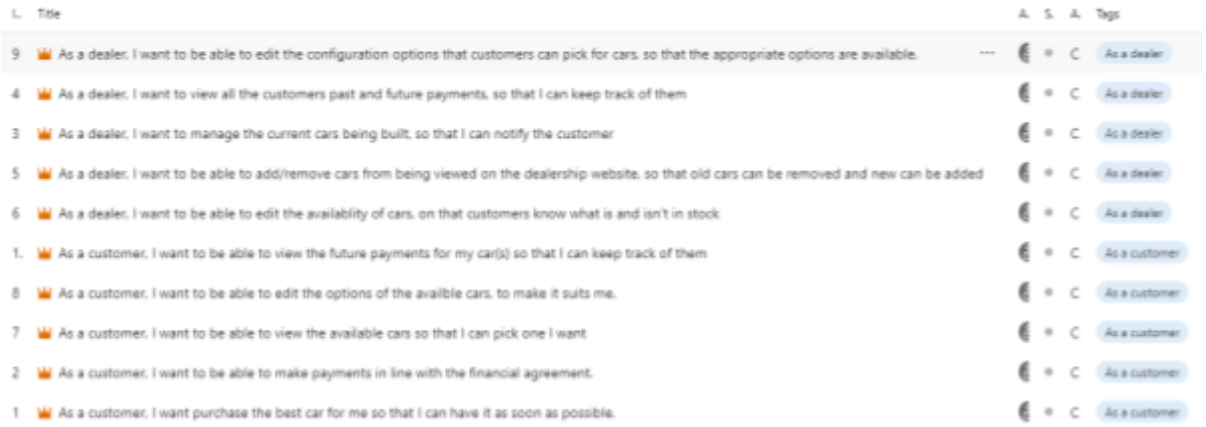


Figure 1.4 - User stories

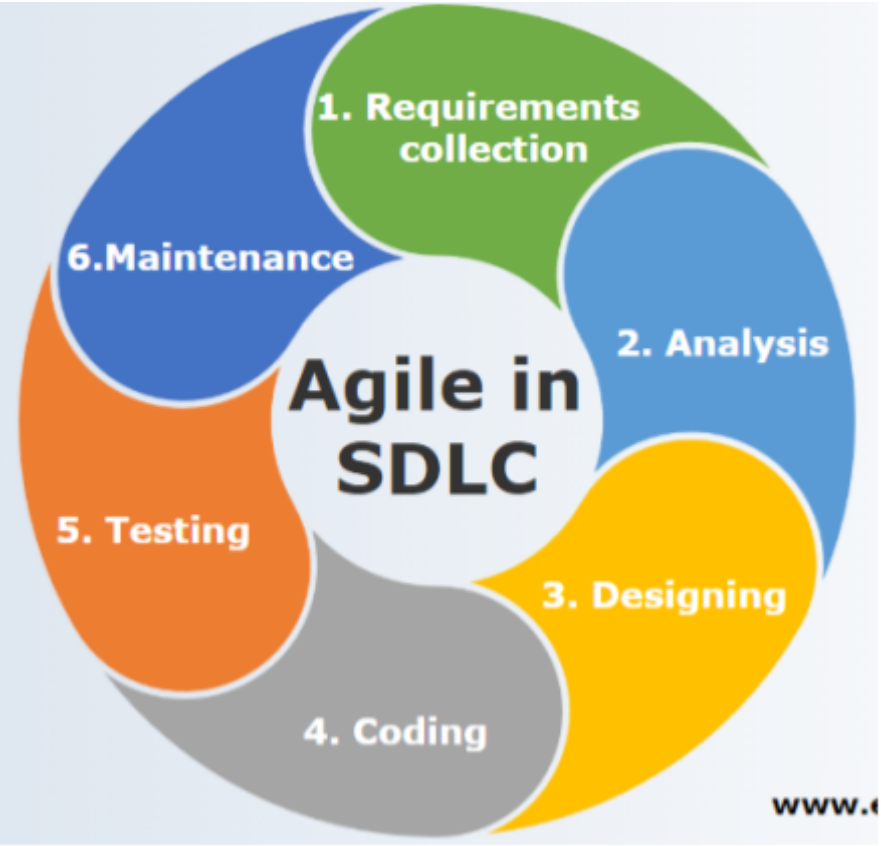


Figure 1.5 - (Chaudhary, 2019)

# Software Architecture and High Level Data Flow

## Use Case Diagram

See below in figure 2.1 a use case diagram. A Use Case Diagram (UCD) describes a set of actions which we call cases that a system should perform in collaboration with external users of the system they are known as actors (Kirill Fakhroutdinov, 2014). Each use case should provide valuable results to the actor or stakeholder.

These diagrams are both behaviour and structure diagrams (Kirill Fakhroutdinov, 2014).

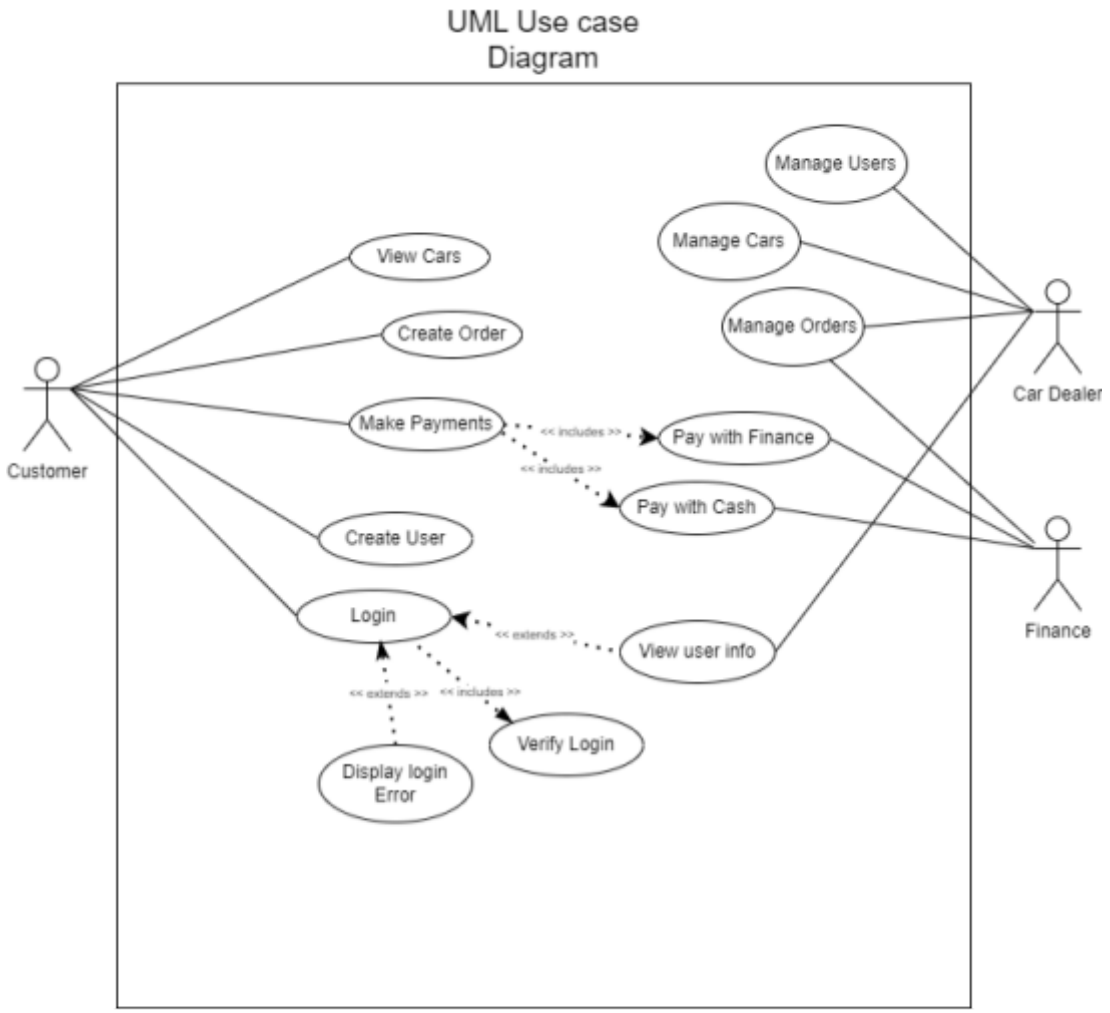


Figure 2.1 - Use case diagram for car dealer service

## Model View Controller Diagram

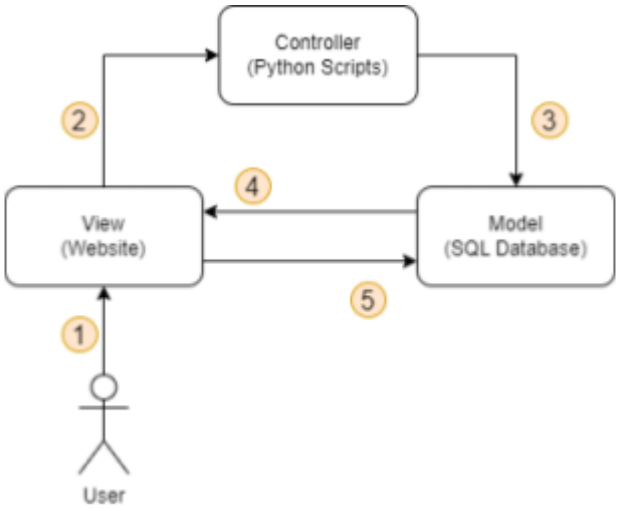


Figure 2.2 - Model View Controller Diagram of dealership service

1. The view is the website. The user interacts with the website to access the data and services the system provides

2. The view notices the controller of a particular event. In this case, the website form or buttons when pressed results in python scripts being run. Depending on the request we then move to 3.

3. The controller updates the model. An example of this is when the user adds a car to the basket, and this results in a row being entered into the users basket table.

4. The model alerts the view that there has been a change. An example of this is producing a confirmation message to the user of the action they have committed.

5. After step 4 the view may grab more data from the model to update itself. An example of this is when a user buys a car the “/mycars” page is updated to show the user's new car.

The diagram and descriptions above have been aided by (Martin, 2019).

## Entity Relation Diagram

Below is the entity relationship diagram, which shows how the tables in the database interact with each other.

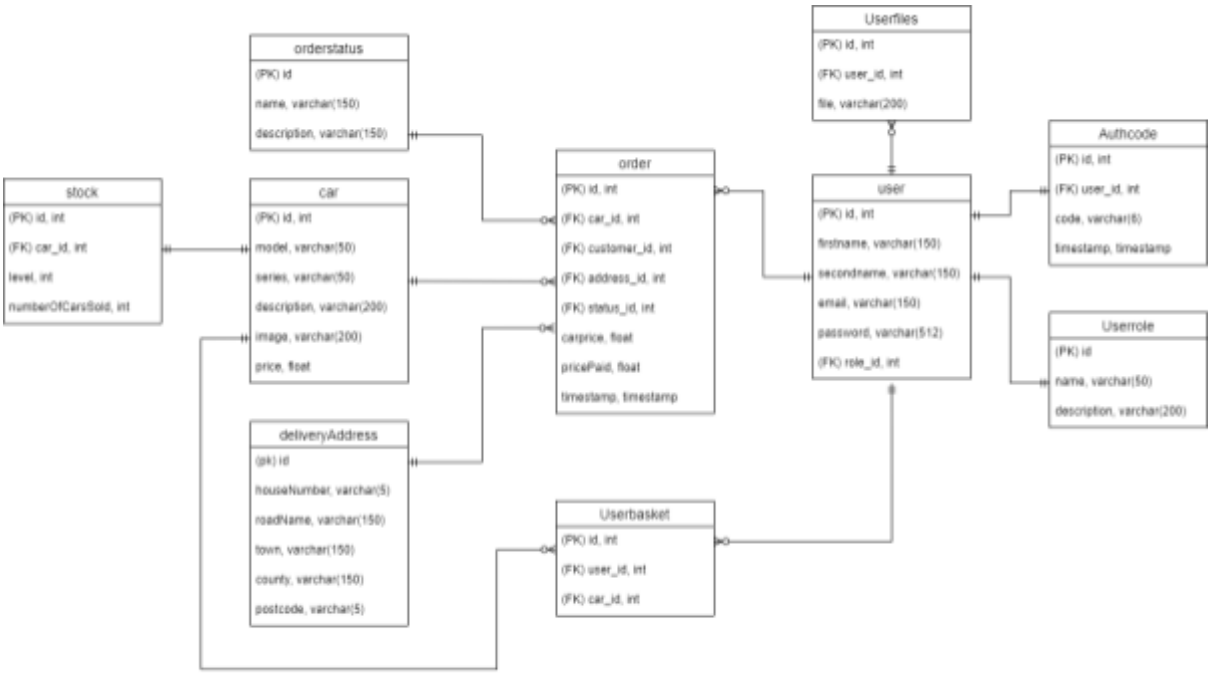


Figure 2.3 - ER Diagram for car dealer SQL database

.

## Class Diagram

This class diagram doesn't directly show how the system is made up. This is because I used flask and SQLalchemy which are python libraries therefore the make of the system is slightly different. This is to demonstrate what it would look like if I had more time and would have used classes more.

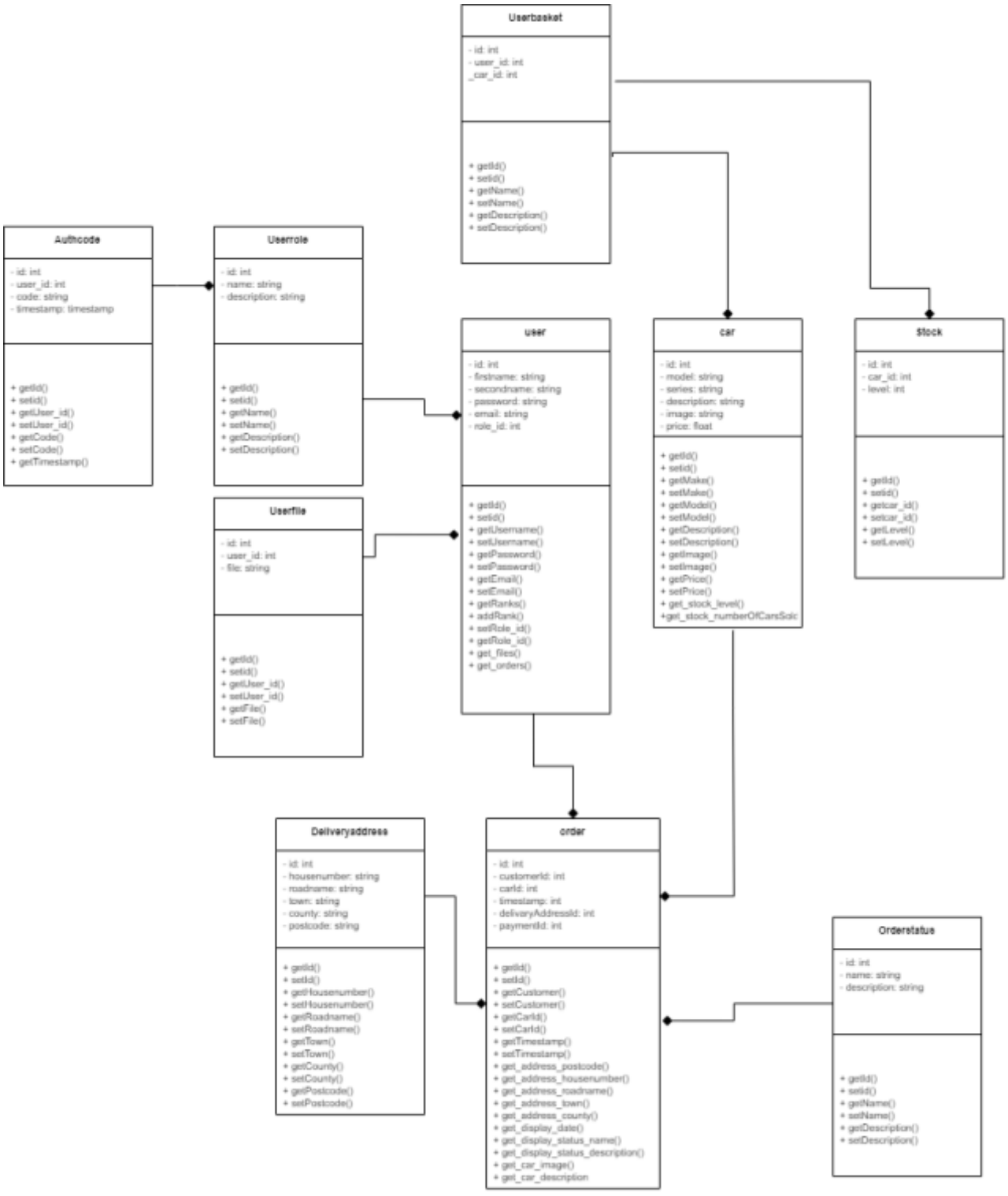


Figure 2.4 - Class Diagram for the system

# Evidence That The Prototype Meets The Business Requirements

## Proving Business Requirement 1

In figure 3.1 below, shows the homepage of the website, customers are able to view the available cars. You have to create an account and login to be able to add a car to the basket.


Figure 3.1 - Home page '/'

## Proving Business Requirement 2 and 5

For a customer to purchase a car must first add it to their basket but to do that they need to login.

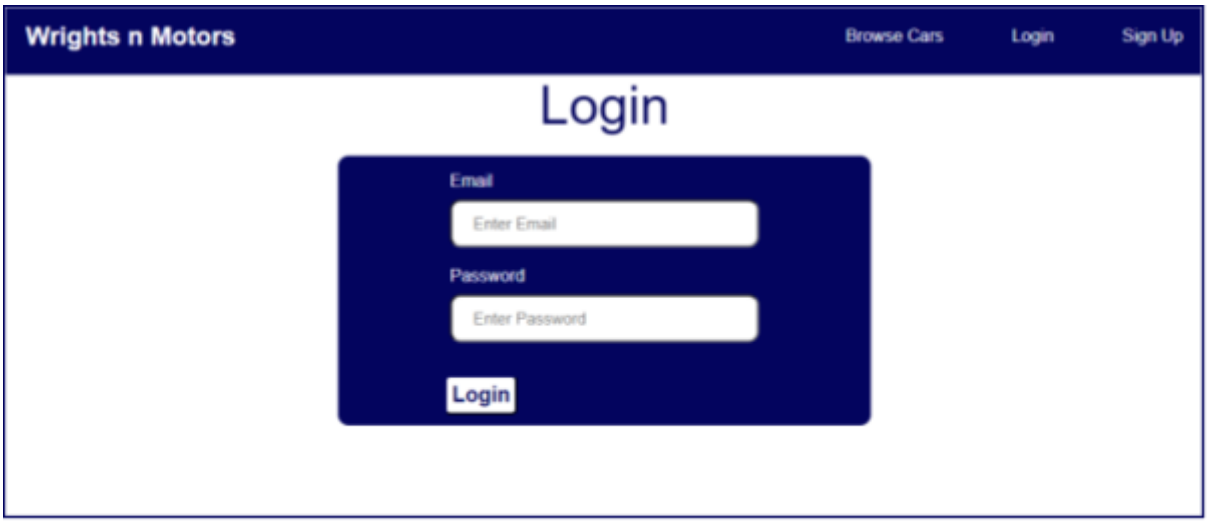


Figure 3.2 - login page "/login"

If a customer doesn't have an account they are required to sign up. See in figure 3.3.


Figure 3.3 - sign up page "/sign-up"

Once a user has signed up, they will automatically be logged in.

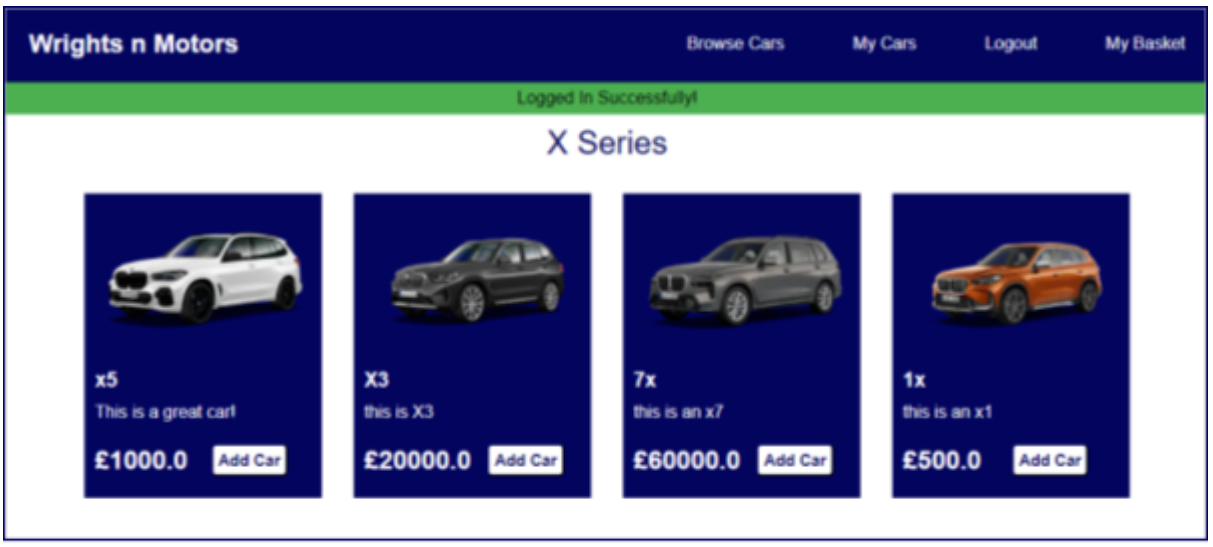


Figure 3.4 - Home page once user logged in

When the user logs in they are displayed with a successful message, they are now able to add cars to their basket.

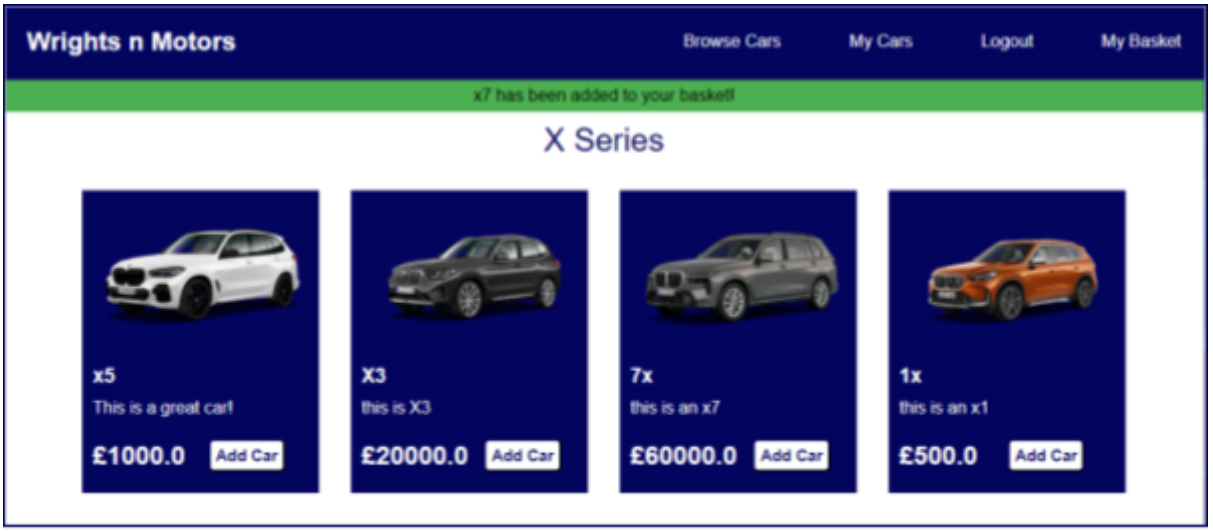


Figure 3.5 - Adding car to basket

When the user adds a car to their basket they are displayed with a confirmation message. The user can then click the “my basket” on the taskbar and see their basket.


Figure 3.6 - Users basket

Once an item is in the users basket, they can remove it or then proceed to checkout.

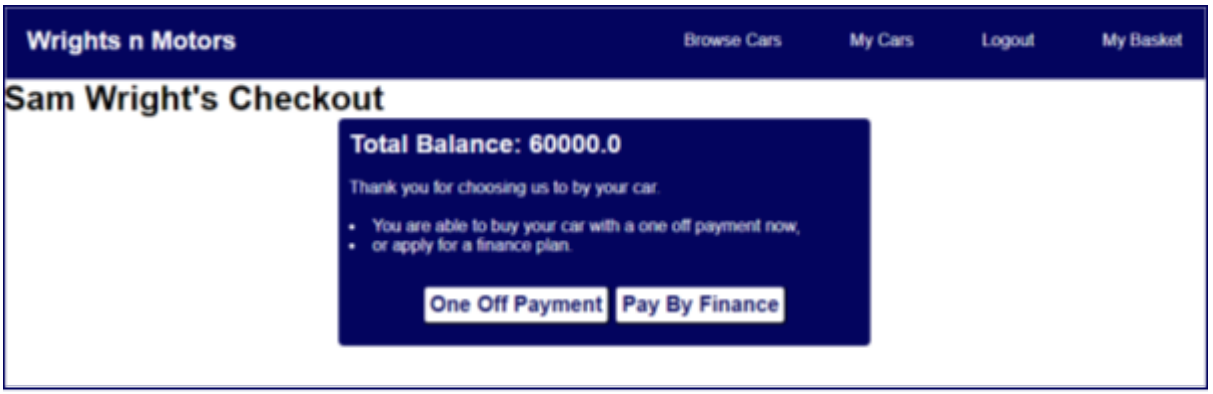


Figure 3.7 - Checking out car selected

A customer is then able to choose between one time payment or apply for finance. We will first do one off payment.


Figure 3.8 - Payment interface

The customer enters their bank details.

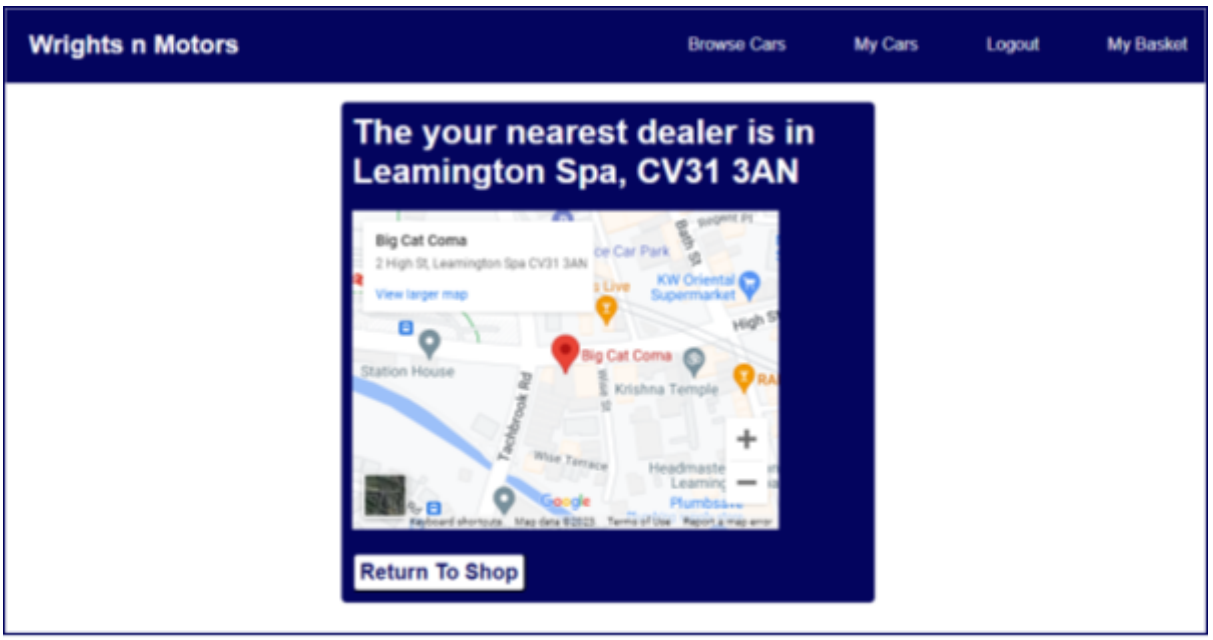


Figure 3.9 - Closest car dealership

The customer is then displayed the closest dealer. We can then go to the “my cars” tab and view our order

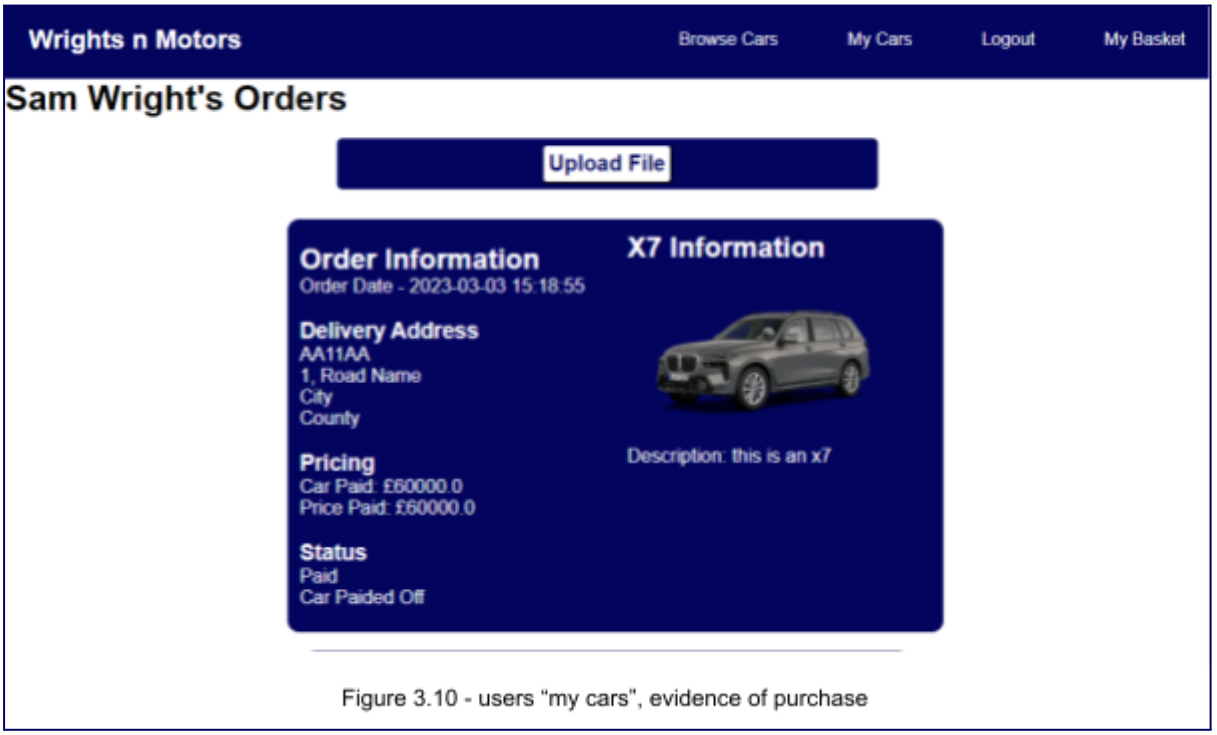


Figure 3.10 - users “my cars”, evidence of purchase

## Proving Business Requirement 3, 4 and 5

If the user wants to apply for finance instead of clicking one off payment in figure 3.7 they can instead click “Pay By Finance”. They then processed to enter bank details as in figure 3.8. Once that is done they are then displayed to enter the personal documents


Figure 3.11 - “/uploadfiles”


Figure 3.12 - Confirmation of upload

The user can then view and remove their uploaded documents on the “my cars” page


Figure 3.13 - Users uploaded documents

The car will then be awaiting approval.


Figure 3.14 - Car Order

Once the finance team has then approved the car the status will change and then the option for payment will be allowed.


Figure 3.15 - Ongoing order

The customer is then able to make a payment.

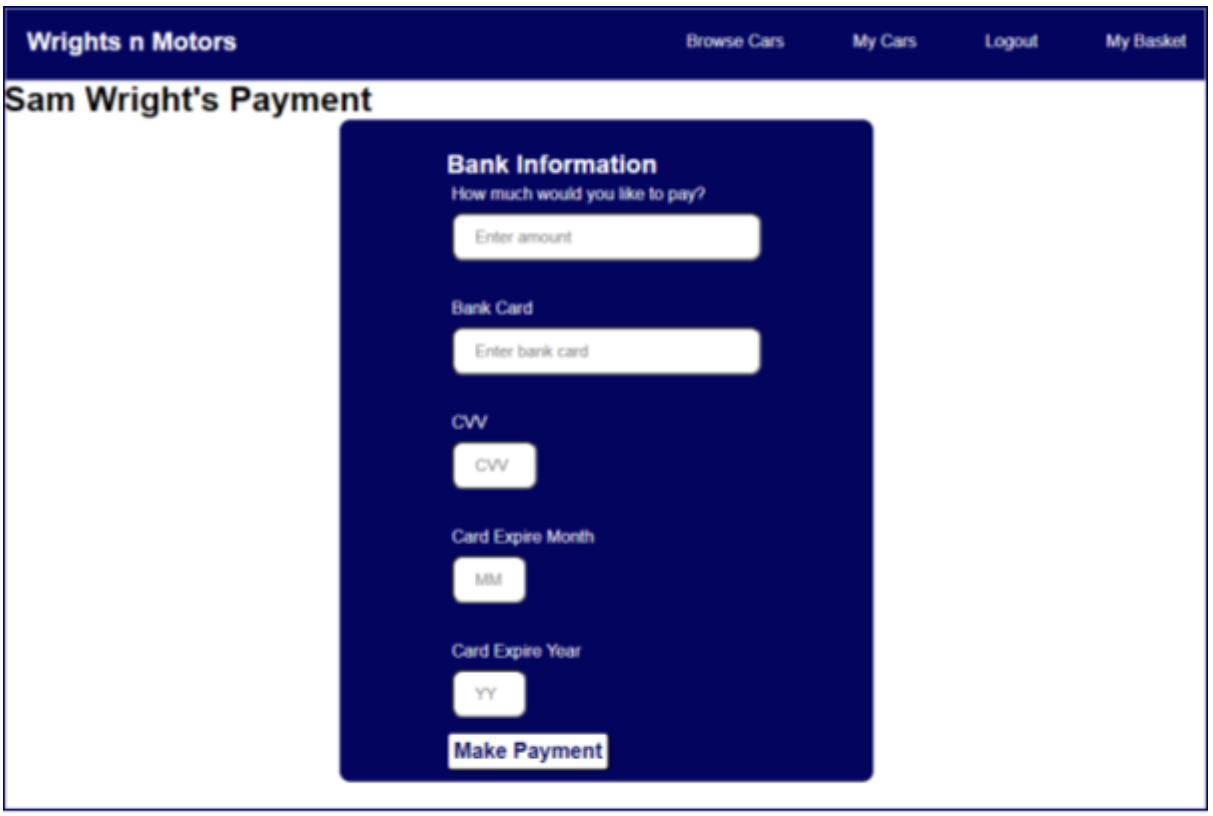


Figure 3.16 - Making a payment for finance

The user is then able to choose how much to pay and enters the details.

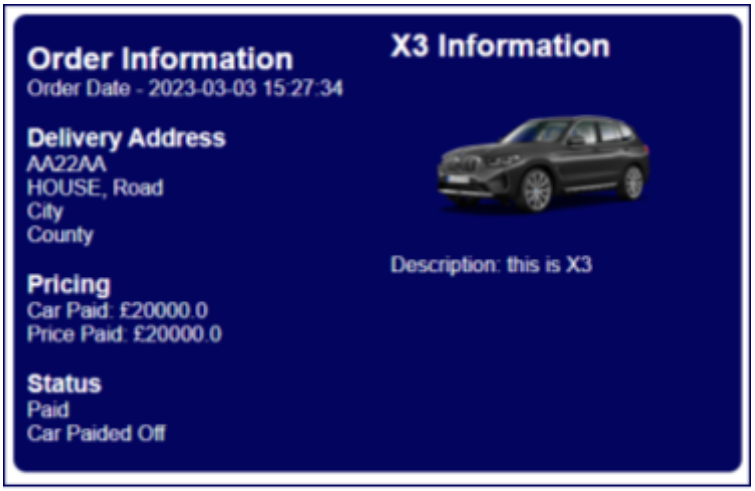


Figure 3.17 - car paid off

Once a customer has paid off the car, they can no longer make payments and the status is updated.

## Proving Business Requirement 6

When a user logs in using an administrator account they are also required to login using an code which is sent to their email.


Figure 3.18 - Authorization code entering interface

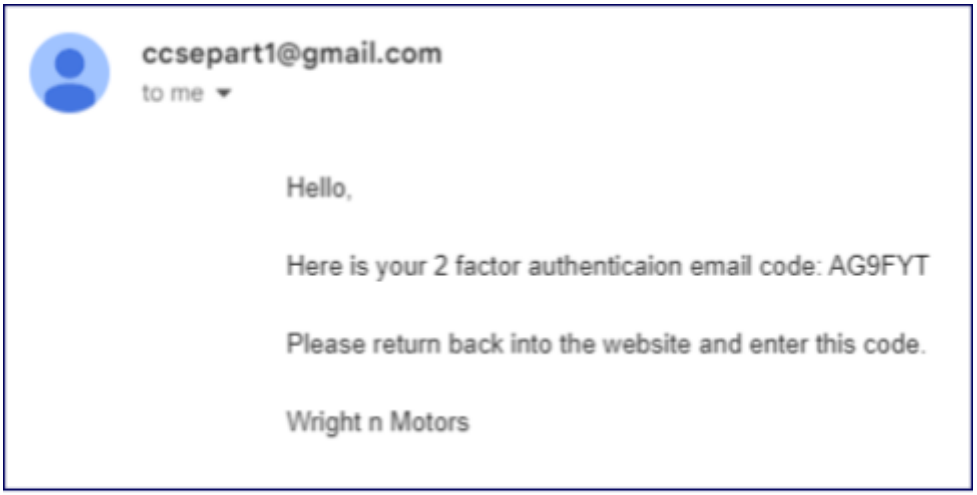


Figure 3.19 - Email for 2FA code

## Proving Business Requirement 7 and 8

Once admin logs in they will be displayed an additional admin button on their taskbar.


Figure 3.20 - Admin logged in

Once you click on the admin portal you will then be displayed with the admin dashboard.

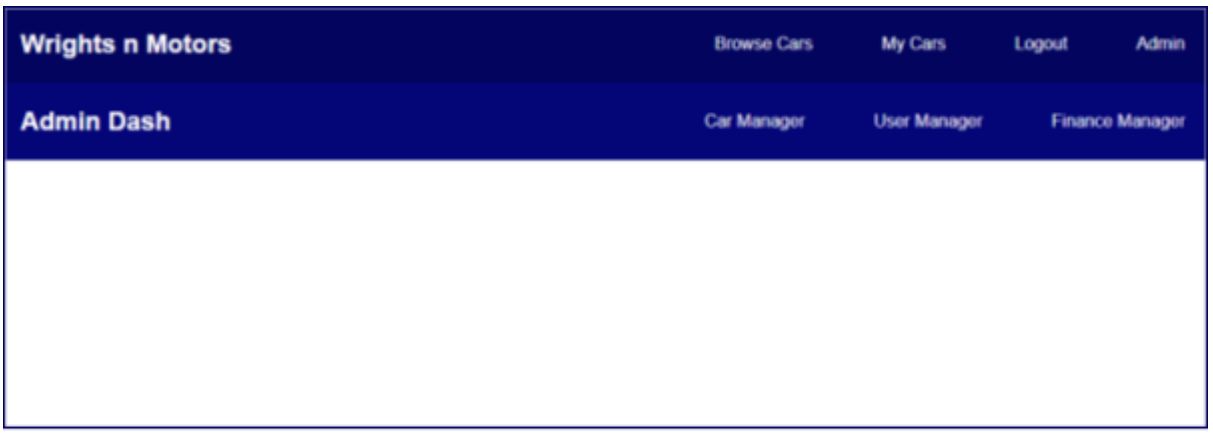


Figure 3.21 - Admin dashboard

The admin then clicked on the car manager, user manager or finance manager. Starting with cars.


Figure 3.22 - Car manager page “/carmanager”

On the User Manager you can see all the user uploaded files as well as all the users orders.

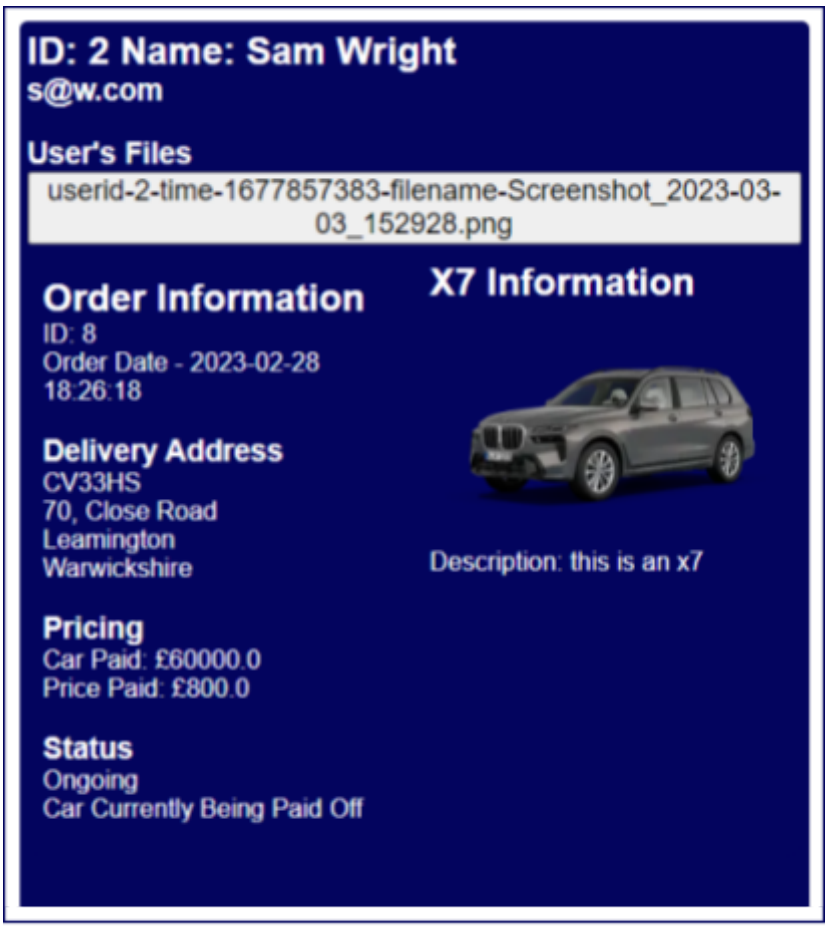


Figure 3.23 - Users information

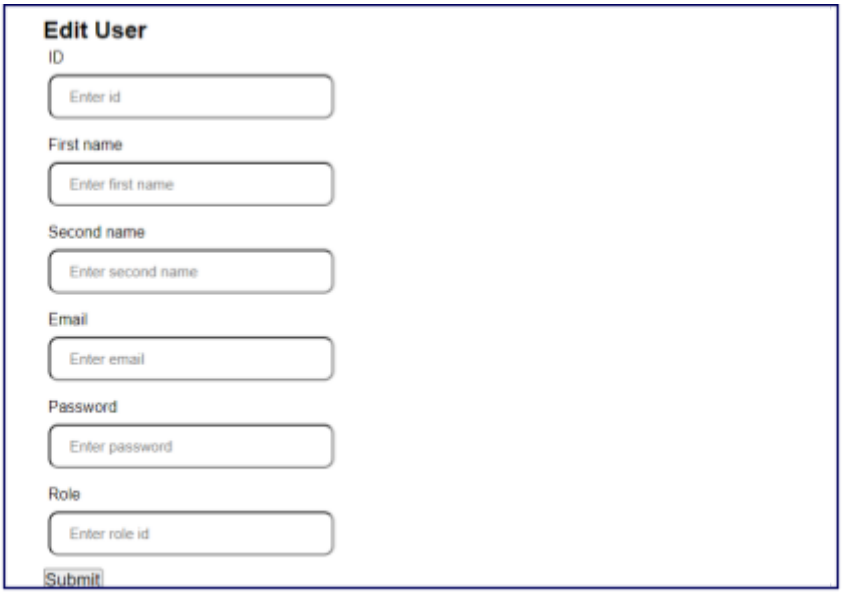


Figure 3.24 - At the bottom of the user manager you can edit the user information


Figure 3.25 - Updating order status finance manager

The finance module allows the finance team to upload the status of an order changing a customer's access.

# References

The videos which helped me to build the projects are:

Main website: https://www.youtube.com/watch?v=dam0GPOAvVI
Navbar: https://www.youtube.com/watch?v=oLgtucwjVII&t=61s
Using files: https://www.youtube.com/watch?v=RYOQOfdN2qg&t=332s
Using files: https://www.youtube.com/watch?v=pPSZpCVRbvQ&t=1s
Using files: https://www.youtube.com/watch?v=sy1MNWt7om4

Chaudhary, L. (2019). *Agile in SDLC | Characteristics and Functionality of Agile in SDLC*. [online] EDUCBA. Available at: https://www.educba.com/agile-in-sdlc/ [Accessed 3 Mar. 2023].

Codecademy (2019). *Software Development Methodology: What is Agile? YouTube*. Available at: https://www.youtube.com/watch?v=GzzkpAOxHXs [Accessed 3 Mar. 2023].

GDPR (2018). *General Data Protection Regulation (GDPR)*. [online] General Data Protection Regulation (GDPR). Available at: https://gdpr-info.eu/ [Accessed 2 Mar. 2023].

Kirill Fakhroutdinov (2014). *Use case diagrams are UML diagrams describing units of useful functionality (use cases) performed by a system in collaboration with external users (actors).* [online] Uml-diagrams.org. Available at: https://www.uml-diagrams.org/use-case-diagrams.html [Accessed 2 Mar. 2023].

Martin, M. (2019). *MVC Tutorial for Beginners: What is, Architecture & Example*. [online] Guru99.com. Available at: https://www.guru99.com/mvc-tutorial.html [Accessed 2 Mar. 2023].

PaulPetrus (2020). *What Is The Difference Between Scrum and Sprint? | Long Version*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=UVx0oj32NVU&t=151s [Accessed 3 Mar. 2023].