**Final Iteration**

**Report**

**Phones, Parts & Repair**

**Computing with Software Development**

**Year 2, Group 2B2**

**Date: 26/04/2021**

**Supervisor**: Keith Nolan

**Students:** X00167137 – Sabina Mihoc

X00163321 – Glen Singleton

X00160105 – Kira Nikitina

Student Compliance with Academic Integrity

All students are expected to complete their courses in compliance with University regulations and standards. No student shall engage in any activity that involves attempting to receive a grade by means other than honest effort, for example:

1. No student shall complete, in part or in total, any examination or assignment for another person.
2. No student shall knowingly allow any examination or assignment to be completed, in part or in total, for himself or herself by another person.
3. No student shall plagiarize or copy the work of another and submit it as his or her own work.
4. No student shall employ aids excluded by the instructor in undertaking course work.
5. No student shall knowingly procure, provide, or accept any materials that contain questions or answers to any examination or assignment to be given at a subsequent time.
6. No student shall procure or accept assignments from any other student from current or prior classes of this course.
7. No student shall provide their assignments, in part or in total, to any other student in current or future classes of this course.
8. No student shall submit substantially the same material in more than one course without prior authorization.
9. No student shall alter graded assignments or examinations and then resubmit them for regrading.
10. All programming code and documentation submitted for evaluation or existing inside the students computer accounts must be the students original work or material specifically authorized by the instructor.
11. Collaborating with other students to develop, complete or correct course work is limited to activities explicitly authorized by the instructor.
12. For all group assignments, each member of the group is responsible for the academic integrity of the entire submission.

**N.B to be filled out by each member of the team.**

By including my name in the form below, I declare that I understand and will abide by the University Regulations and Policies covering Academic Integrity. I accept that each member of the group is responsible for the academic integrity of the entire submission. I will retain a copy of this agreement for future reference.

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name (1):** | **Sabina Mihoc** | **Student No.:** | **X00167137** |
| **Student Name (2):** | **Glen Singleton** | **Student No.:** | **X00163321** |
| **Student Name (3):** | **Kira Nikitina** | **Student No.:** | **X00160105** |
| **Student Name (4):** |  | **Student No.:** |  |
| **Module title:** | **Project** | | |
| **Programme Title:** | **Computing with Software Development** | | |
| **Date: DD/MM/YYYY** | **26/04/2021** |

**Division of work**

**Sabina Mihoc :**

- Booking form,

- Booking display, cancel booking option for user,

- Progress bar for the booking status,

- Email confirmation,

- Staff interface for the bookings with add, update, and delete functionality,

- Staff Search & filters for the bookings,

- Edit profile page,

- Selenium IDE, Selenium Webdriver and unit tests for the booking system.

**Glen Singleton:**

* Password Reset
* Password Change
* Contact Form
* Django ImageKit
* Vouchers

**Kira Nikitina:**

* Shop
* Search
* Cart
* Order
* View / Print Order
* Wishlist
* Stripe Payment
* Thank you Page
* Reviews

**Table of contents**

1. Introduction……………………………………………..
2. Project Plan and Requirements document……………....
3. Project Goals………………………………………………..
4. Feasibility of Project………………………………………..
5. Analyse Requirements - Use Case Diagram…………….
6. Use Case Descriptions……………………………………..
7. Class Diagram……………………………………………….
8. Iteration 1 Report…………………………………………….
9. Iteration 2 Report…………………………………………….
10. Iteration 3 Report…………………………………………….
11. Conclusion…………………………………………………...

**Introduction**

For a duration of over 12 week we, as a team of three second year Computing students, worked together to design, develop and test an e-commerce shop for electronics with a booking system for repairs. We hoped to gain more in depth knowledge and experience in combining everything we have learned so far in a big project. We chose this topic because we believed it would be a very useful experience for our future career.

However, apart from our previous knowledge we did a lot of research to learn and implement new functionalities. We used different tools and technologies such as Django, Python, HTML, CSS, JavaScript, SQLite, Git, Selenium IDE and Azure Devops to help us reach our goal and overcome the challenges we faced.

Another challenge we had to overcome was time management, but having criterias to meet for each iteration and deadlines helped us to reach most of our goals.

By taking part in this project we learned the importance of working as a team, sharing ideas, helping each other and the importance of communication within the team members.

**Project Plan and Requirements document**

**1.Project Goals**

Our goal is to develop a fully functional eCommerce website. We are planning to create a website that allows users to buy mobile phones or make bookings to repair their devices. We set our goals for the website considering what would be useful and enjoyable for our key users. Our key users are the people who are interested in purchasing new phones, parts or fixing faulty devices.

To ensure a positive user experience we are aiming to make our website easy to use, avoiding overwhelming colour schemes and unnecessarily animations that could potentially slow down the website. The navigation bar should allow the users to easily find what they are looking for and provide more details only if users are interested. To increase the number of users we aim to provide a responsive mobile-friendly website that allows them to shop and make a booking using their phones.

Our online phones and parts shop can be viewed and accessed by all the users. They can use the search bar to look up the products available. To add products to the cart a user must create an account. To register, they will have to enter their name, email address and create a password. When making a purchase the registered user will have to fill in a different form for the delivery address and payment card information. Registered users will have access to more features. They will be able to edit their profile, add products to a wish list, apply vouchers, view order history, rate a product and write reviews. Also, they have the right to request a deletion of the account which will be approved by an admin.

The online booking system is designed to help control the number of faulty devices coming in each day. A registered user can make an appointment in a few easy steps. They will have to select a repair option from a list, the booking type which can be in store drop off or post the device and a date for the booking. Lastly an estimated cost for the repair will be displayed. Based on that the user can proceed or cancel the booking request. For more booking options or any other queries, the users can contact the shop using the contact form available on the website.

The admin of the shop will be viewing and controlling the products stock, reviews and bookings. The admin will also be responsible for handling the registered user accounts and dealing with any requests.

**2. Feasibility of Project**

**Prioritise Requirements**

As a team we went through the process of prioritising the requirements for our website. The purpose of that step was to identify the base we wanted our website to have. Within the given timeframe of 12 weeks, we believe we would be able to deliver a website with functionality similar to that of an established retailer such as:

* three types of users such as non-registered user, registered user and staff/admin
* payment functionality
* shopping cart
* search bar
* contact form
* repair booking system

**Iteration Goals**

To achieve our goal, we will be delivering our website through several iterations. For the first iteration (due 5th March), we believe that a basic eCommerce website would be sufficient to show the direction we wish to take with our project. We will develop the website using Django Framework, Python, HTML, Bootstrap, JavaScript and Css to style the website. We will collaborate and organise our tasks using Azure Devops.

For the second iteration (due 26th March) we plan to flesh out the website with more features and functionalities. To reach our target for the website we decided to include star-rating, reviews, social share, print receipt, wish list and voucher codes as additional functionalities.

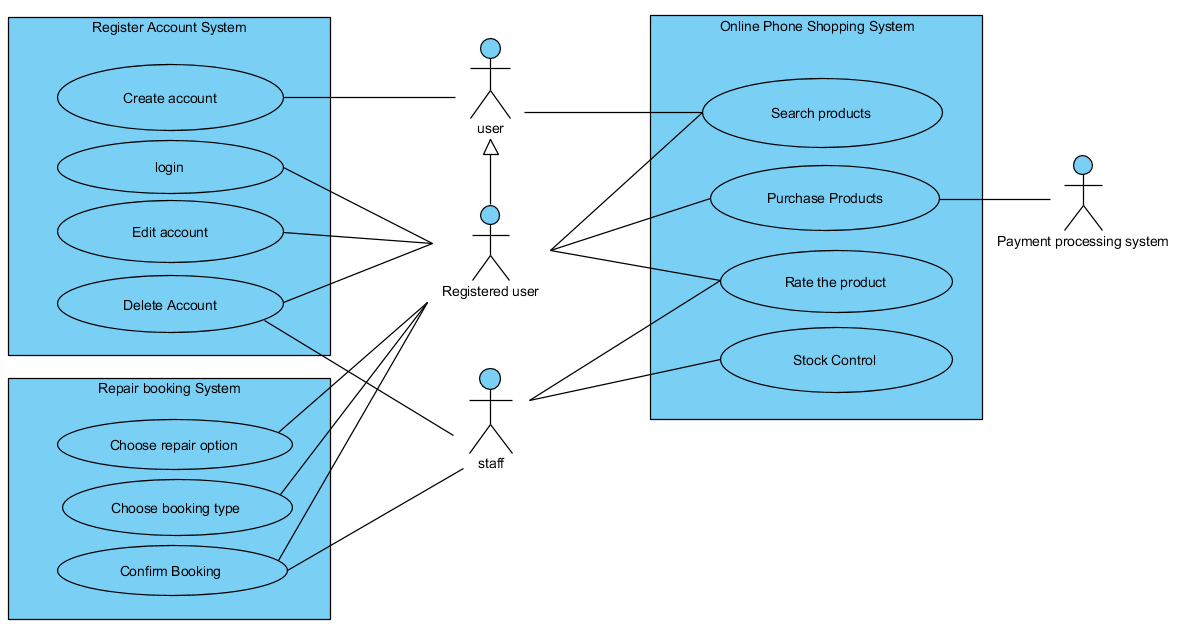
For our final iteration (due 26th Apr) we will do final tests to make sure all the functionalities work as expected. Also if time allows we will look at ways to improve the layout and appearance of the website. Lastly, we will deploy our website from our local server to the internet using PythonAnyWhere.

This feels very feasible as there are functions available within Django that would be able to supply the main parts of what we want for our website (eCommerce with database functionality) and any additional features that we haven’t got experience with within Django yet, we can research and implement into our project.

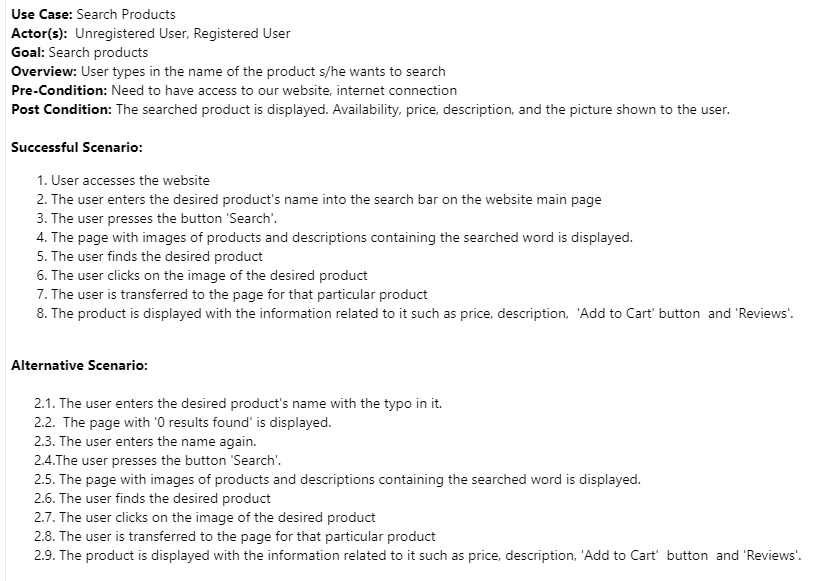
While functionality may not be a problem with the feasibility of the project, there are still risks, for example, time taken researching and implementing new functions as well as the risks caused by a lack of communication, which we will try to minimise as much as possible by sharing our challenges with all the team members, listening to each other and helping each other through the whole process of delivering a website.

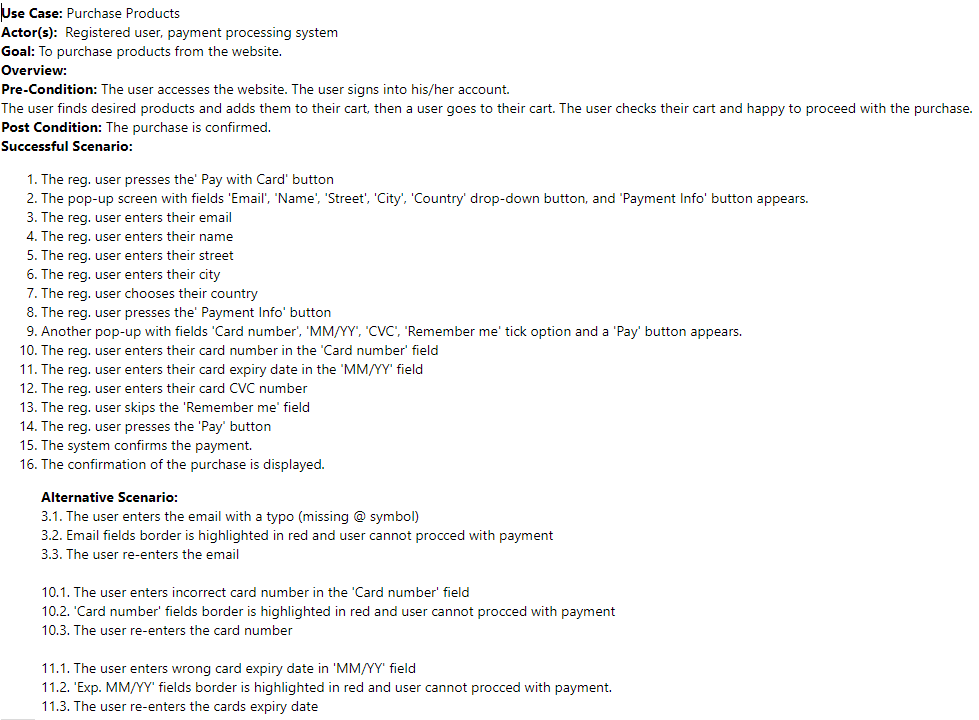
**3. Analyse Requirements**

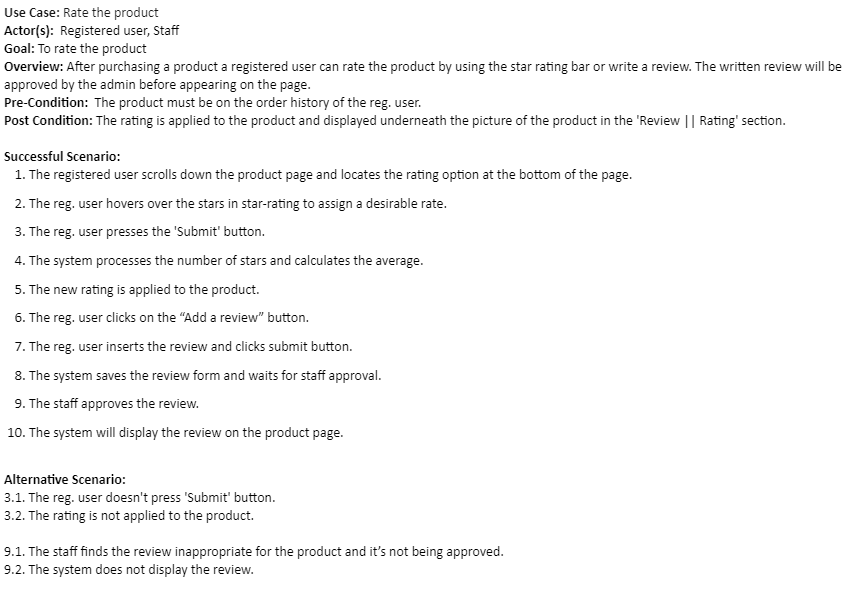
**Use Case Diagram**

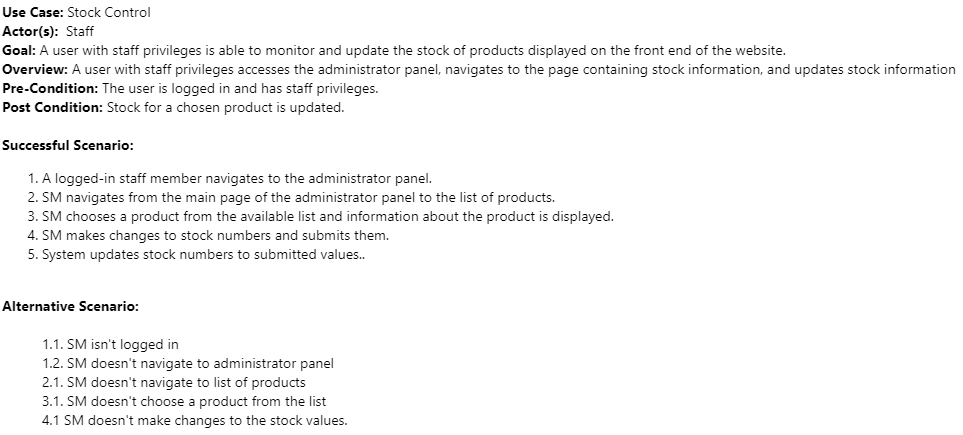


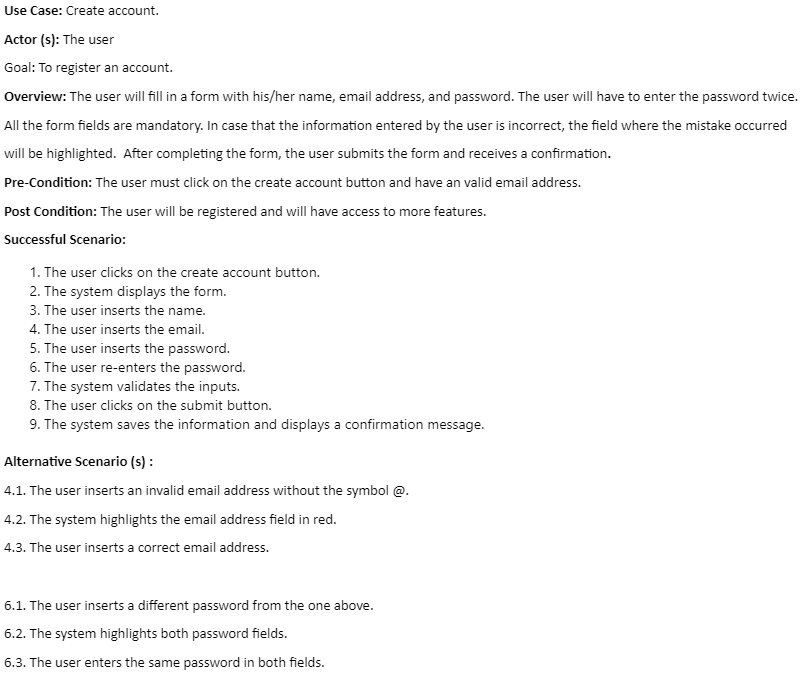
**Use Case Descriptions**

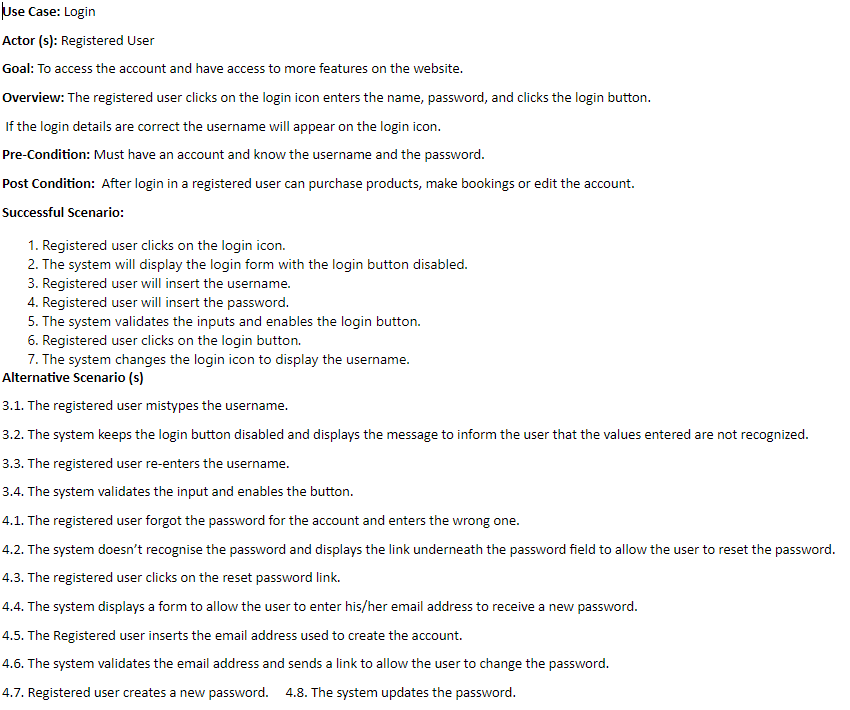


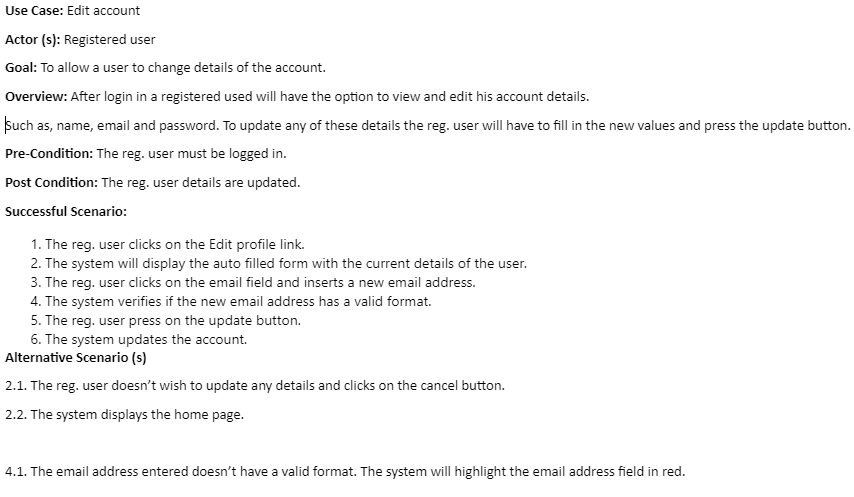


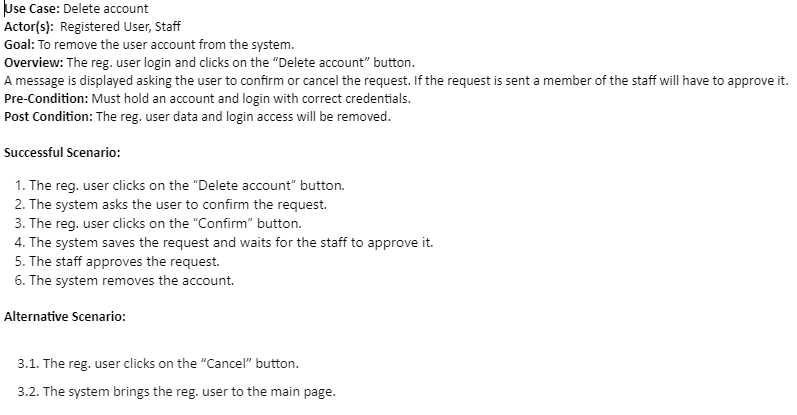


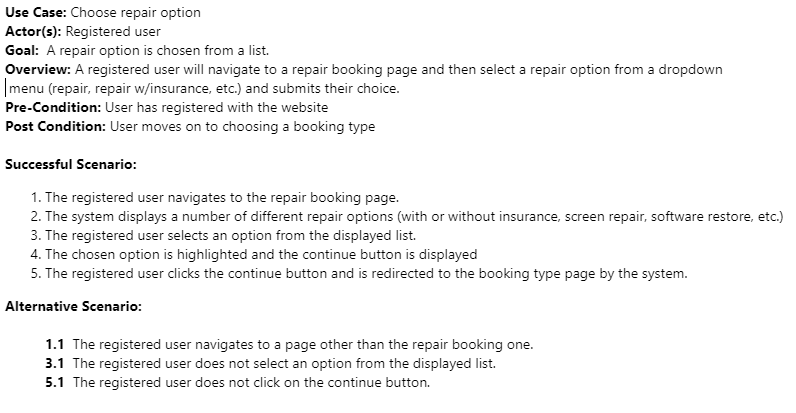


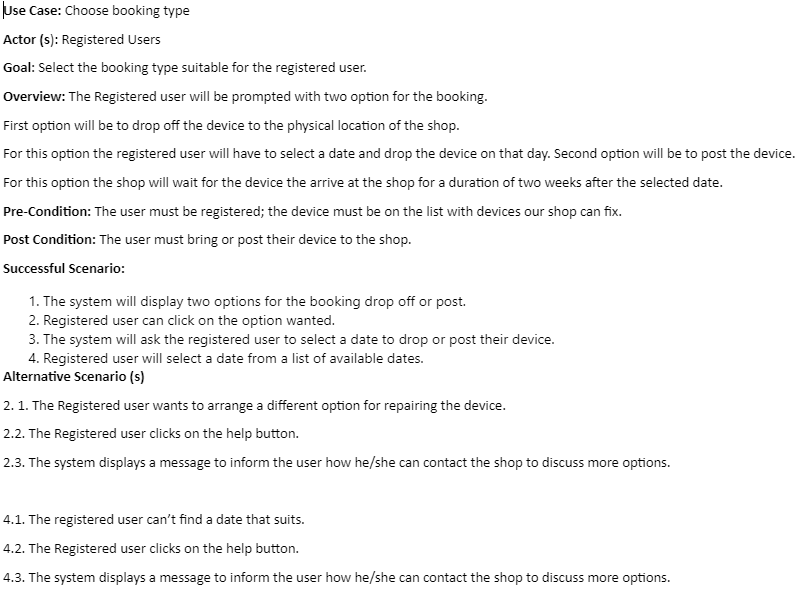


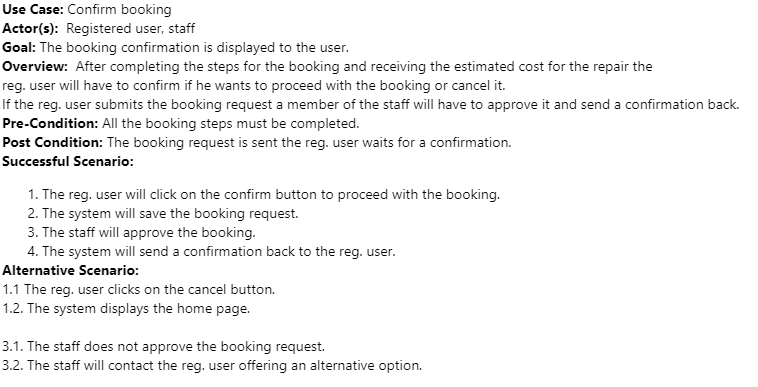




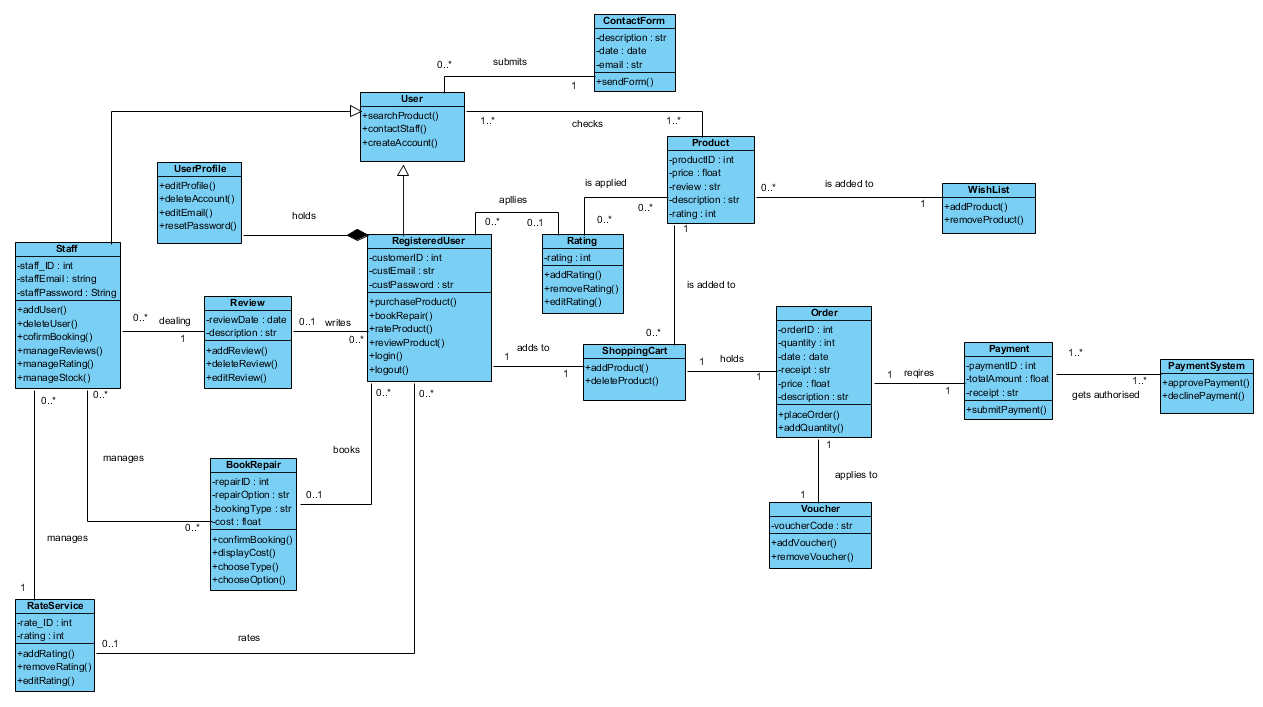








**4.Class Diagram**



**Iteration 1**

**Report**

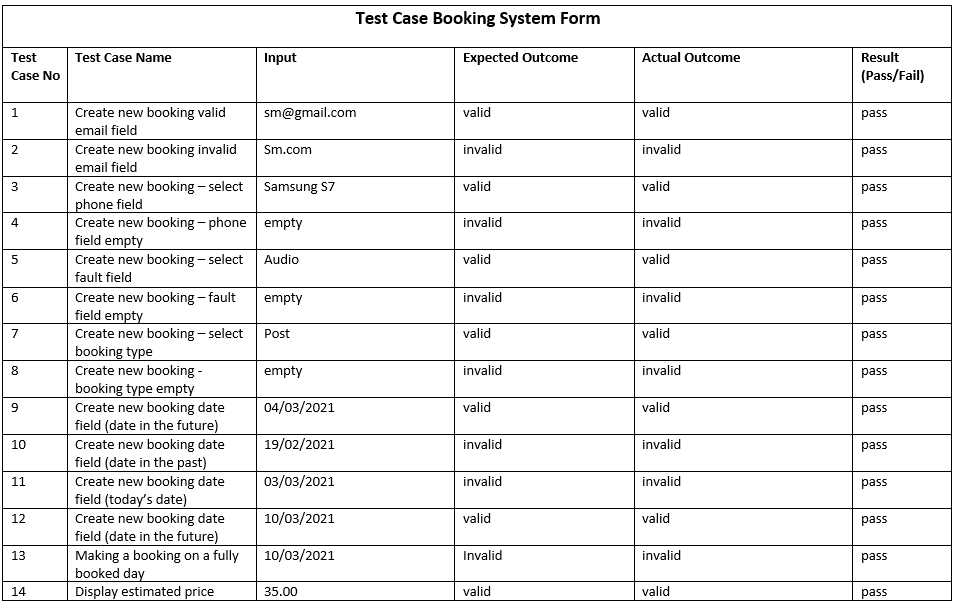
**Booking System**

Considering the user requirements for the booking system we developed a prototype with CRUD functionalities. This allows the registered user to fill in a booking form, submit it, view the booking details, or cancel it. To make a booking they will have to insert their email, select their phone model, the fault to be repaired, booking type and the date. The first step was to create the models needed for the database and the booking form. The administrator has control on what information is available to select in the booking form. He can add/delete the phones or faults from the list, view and approve all bookings.

During the development process for the booking system we encountered several bugs including typos, missing imports, logical errors in the functions views/urls and logical errors made when creating the database such as placing the foreign key to the wrong model.

The focus for the first iteration was to check that the booking form is working as expected. We conducted our tests using Selenium IDE. We tested different scenarios such as the value selected remains in the field, can the user submit an incomplete form, is the data saved to the database, is the username saved automatically, is the email address valid, is the date valid and available. During our first tests we discovered that we needed to adjust our date field not to allow the user to select a date from the past or the current date. Also, we had to implement a counter for the bookings to only take 5 per day.  After this we thought about ways to improve the user experience while making a booking by adding a “other” choice if the user can’t find or don’t know what’s the phone fault.

At the end of the first iteration we can conclude that the booking form is working correctly, displaying the appropriate messages for invalid choices.

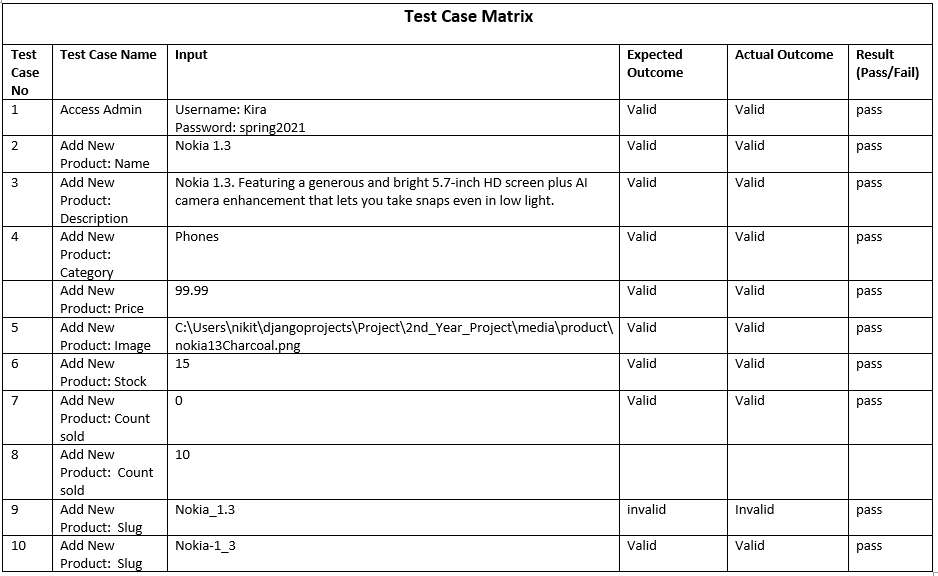


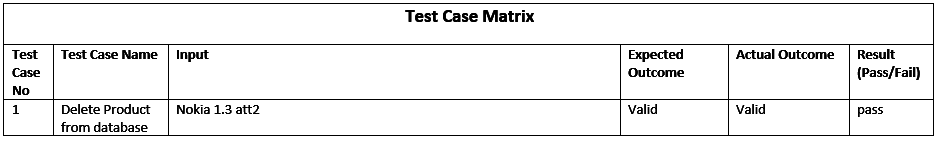
**Shop System**

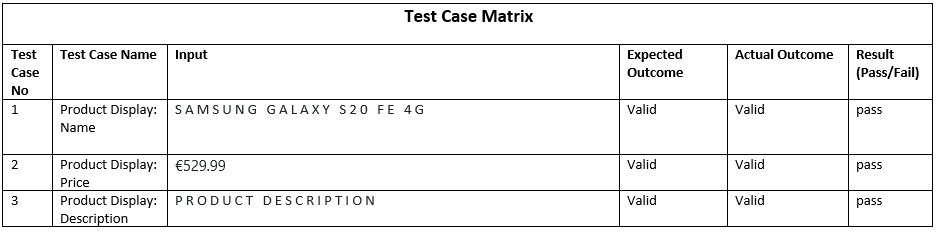
Another goal for our first iteration was to have a ’shop’ application developed for our website.

Our shop will consist of products organised into different categories. Both products and categories will have attributes related to it. To achieve this we worked on models, static, media and admin files created for this application and also on the main project settings file.

During development of a ‘shop’ application for our website we have encountered some errors. The ‘shop’ application code contained some ‘TemplateSyntaxError’ errors such as missed endif tag, extra curly brackets, several wrong ‘Euro’ signs  placement. We also encountered a few ‘NoReverseMatch’ errors which meant that we had typos in our URLs. We also encountered ‘ValueError’ after adding some data to our database, for some reason images didn’t apply to a number of products, which was noticed after we tried to run the server. To fix this bug we had to access our database through the ‘SQLite’ and check the data, after it was obvious images were missing, we accessed our database through the admin page and manually added the missing data. Although this bug was fixed after accessing the website we found that the product images weren’t displayed properly, and we yet need to fix it.







**Cart System**

After we have finished the ‘shop’ application we started to work on the ‘cart’ application. However, the ‘cart’ application is still in the development process and will be brought into our second iteration.

**Accounts & Authentication System**

A vital part of the entire website is the ability to create accounts and log into them, most parts of the website use accounts in order to relate a process to a user. Due to this, we decided to put the accounts and authentication system in the priorities for iteration 1.

In order to register an account, the user must input a username, an email and a unique password, the password is then confirmed in a second field and the user is able to complete registration.

Following registration, users are able to change their password, and in the case of them forgetting their password, they have the option to reset it.

Administrators have the option of editing account information through the admin panel as well as deleting user accounts currently.

Our focus for the first iteration was to have the accounts & authentication system functioning regularly at a basic level (creation, password change/reset) as well as allowing users to update their own account information, however this function was not completed in time for the first iteration.

Multiple errors occurred during the development of the accounts & authentication system, chief among them being that accounts could be created but returned an error afterward. This was caused by accounts being assigned to a non-existent group and was solved by creating the missing group (customers) in the administrator panel.

**Iteration 2**

**Report**

**Booking System**

In the booking system for the second iteration, we had no outstanding errors in the booking form and we moved forward in developing an interface for the staff. This is available only for admin accounts and regular users will not be able to access it. The interface allows the staff to easily manage the bookings and the choices the user has in the form when creating a booking. The system has three main categories “New Bookings”, “In Progress” and “Completed”. The booking will move from one category to another when the staff changes the status of the booking. When a booking is approved it will move to the “In progress” section and will stay there until the device arrives at the shop and is repaired. After that it will move to the last category and the user will be notified by email when the device is ready to be collected. Lastly if the device is collected from the shop the staff can mark the booking as completed and the booking will be removed from the display system, but it would be kept in the database. The user would be available to see the progress of the booking when is logged in. We also developed a search filter for the staff to easily find bookings by their status, type, model and so on. They can display all the booked dates and the number of bookings on that day.

We conducted several tests to verify that all the above works as expected. We encounter errors and fails in displaying the booking into the right category and errors when creating the filters. We manage to get them fixed and we will move the booking system into the next iteration with all the tests passed. More tests will be conducted in the final iteration.

**Contact Form**

At the start of iteration 2, we identified that customers did not have a convenient avenue through which to contact the staff on the website.

To this end, we placed a contact form among the priorities for iteration 2.

The way that the contact form works is simplistic, the user inputs their email, the subject of their email and the body of the email into the specified fields. When the user submits the form, its contents are then sent to an email associated with the website through Google’s SMTP service.

Unfortunately, due to an unforeseen issue, the sender of the email is also the recipient (the email account set up as the receiver).

However, we managed to find a workaround in the form of including the user email inputted into the email field in the email (Though we identified that in an actual industrial environment this would be a massive security issue, as such, we will attempt to solve this issue early in the final iteration).

**Cart System**

For our second iteration we continued on delivering the ‘cart’ application. After the code for the application was finished we carried our routine checks and tests, we encountered an error which was the wrong placement of if else tags in our cart.html file which caused us a ‘TemplateSyntaxError’. We have also had few typos in our views for the cart application, which caused us  ‘module has no attribute’ and ‘object has no attribute’ errors. However, those were fixed, and the application started to work accordingly.

**Order System**

Another building block for our online shop, ‘order’ application was  developed during this iteration.

Alongside ‘order’ application a payment function has been developed.

We used ‘stripe’ for our payment option as it is contained in the pip package manager for Python and all we had to do was to install it from the library. We registered our business at stripe website where we gained access to API keys which we registered in project settings.

While testing a payment we encountered an error ‘invalid API key, API key cannot be used more than once’, to resolve this error we had to access our stripe account and revoke the API key, after that we obtained the new API keys which were re-registered in  our project settings.

As a part of the ‘order’ application we developed ‘Thank you’ page, ‘Order History’ and ‘Print Receipt’.

**Wish List System**

We have also decided to develop a ‘wishlist’ application for our website. While doing so we have encountered an error ‘NoReverseMatch’ with Exception Value ‘wishlist is not a registered namespace’. During the investigation of this error, we have found the typo in a path for our ‘wishlist’ application which we registered in our projects URLs. Therefore, after we have resolved the bug, the application worked as expected.

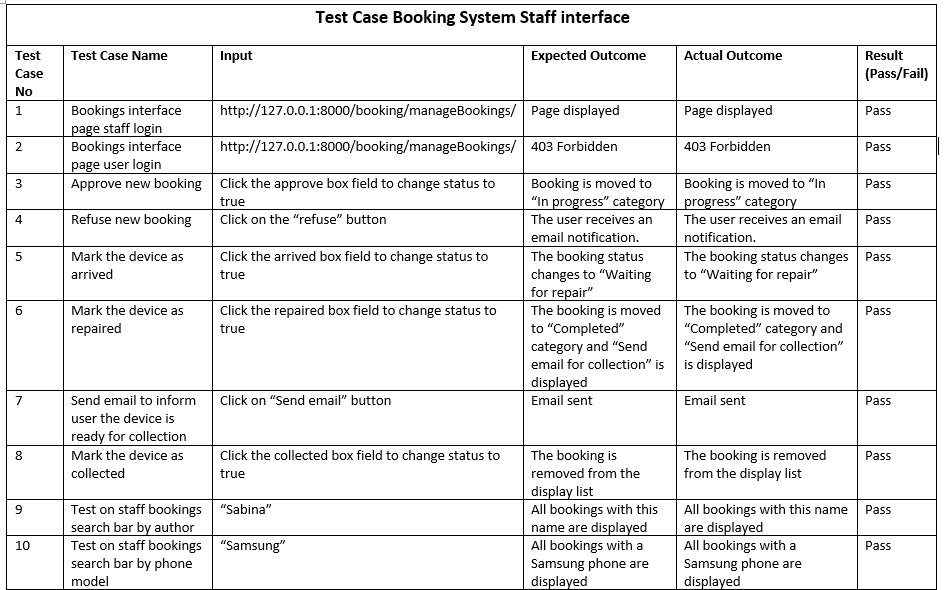
**Styling Improvements**

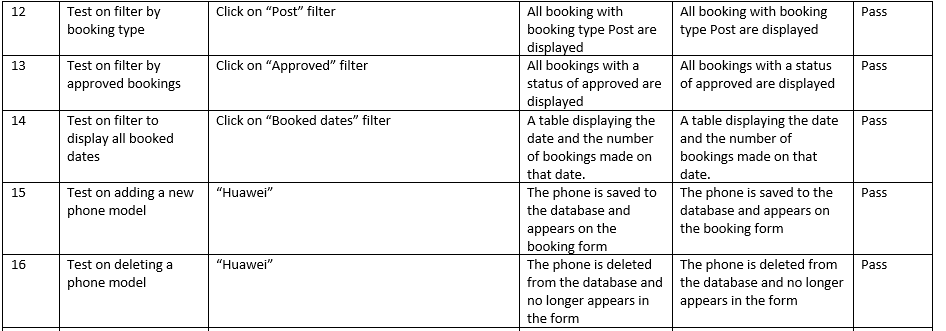
While not a priority during iteration 2, several styling improvements were made to the website.

This can be seen by the website dynamically scaling to the size of the device it is being viewed on.

The footer of the website has also been changed to a fixed element, rather than being at the bottom of each page, it is now fixed to the bottom of the device viewport.

**Test Cases**





**Iteration 3**

**Report**

In our last iteration, we have worked on a few aspects for our website. One of them was styling our website. We wanted it to be as user friendly as possible and have a uniformity to it. It was important for us that for an ease of access our top navigation bar and our footer were fixed, so if the potential client is scrolling through the page - search bar, shop menu, booking a repair and contact us form for non-registered customers and also main customer menu, wishlist and basket for our registered customers are always visible to them.

We have also discovered some features that needed correction such as a wishlist and basket being displayed for everyone even though our goal was that only registered customers can purchase or add items to their wishlist, so we had to add a few ‘if else’ statements inside our code to gain control over that.

**Reviews**

We have added reviews to our products. The idea about the reviews is that a registered user can write a review to a product, but a non-registered user can only read the reviews. The only error we encountered while developing this functionality was ‘no table: shop\_review’ which occurred because we forgot to do the migrations at first, but that was easily fixed.

**Django ImageKit**

Throughout the entire project, the images used for the products were not of a consistent size.

To remedy this we implemented the use of an application called Django ImageKit.

Django ImageKit caches a thumbnail image of specified dimensions in the website for each item in the model it is used in.

**Vouchers**

A usual feature of eCommerce websites we implemented in this iteration was vouchers.

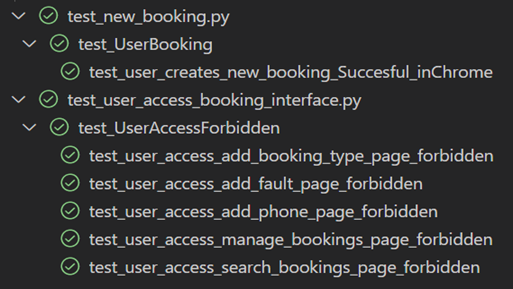
Vouchers allows users to input a code at checkout to reduce the price of products in their cart.

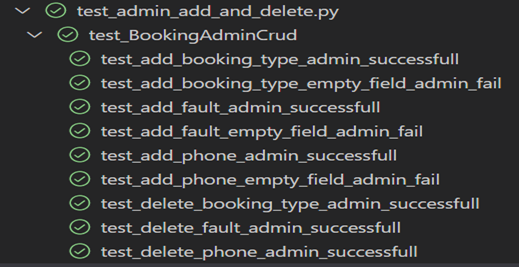
An issue we encountered while testing was the fact that the system would throw an error if the voucher was set to 100%, though this was found to be an issue with the Stripe payment system rather than the vouchers themselves.

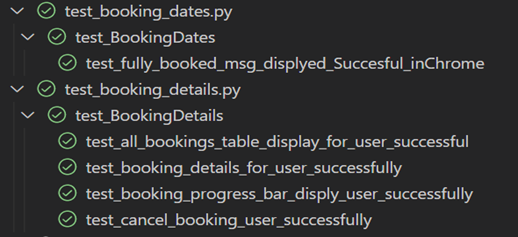
**Booking System:**

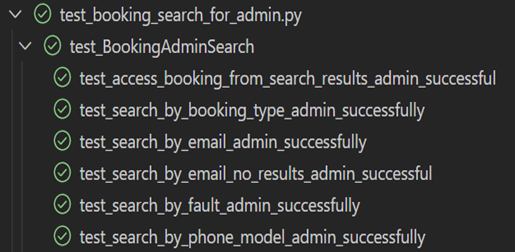
In the final iteration for the booking system, I worked on adding extra styling that will make the system look better. Then I start the testing process. First, I wrote the unit tests using Django Testcases to test if the classes and the methods for the booking objects work as expected. I created a few successful scenarios and a few incorrect scenarios to make sure the code can handle errors. Such as an attempt to make a booking with an empty mandatory field, where I checked if the form would submit invalid data. I also test main functionality such as making a booking, adding a new phone, deleting a phone, user access restrictions and so on. I conducted my acceptance tests using automated tests using Selenium Web driver and some manual tests using Selenium IDE. During these tests I encounter small errors such as the wrong title displayed on the page.





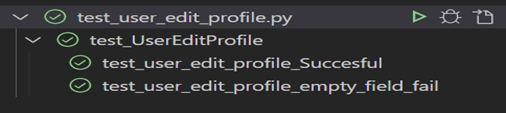


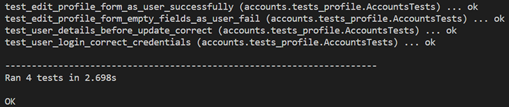




**Edit Profile:**

I created an edit profile page to allow the user to easily change his username, email, first and last name. In this page the user can also mark their account to be deleted. The staff will be able to see the users who requested their account to be deleted and check if the user has active bookings or active orders and if they do not then the account would be deleted. As we implemented this functionality in the last iteration, we did not have time to create a system to manage the users and delete their profile on request. I created unit tests and webdriver tests to check if the user can update his information and if invalid data will be submitted.





**Conclusion**

As we reached the deadline for the project, we can happily conclude that we delivered a fully functional e-commerce website that meets the main requirements specified in this document.

We included important functionalities that each e-commerce website must have and more such as a wish list, reviews and a customer profile page.

Through a lot of research we were able to deliver a completely previously unknown to us functionality of a booking system. We also created a staff interface to allow them to easily manage the bookings from new booking to complete state. We implemented security features to make sure that only the staff can access this interface and it's not visible to the customers.

As a team we faced a lot of challenges with our repositories and the code itself but we overcame them together by helping each other.

Development of the website is an ongoing process and there is always room for improvement and creation.

For future projects we will try to allocate more time for in depth testing to prevent as many bugs as we can. Also we will look at interface improvements, adding more categories and products, Latest Catalogues, Compare options, Cookies, Customer Feedback and more advanced features that we can include such as two factor-authentications. Another important aspect for our future projects would be deploying the website.