Project Report

Car Space Renting System

COMP9900-9900-H11A-Team Winning5

Car Space Renting System

UNSW SYDNEY

Name	Role	UNSW Email	Student ID
Xinyang Zheng	Scrum Master Backend Developer	z5333874@ad.unsw.edu.au	z5333874
Tonghao Liu	Frontend Developer	z5278137@ad.unsw.edu.au	z5278137
Mincong Zhou	Frontend Developer	z5358282@ad.unsw.edu.au	z5358282
Yetai Pu	Backend Developer	z5380563@ad.unsw.edu.au	z5380563
Yifan Wang	Backend Developer	z5340585@ad.unsw.edu.au	z5340585

submission date:04/08/2023

Table of Contents

1. overview

1.1 background

This project creates a website that provides users with the service of renting a private parking space or providing a parking space for rent. In real life, there are a lot of drivers who often can't find a suitable parking space, many of them may need to go for a long time to find a parking space to use because of the need to commute to and from work, and many of them can't find a suitable parking space because of going to an unfamiliar place, and at the same time, a lot of families have extra parking spaces but can't benefit from it. We have developed this parking space rental system to solve this problem. The system allows private parking space owners to rent out their parking spaces, thus increasing the number of available parking spaces. At the same time, drivers can search and reserve these parking spaces on the website, thus making it easier to find a parking space.

During the design and implementation of the system, we followed a series of development cycles, each with clear objectives and expected outcomes. Our project was scheduled with phased demos and a final demo in week 10.

During the process, we implemented a lot of features, including user registration and login, displaying a list of parking spaces, searching, filtering parking spaces, and booking and paying for parking spaces. We also implemented user ratings and feedback on parking spaces, where users can rate parking spaces and get parking recommendations, a points system that allows users to deduct fees through points, and many other features.

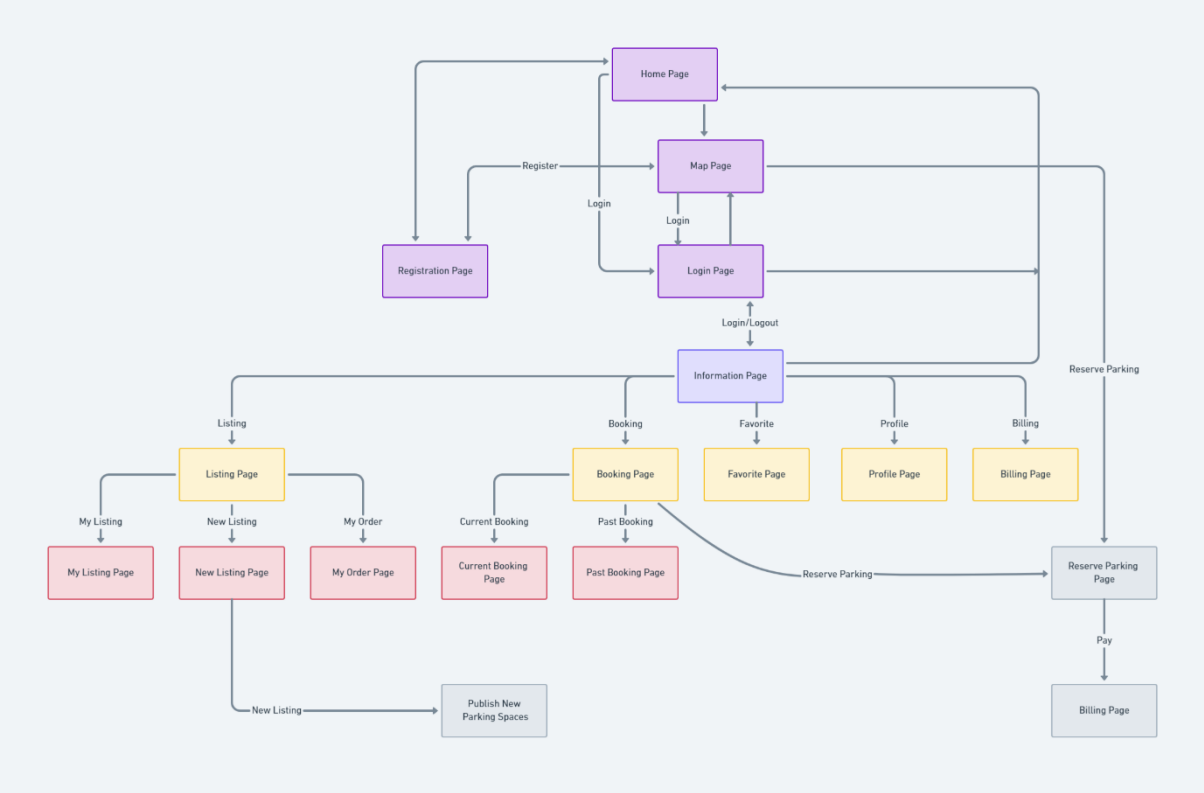
1.2 page structure

The page structure of this project is designed to be user-friendly and easy to navigate and includes the following main pages:

Home Page: The home page consists of a search box and a map. Users can enter the name of the place in the search box and the map will show the parking spaces and their prices in the corresponding area. In the upper right corner, there are registration and login buttons, users can use these two buttons to jump to the corresponding page.
Map Page: The map page displays a list of parking spaces. Users can search and filter parking spaces on this page. After clicking on a parking space, the details of the parking space will pop up at the top of the map page. If the user clicks Book or Favorite at this point, the system will jump to the login page.
Registration: This is where we can register for a new account and fill in our personal information.
Login Page: Users can login to their account on this page.
Information Page: After login, the registration and login buttons in the upper right corner will be changed to information buttons. Users can use this button to jump to the information page. There is a list on the left side of the information page which includes the following sections:

List: Users can view and manage their posted parking spaces here, including modifying and deleting them. In addition, users can also view information about their parking spaces being booked, as well as post new parking spaces.
Reservations: Users can view the currently reserved spaces and completed reservations here.
Favorites: Users can view the list of their favorite parking spaces here.
Personal Information: Users can view and edit their personal information here, including username, password, contact information, and so on.
Billing: Users can view their card information, balance, and points here.
Logout: Users can logout through this button, the system will jump to the login page.

Page Structure Flowchart:



1.3 system architecture

1.3.1 system design

The system design of the parking space rental system is based on the Spring Boot framework, which simplifies the development of Spring applications by providing a convention over configuration approach. The entry point for the application is SpringbootSqliteApplication.java, which initializes the Spring Boot application. SpringbootSqliteApplication.java is the entry point of the application, which starts the Spring Boot application. When we run this class, Spring Boot automatically configures our Spring application, which is achieved through Spring Boot's auto-configuration feature. When running such, Spring Boot automatically configures the Spring application, including initializing controllers, services, repositories, and other components.

The system runs in a coordinated manner where the user sends a request, the controller receives the request and invokes the service, the service invokes the repository to manipulate the database and get the result, then the service returns the result to the controller, which converts the result into the format of the response model and returns it to the user. If an error occurs during the running of the application, the exception handler catches the error and returns an error message.

1.3.2 front-end design

User interaction handling: In this project, user interaction is mainly realized through React's event handling mechanism. For example, in the "Login" component, there is a function that handles the user's login, which will be triggered when the user clicks the login button. In the "Register" component, there is also a function that handles user registration, which is triggered when the user clicks the Register button. These functions are implemented using React's event handling mechanism, making the handling of user interactions more intuitive and easier to understand.

Front-end and back-end communication: In this project, the front-end and back-end communication is mainly realized through the fetch API. For example, in the "fetch.js" file, defined a function for sending HTTP requests, this function can be used to send GET, POST, PUT, DELETE and other requests to the back-end. This function is designed and implemented in a way that makes the communication between the front-end and the back-end more concise and efficient.

Application entry: In this project, the front-end entry is the "index.js" file. This file is the starting point of the React application and it is responsible for rendering the entire React application. In this file, the React and ReactDOM libraries are imported, then the App component is imported, and finally the App component is rendered to the page using the ReactDOM render method. This process is a key step in starting a React application.

Componentized design: In this project, the front-end design follows the principle of componentization. All user interfaces are composed of a set of components. These components include page-level components, such as "Home", "Map", "Info", etc., as well as smaller components, such as "Accordion", "Alert", "BankCard", "Dialog", etc. These components are independent and reusable. These components are independent and can be reused, which makes the front-end design more modular and easier to maintain.

State Management: In this project, state management is implemented through Redux, a library for managing state in JavaScript applications. In this project, Redux is used to manage the user's login status, information about parking spaces, and other state. This makes the management of state more centralized and easier to understand and maintain.

Styling: In this project, the styling is realized by CSS and Sass. CSS is a language used to describe the style of HTML documents, and Sass is a CSS preprocessor, which provides some

powerful functions , such as variables , nesting , mixing and so on , which makes the design of the style is more flexible and easier to maintain .

1.3.3 back-end design

Service: The service layer contains the business logic which acts as a bridge between the controller and the repository. For example, when UserController receives a request to get user information, it will call UserService to handle the request. UserService will call UserRepository to get the user information from the database and return it to UserController.

Repository: the repository is responsible for interacting with the database. Each repository corresponds to an entity, such as UserRepository.java interacts with the User table. The repository uses Spring Data JPA to simplify database operations. Spring Data JPA is a library based on JPA (Java Persistence API), which provides a simple way to implement CRUD (create, read, update, delete) operations.

Configuration: Configuration classes (e.g., MyConfig.java) provide configuration information, such as database connection information, that is required for the application to run. This configuration information is read by Spring Boot at startup and used to initialize the application.

Exception Handling: When exceptions occur in the application, the exception handler catches them and converts them into user-friendly error messages. These error messages are caught by the controller and returned to the user. For example, when a user requests a non-existent resource, NotFoundException.java is triggered and an error response is returned.

Response Model: An application uses a uniform response model to return a response. The response model defines the format of the application's response. Whether it is the result of normal business processing, or an error caught by an exception handler, it is converted to this format and then returned to the user by the controller.BaseResponse.java and ErrorResponse.java define the standard response format for an application. This ensures that all responses from an application have a uniform format.

1.3.4 database design

Our project uses SQLite as a database, which is a lightweight database. SQLite is an embedded SQL database engine that does not require a separate server process and allows the entire database to be stored on disk as a single cross-platform file. SQLite supports transaction processing, allowing multiple queries to be executed at once and the results to be submitted after all the queries have been successfully executed. SQLite database files can be shared between different machines and operating systems.

In this project, we can use SQLite to store user data, configuration information, and even application state information. SQLite can also be used to implement complex queries and data analysis functions.

In this project, SQLite is used as a lightweight solution for storing and retrieving data. Since it is serverless, it can be directly integrated into our application without additional setup or

configuration. In addition, since SQLite supports transactions, we can ensure data consistency and integrity.
Entity represents a table in the database. Each entity class corresponds to a database table. In our project.
User.java corresponds to the User table.
Explain.java corresponds to Explain table.
Favorite.java corresponds to the Favorite table.
Parking.java corresponds to the Parking table.
Reserve.java corresponds to the Reserve table.
Stall.java corresponds to the Stall table.
Each property in the entity class corresponds to a column in the table.

1.3.5 interface design

Each functional module in the project is designed as a separate component. This design approach allows each component to be developed and tested independently, improving development efficiency and reusability, and improving code maintainability.
The project's fetch.js file encapsulates the network request interface, so that other components can call these interfaces to get data, without having to care about the specific details of the network request . This design approach reduces the coupling between components, making the code easier to maintain and extend.
In some components, such as History, Info, etc., state management is used to control the behavior of the component. This design approach allows the state of a component to be shared between components, improving the maintainability and readability of the code.
In some components, such as Accordion, Alert, Dialog, etc., event handling is used to respond to user actions. This design approach allows components to change their state in response to user actions, improving the user experience.
In some components, such as BankCard, Favorite, Stall, etc., data flow is used to control the behavior of the component. This design approach allows the behavior of the component to change according to the data, improving the maintainability and readability of the code.
Controllers are part of the MVC (Model-View-Controller) architecture that handles user requests and returns responses. In this project , each controller corresponds to an entity , our project contains UserController.java to handle requests related to the User entity , ExplainController.java to handle requests related to the Explain entity , FavoriteController.java to handle requests related to the Favorite entity, FileUploadController.java handles requests for file uploads, ParkingControllers.java handles requests related to the Parking entity, and ReserveControllers.java handles requests related to the Reserve entity. requests, and StallController.java handles requests related to the Stall entity.
When a user sends a request, the request is mapped to the appropriate controller method. The controller method calls the appropriate service to process the request and return a response. This design approach allows the various parts of the system to work independently and in concert at the same time, improving the maintainability and scalability of the system.

2. Functionalities developed

2.1 Home

The main page allows user to register, login and search car space. Also, if the user scrolls down, the car space will display in the map.

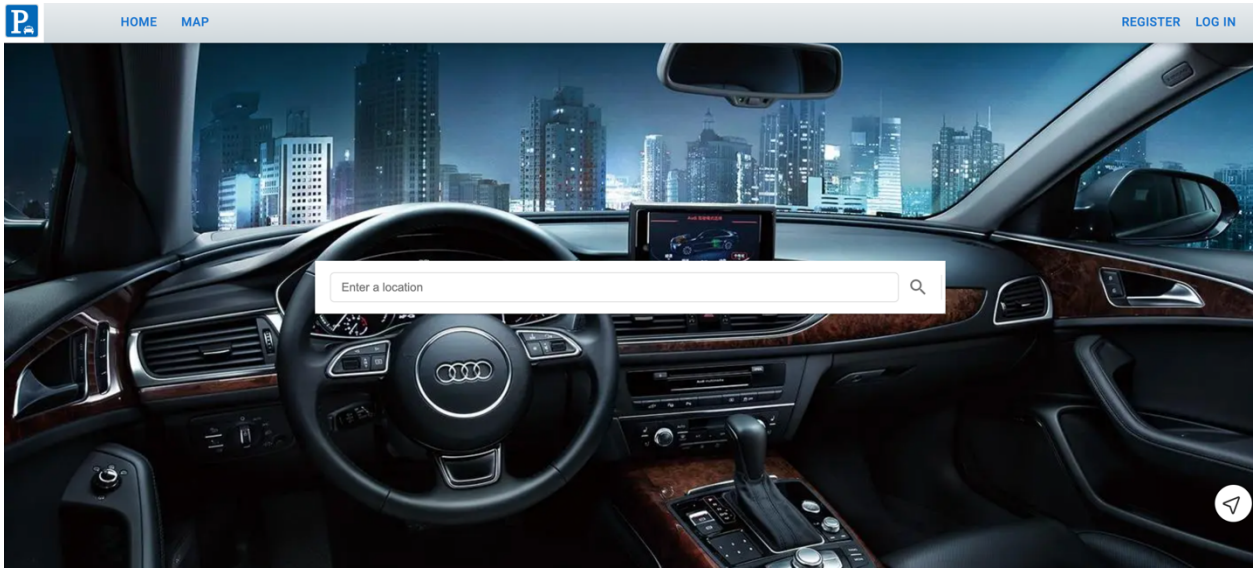


Figure 2.1.1 Home page1

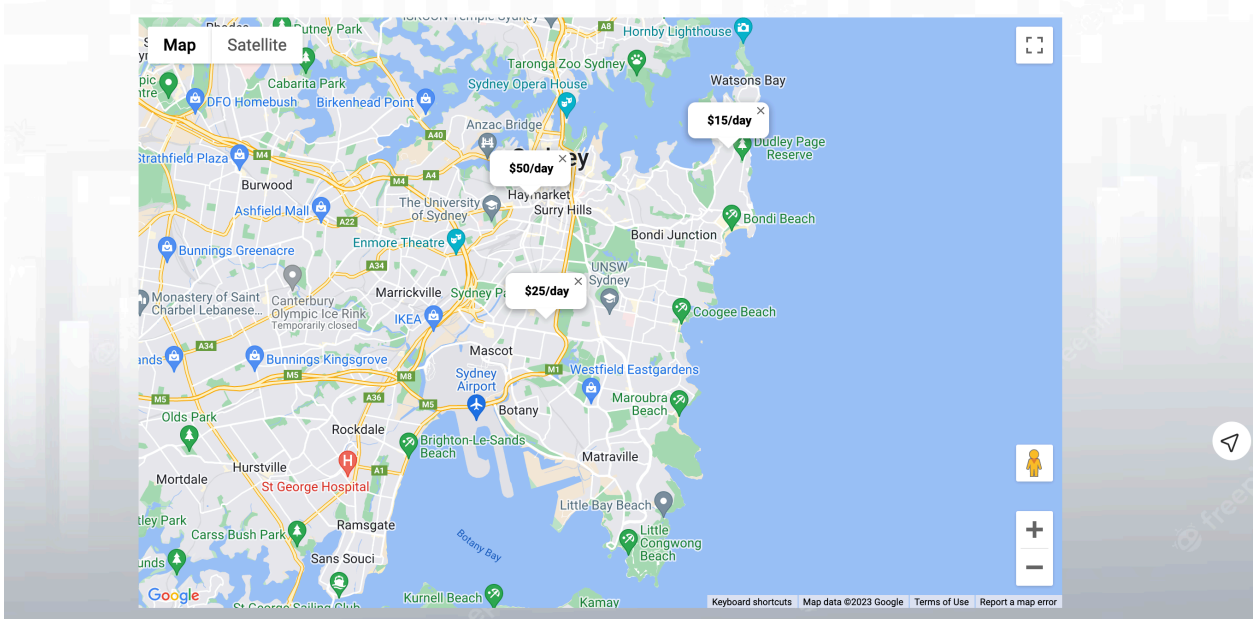


Figure 2.1.2 Home page2

Condition	Result
Click “REGISTER” button.	Navigate to the register page.
Click “LOG IN” button.	Navigate to the login page.
Click “HOME” button.	Navigate to the home page.
Click “MAP” button.	Navigate to the map page.
Input location in the search box.	Get the recommended car spaces.

Input location and click search icon.	Navigate to map page with location searched.
Click search icon without input location.	No response.
Click price icon.	Navigate to Map page and display corresponded car space details.

2.2 Registration

The registration page corresponds to **Provider 1**, **Consumer 2**, and **Admin 2** in user stories. It allows user to input some personal information. If the user input invalid messages, system will alert related messages to inform user to reinput.

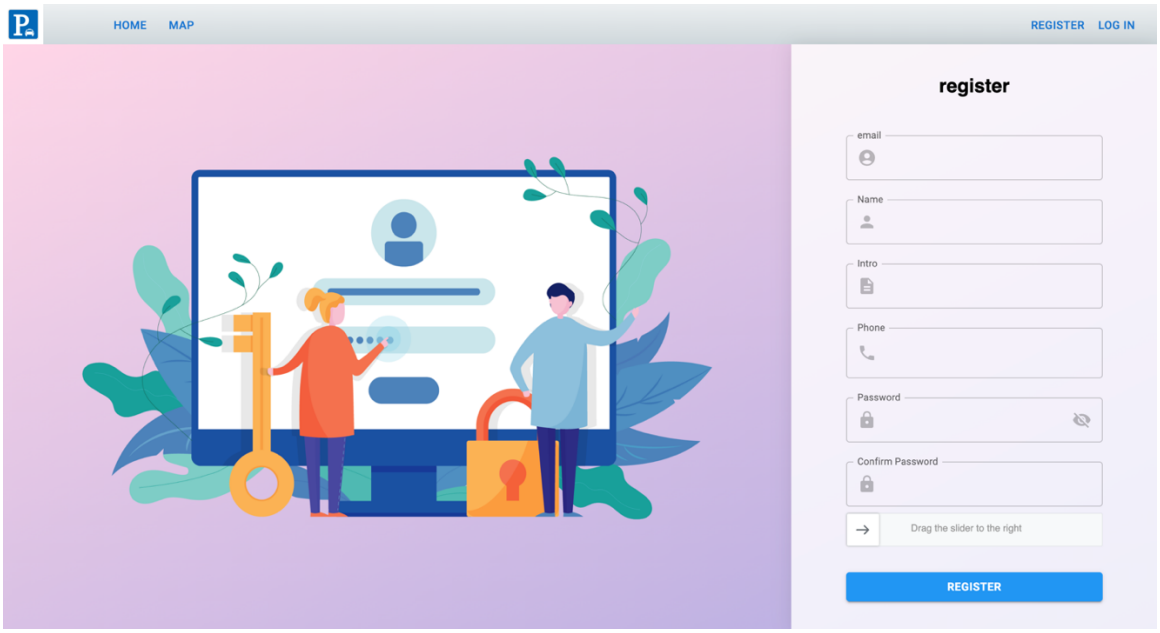


Figure 2.2 Registration

Condition	**Result**
Email is invalid	Alert “Please enter the correct email address” message.
Phone number not 10 digits	Alert “Please input correct phone number” message.
Password • does not contain lowercase letter. • does not contain uppercase letter. • does not contain number. • less than 8 digits.	Alert related message.
Either Name or Intro box is empty.	Alert “Complete information entry” message.
Confirm Password does not match the original password.	Alert “Two passwords do not match” message.
Human confirmation is failed.	Alert “The man-machine check is failed” message.

Click “Register” button without previous errors.	Navigate to the Login page.

2.3 Login

The Login page corresponds to **Provider 1**, **Consumer 2**, and **Admin 2** in user stories. Users who login correctly will navigate to the Information page.

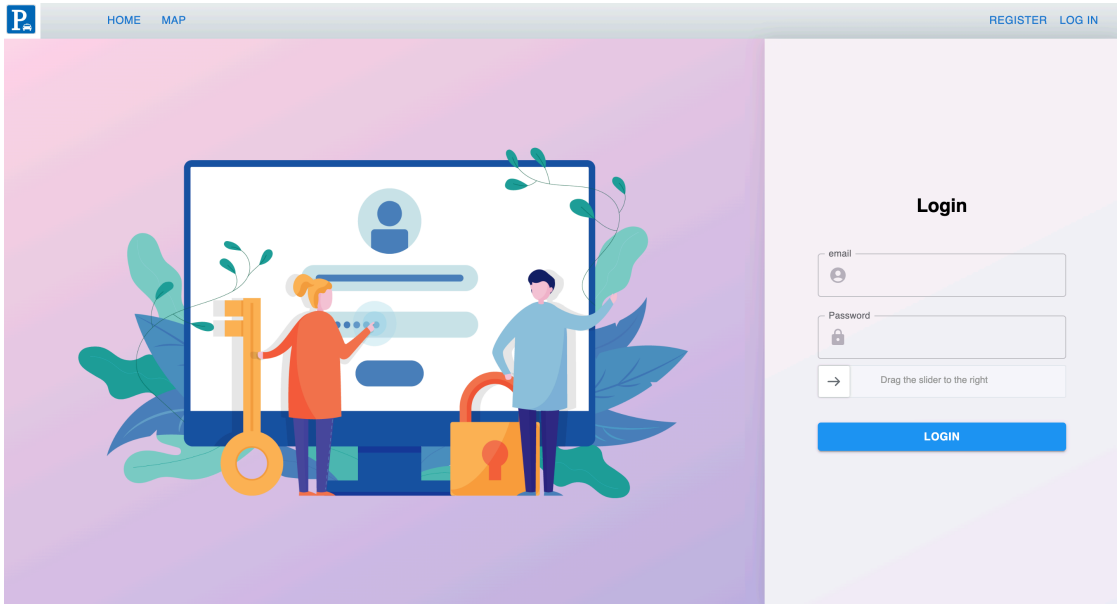


Figure 2.3 Login

Condition	**Result**
Either Password or Email is invalid	Alert “Invalid nickname or password” message.
Human confirmation is failed.	Alert “The man-machine check is failed” message.
Click the “Register” without previous errors.	Navigate to Information page.

2.4 Map

The Map page corresponds to **Provider6**, **Consumer3** and **Consumer4** in user stories. It displays registered car parks. User can filter car spaces by price, distance, etc. Also, users can see more details after clicking one car space.

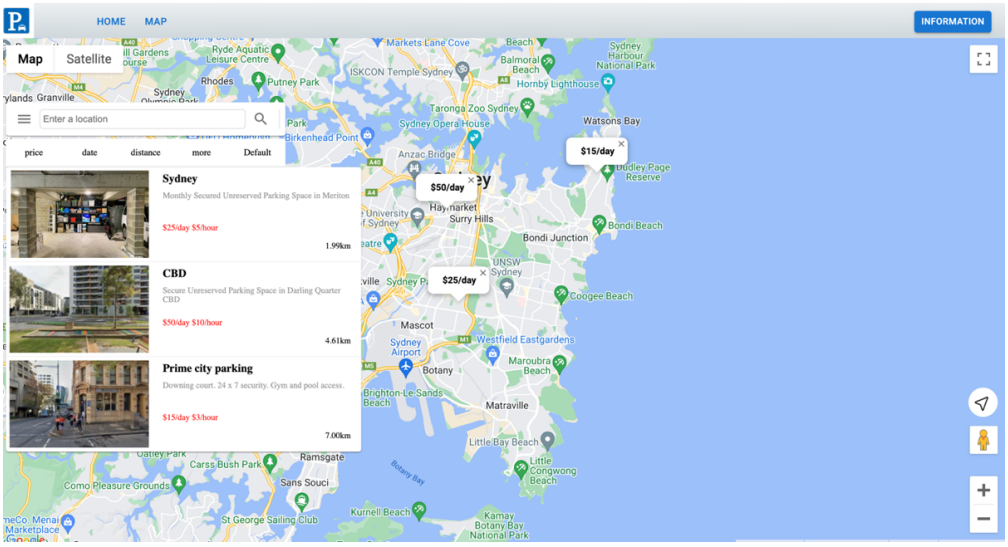


Figure 2.4.1 Map1

Condition	**Result**
Input location in the search box.	Get the recommended car spaces.
Click “Map” button.	Display terrain map.
Click “Satellite” button.	Display satellite map.
Click three-line icon.	Hide car space list.
Click “price” filter.	Display price input box.
Click “date” filter.	Display date input box.
Click “distance” filter.	Display distance input box.
Click “more” filter.	Display space type, way to access and vehicle type input box.
Click “favorite” filter.	Display favorite page.
Click one car space.	Display corresponded car space details.

After clicking “price” filter.

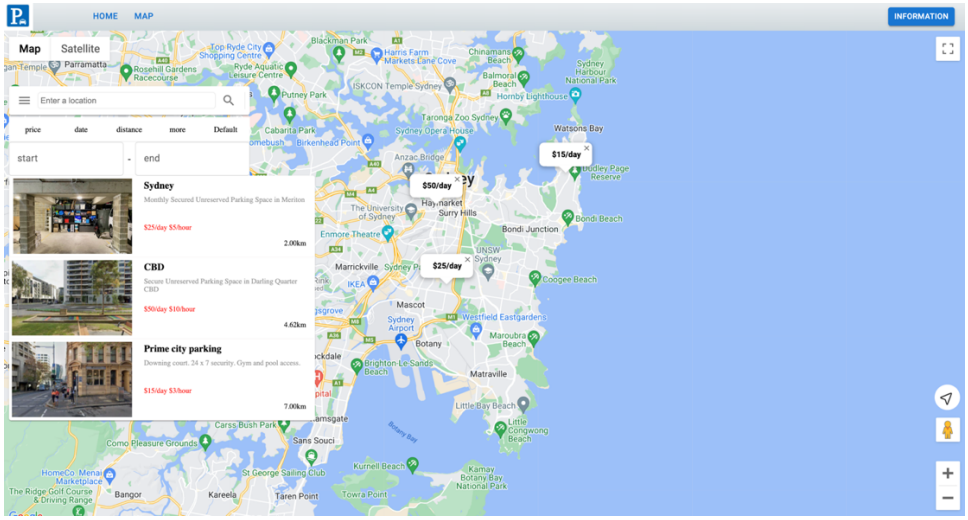


Figure 2.4.2 Map2

After clicking “date” filter.

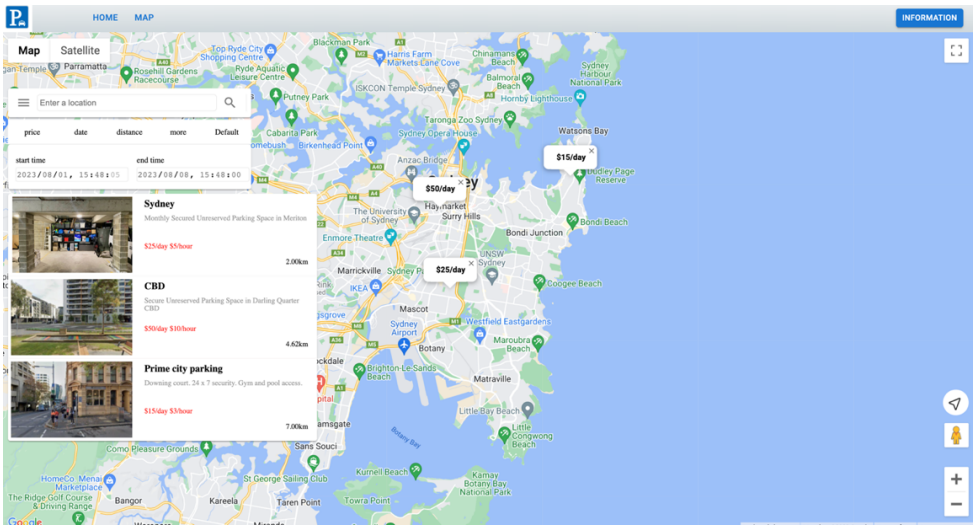


Figure 2.4.3 Map3

After clicking “distance” filter.

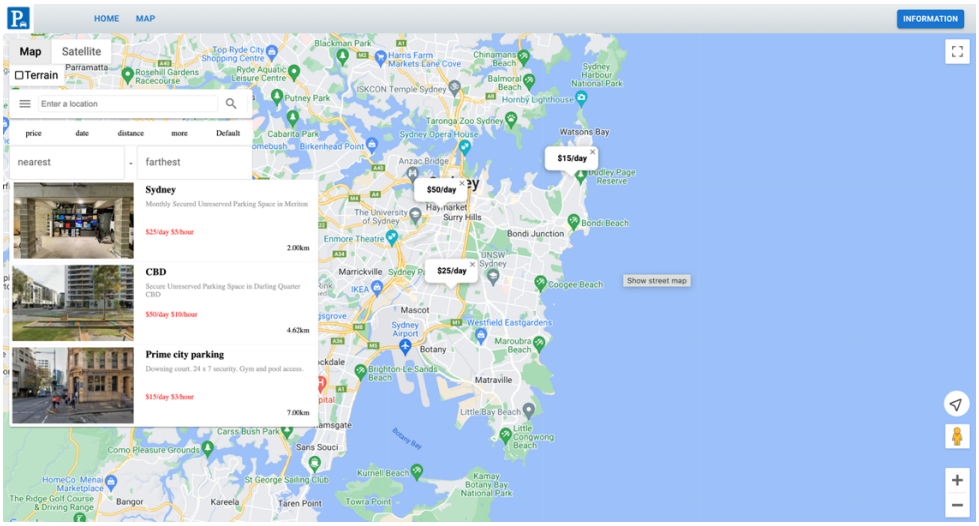


Figure 2.4.4 Map4

After clicking “more” filter.

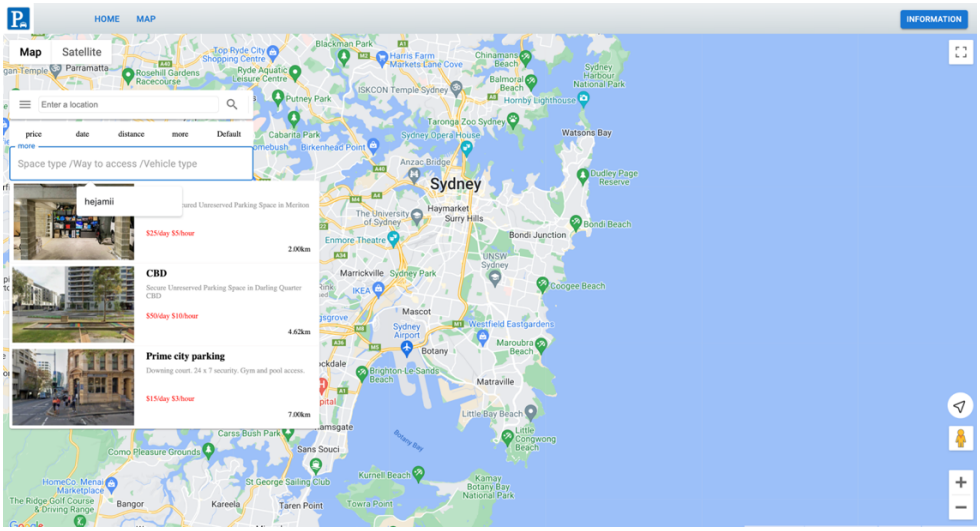


Figure 2.4.5 Map5

After clicking one car space.

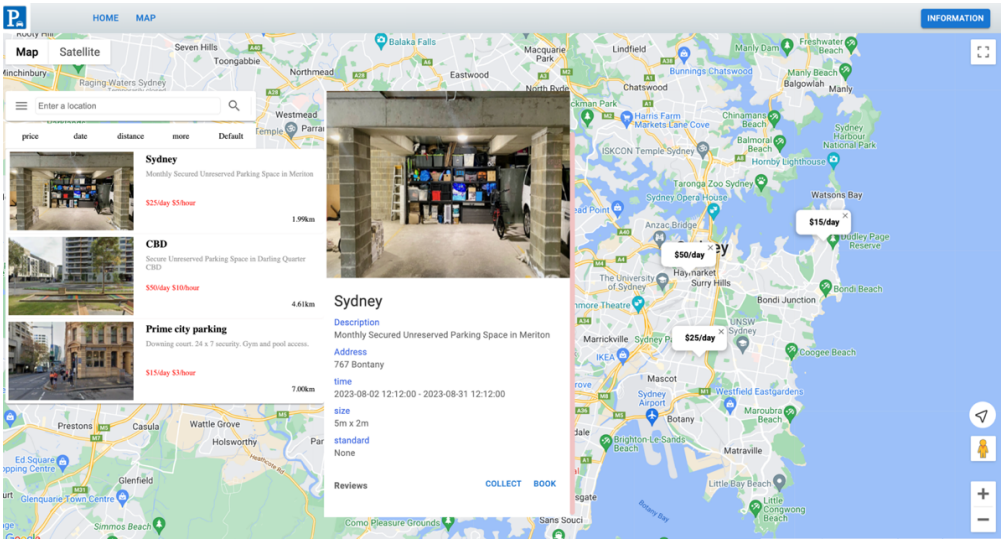


Figure 2.4.6 Map6

2.5 Listing

The Listing page corresponds to **Provider2** and **Provider4** in user stories. It displays providers car space details and booking details.

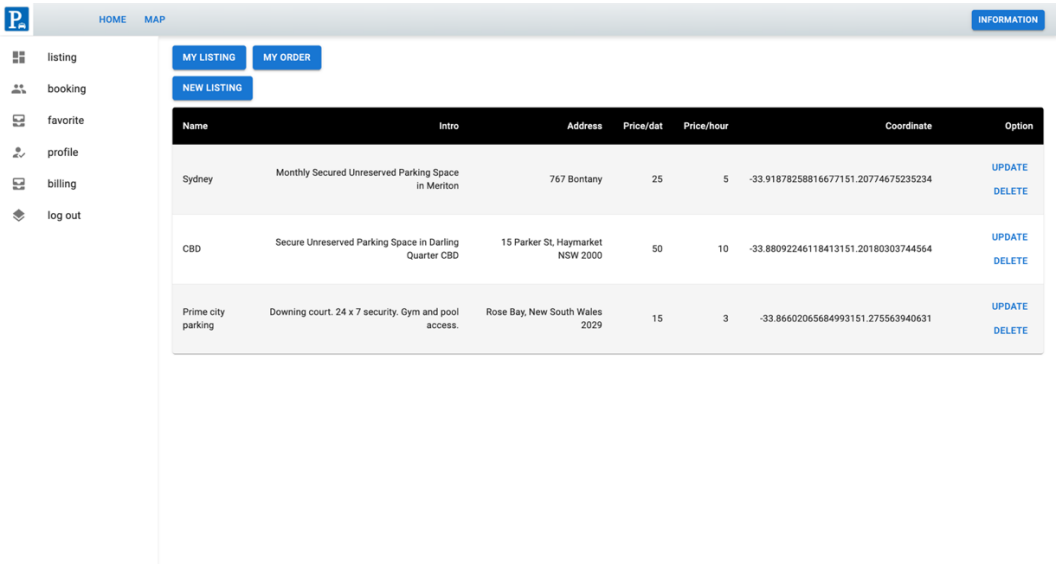


Figure 2.5.1 Listing1

Condition	**Result**
Click “MY LISTING” button	Display provider car space listing page.
Click “MY ORDER” button	Display car space order by user’s page.
Click “NEW LISTING” button	Display new listing page.
Click “Update” button	Display provider car space details page.
Click “DELETE” button	Delete the car space

After clicking “MY ORDER” button.

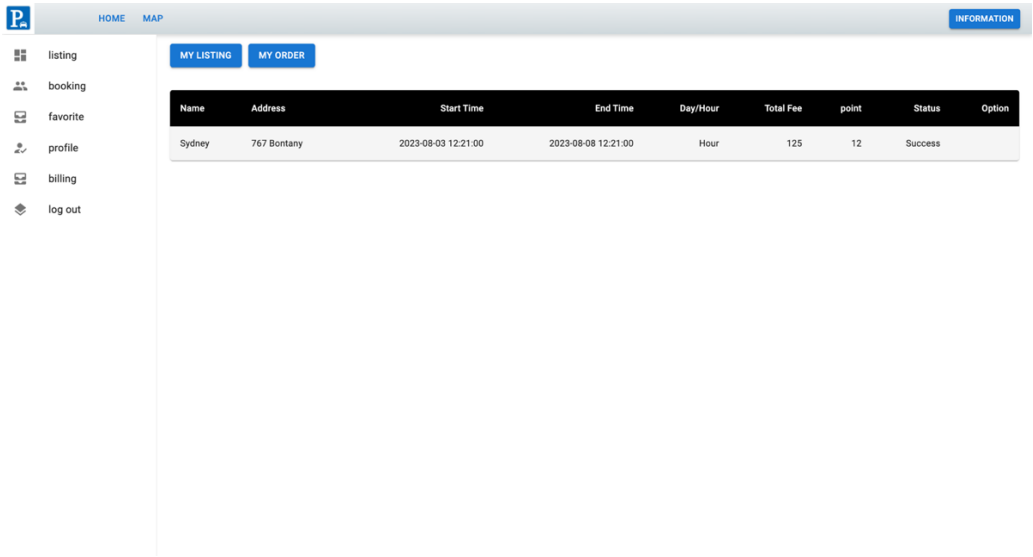


Figure 2.5.2 Listing2

After clicking “NEW LISTING” button.

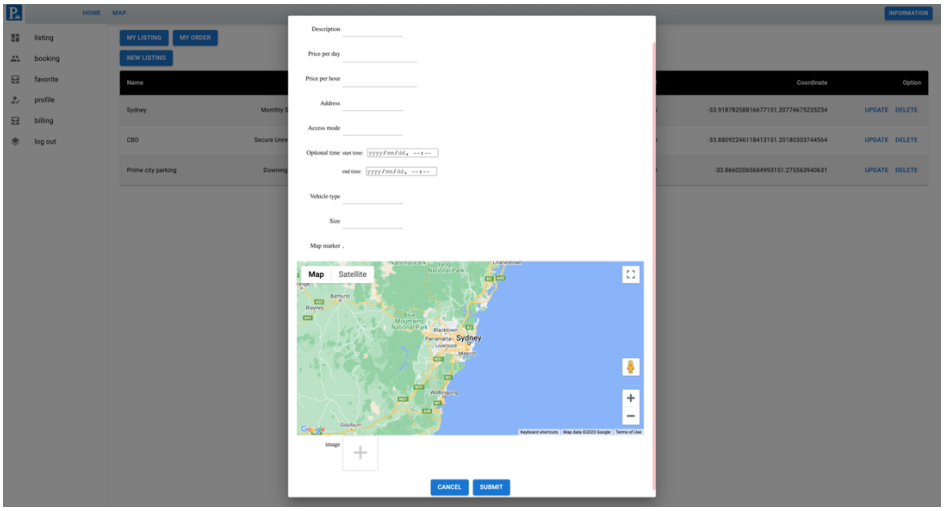


Figure 2.5.3 Listing3

Condition	**Result**
Information is empty.	Alert related message.
Start time is after end time.	Alert “The end time should be before the start time” message.
Start time is before current time.	Alert “The start time should be longer than the current time” message.
Click “CANCEL” button.	Cancel new car space.
Click “SUBMIT” button without previous errors.	Reserve new car space.

After clicking “Update” button.

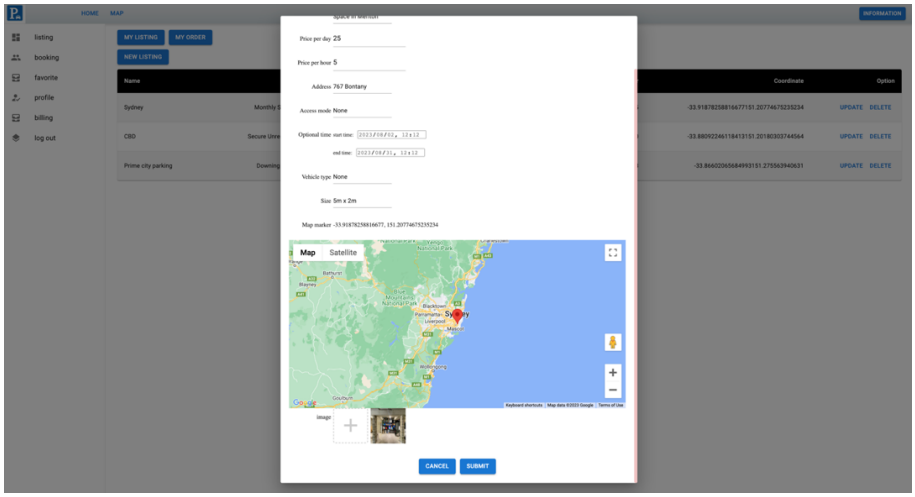


Figure 2.5.3 Listing4

Condition	Result
Click “CANCEL” button.	Cancel update.
Click “SUBMIT” button.	Reserve update.

2.6 Booking

This part includes the **Provider5, Customer1, Customer5, Customer6, Customer7, Customer10** and **Customer11** in the proposal.

For the booking part, we have three buttons to make the option of the orders. And for the current booking we have two buttons to cancel or comment the booking order.

The main page of Booking shows some information of the booking order, such as the Name, Address, the total fee etc. And for the “CURRENT BOOKING” there is a button “CANCEL” letting the consumers and admin can make the option to cancel the booking order. And there is a button “COMMENT” letting the consumers and admin can finish the order by leaving a comment of the booking order.

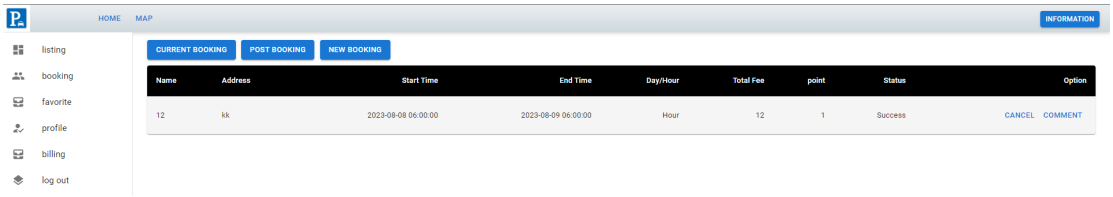


Figure 2.6.1 The current booking page

Condition	Result
Click the “CURRENT BOOKING”	The page shows all the booking orders those are in progress.
Click the “PAST BOOKING”	The page shows all the booking orders those already finished.
Click the “NEW BOOKING”	Switch to the Map page to make new booking.

Click the “CANCEL”	The start time of the booking is more than 24 hours ahead the time now, the consumer and admin can cancel the booking order.
Click the “COMMENT”	Let the consumer to leave a comment on the booking order, and the admin can also finish the order manually by clicking this button.

For the “PAST BOOKING”, this page shows the information of the finished booking orders including their status.

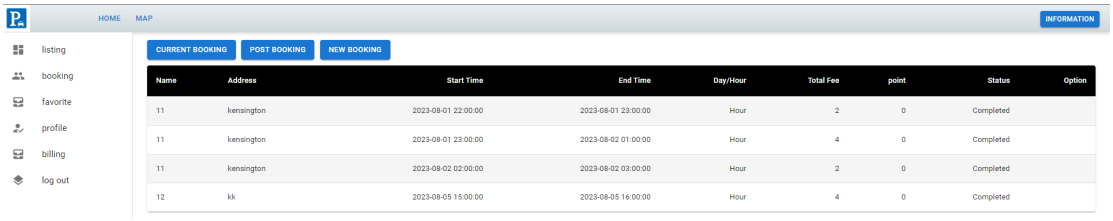


Figure 2.6.2The past booking page

For the booking functionality in the main page, when the consumer start a booking order, we set a timer than limit the consumer to pay the bill in 5 minutes, and the website will automatically calculate the total fee of the booking order.

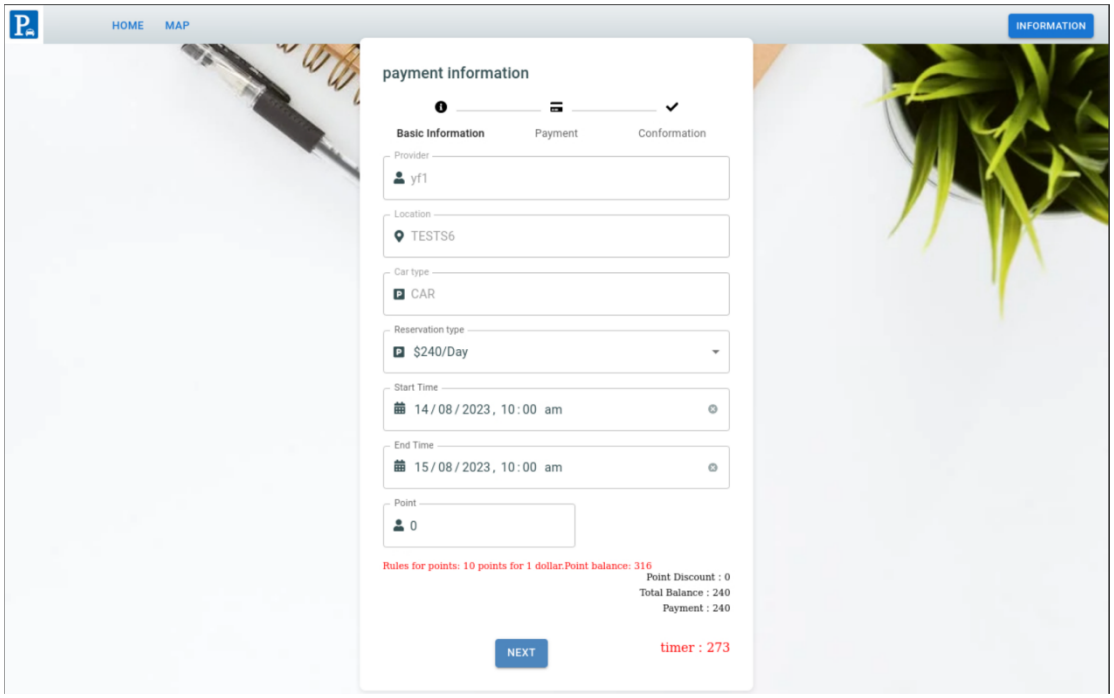


Figure 2.6.3 Bill page

When the user click the “COMMENT” button, the user can leave a comment to the parking space he has booked in the corresponding order.

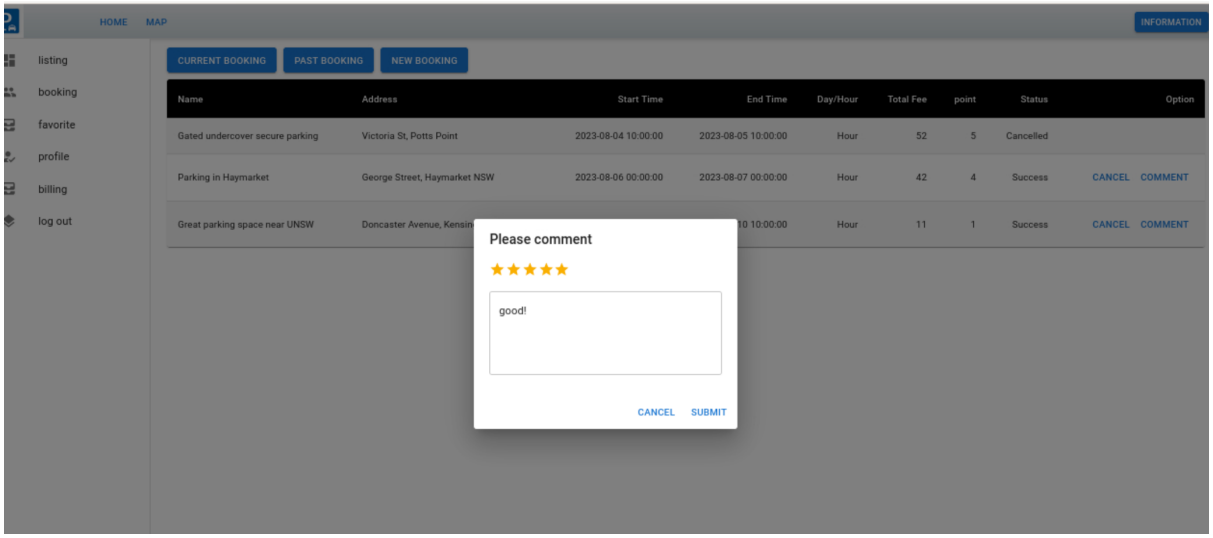


Figure 2.6.4 Comment booking page

When the user click the “CANCEL” button, the user can cancel the order 24 hours ahead the starting time of the order.

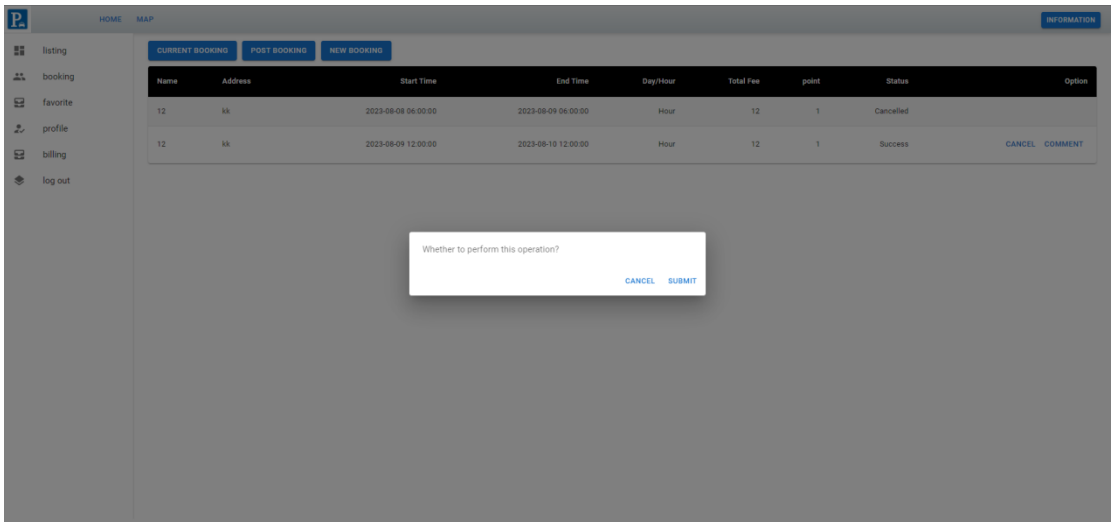


Figure 2.6.5 Cancel booking page

2.7 Favourite

When the user switch to the favourite page, all the parking spots he collected will be saved in the favourite list as shown below.
And the favourite page also shows the information of the parking spaces the user liked.

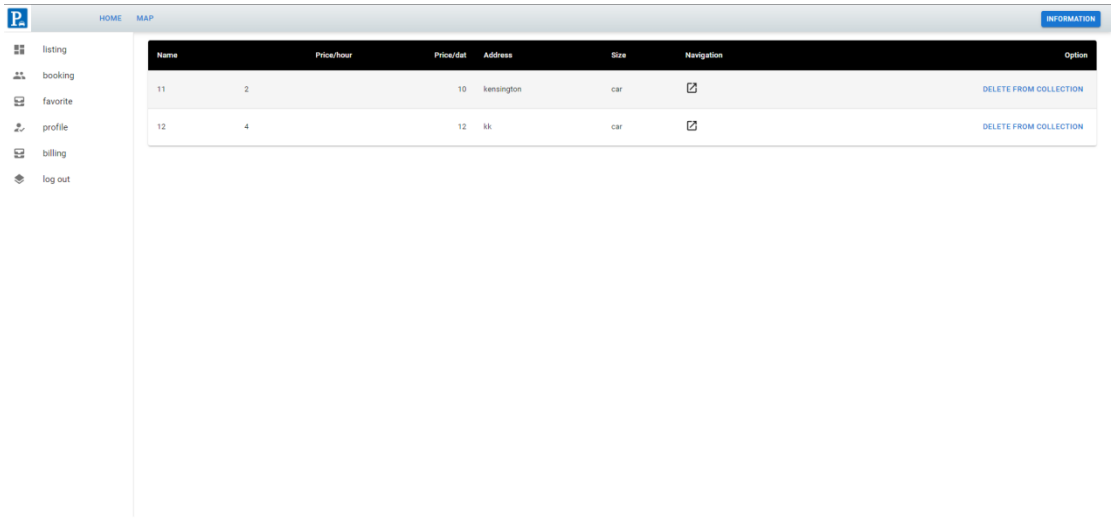


Figure 2.7.1 Favourite page

Condition	Result
Click the “DELETE FROM COLLECTION”	The parking space which was chosen by the user will be deleted from the like list of this user.

There is also a button named Delete from Favourite in the favourite page can delete the parking space were added into the favourite list by the customer.

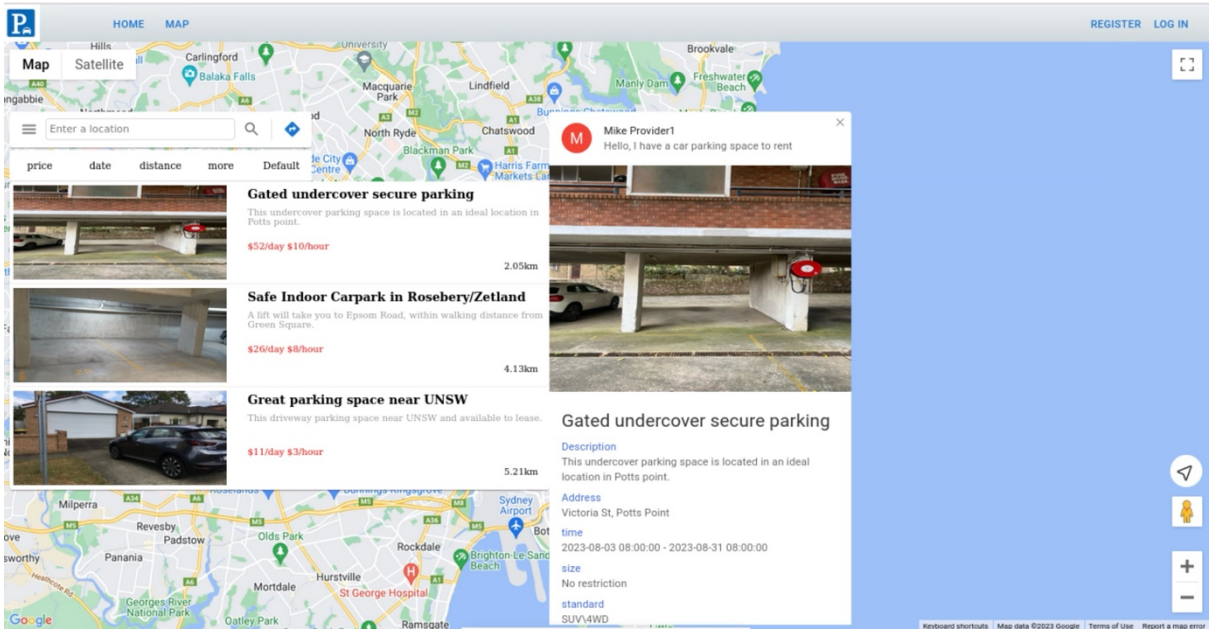


Figure 2.7.2 The map page

This part includes the Customer8 in the proposal.

Condition	Result
Click the “COLLECT”	The parking space which was chosen by the user will be added into the like list of this user.

When the user switch to the map page and then click a parking space information and there will be a button named “COLLECT”, when you click on this button, this parking space will be collect into the user’s favourite list, and the “COLLECT” will turn to “COLLECTED” shows that this parking space has been collected into the favourite list.

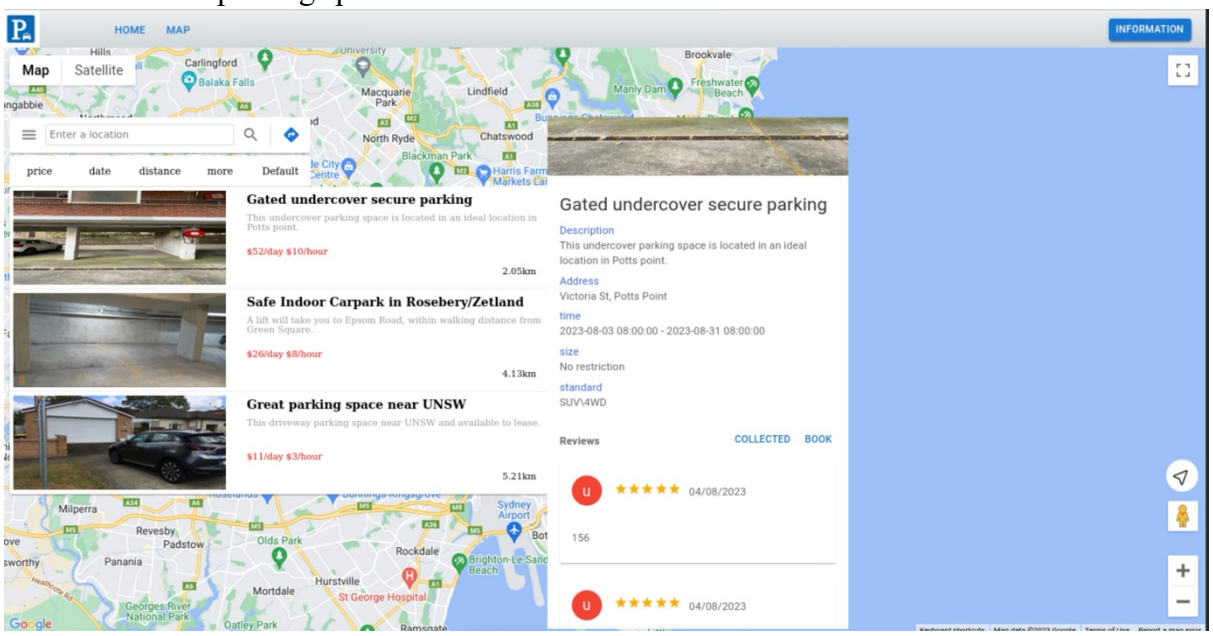


Figure 2.7.3 The map page

2.8 Profile

This part includes the **Customer9** in the proposal.

The profile part shows the main information of the user, including the Email (unique and can’t be changed after registered), Name, Intro, Phone number of the user, and the last three information can be changed by the user himself/herself.

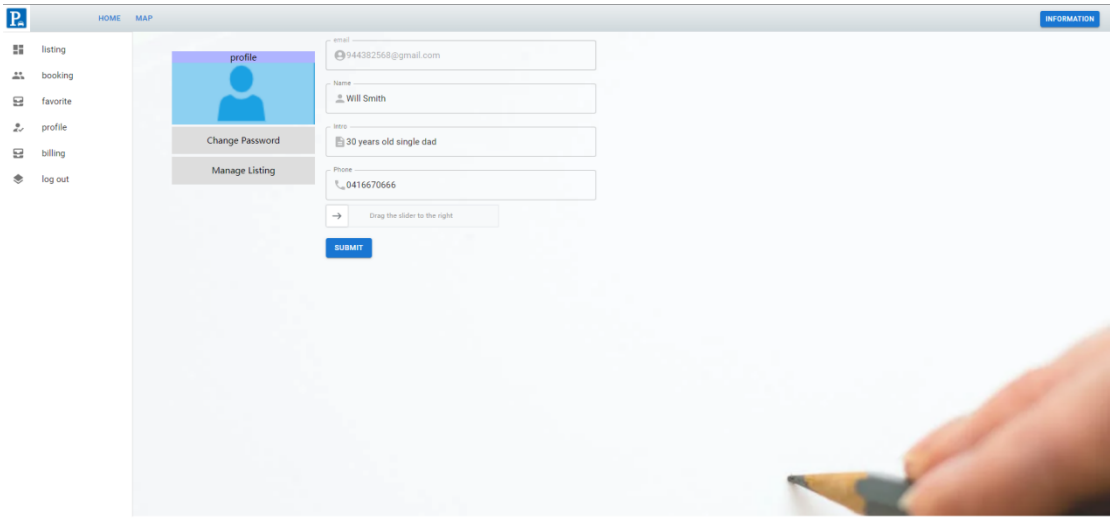


Figure 2.8.1 The profile page

Condition	Result
Click the “Change Password”	The user will switch to the page allow him/her to change the password.
Click the “Manage Listing”	The user will switch to the Listing page to manage the listing he/she updated on the website.
Click the “Submit”	The change of user’s information will be submitted and saved.

The Change Password button: the button provides the option to change the user’s password, and it needs the user to fill in the blanks of the previous password and the new password then the user should confirm the new password, then make the Man-machine verification. And then the user can submit the change of password or cancel the changing option.

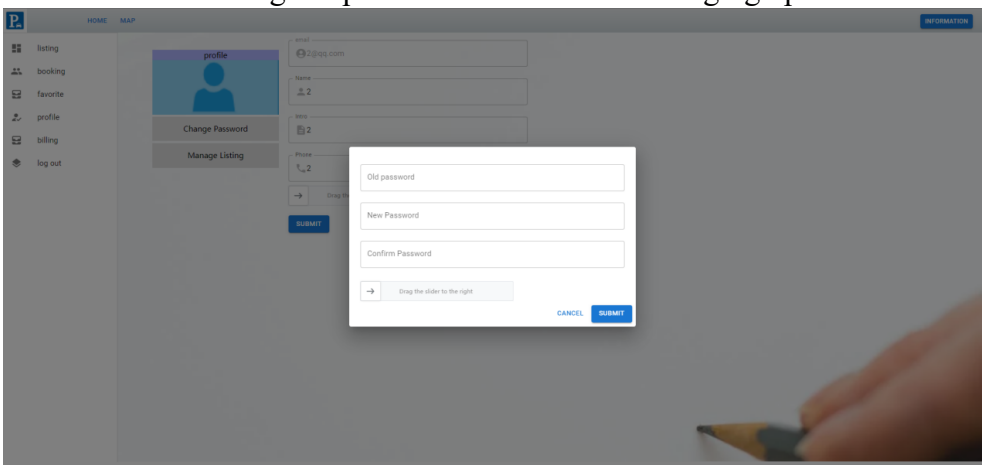


Figure 2.8.2 Change Password Page

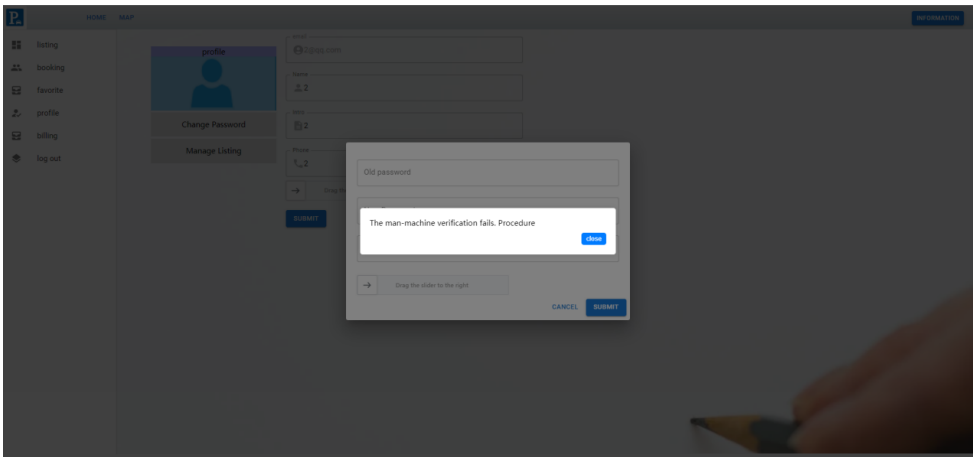


Figure 2.8.3 Change Password Page(Warning for no human-machine verification)

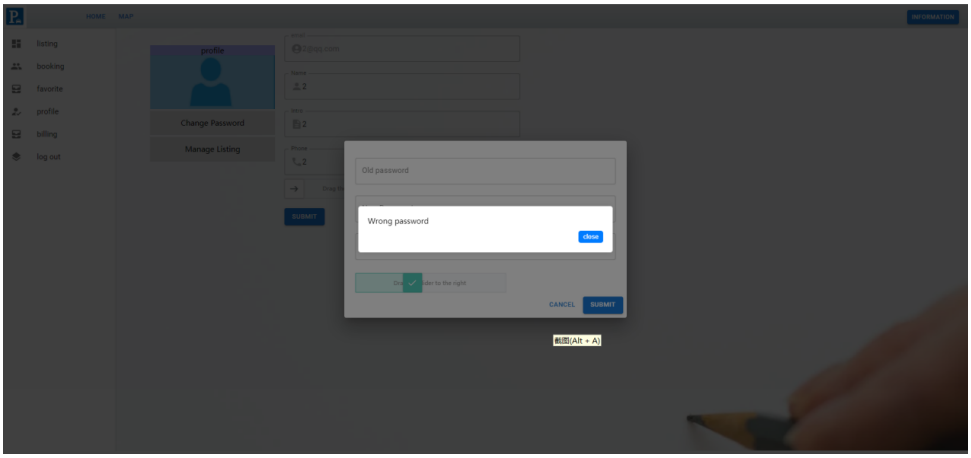


Figure 2.8.4 Change Password Page(Warning for the wrong past password)

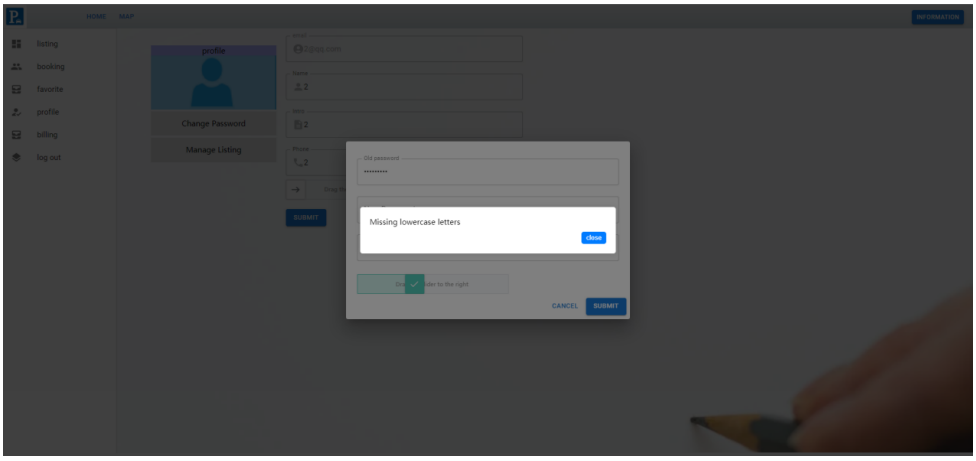


Figure 2.8.5 Change Password Page(Warning for the wrong password format)

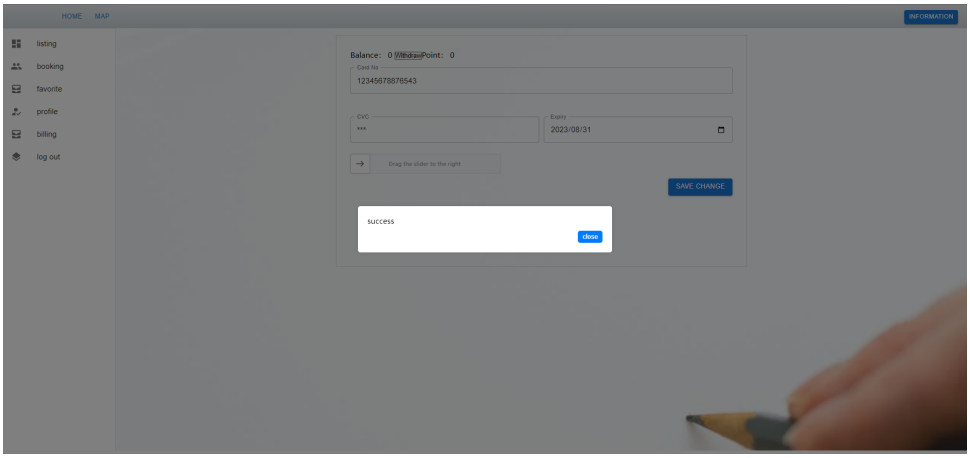


Figure 2.8.6 Change Password Page(The password has been changed successfully)

2.9 Log out

The log out button: this button allows the user to log out his/her account and switch to the main page of the website.

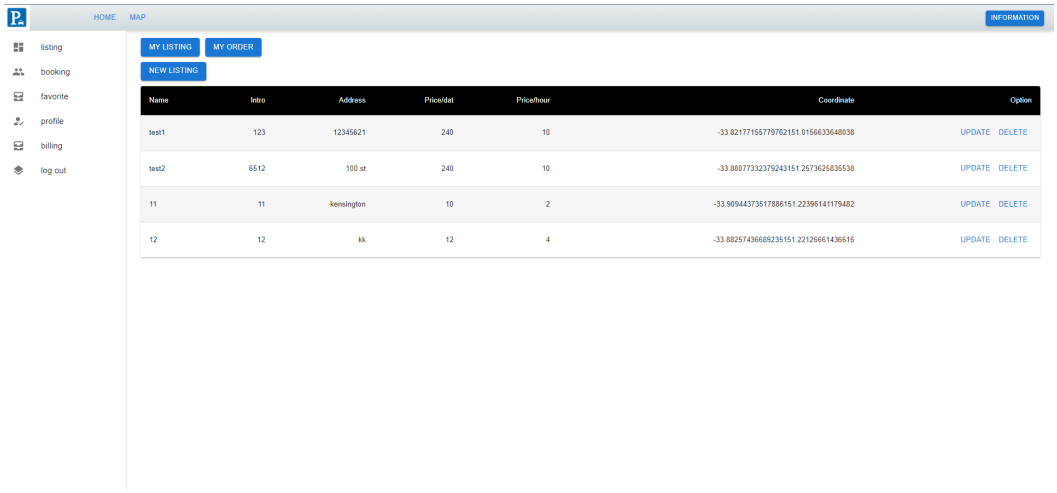


Figure 2.9.1 The information page

Condition	Result
Click the “Log out”	The user will log out and switch to the main page.

2.10 Billing

This part includes the Provider3 in the proposal.
The Billing page has the blanks for user to fill in the bank information and then withdraw their earning and this page also shows the point balance for of the user, and the web can verify the valid time of the card, and also asks the user to make a human-machine verification before they withdraw the earning.

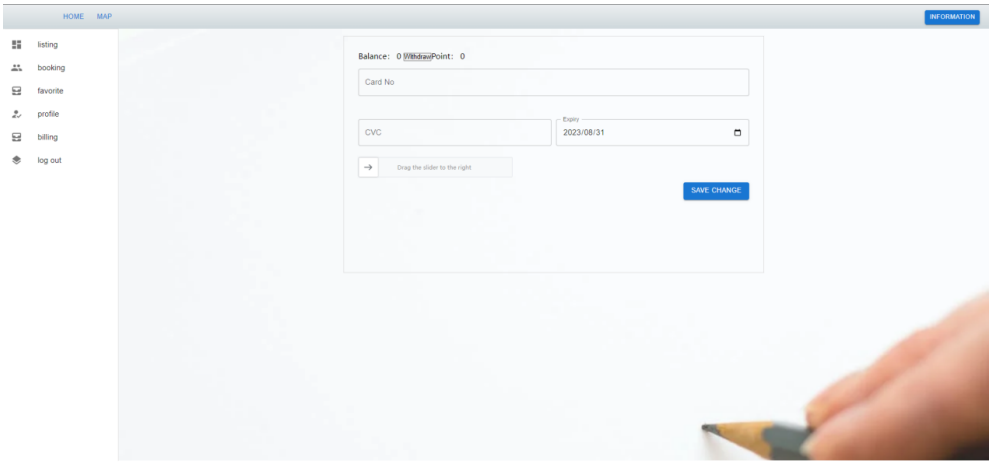


Figure 2.10.1 The billing page

Condition	Result
Withdraw	The balance will be transfer to the bank account that the user saved in the billing page.
Save Change	The user can save the bank information of him/her and withdraw the money to the bank account.

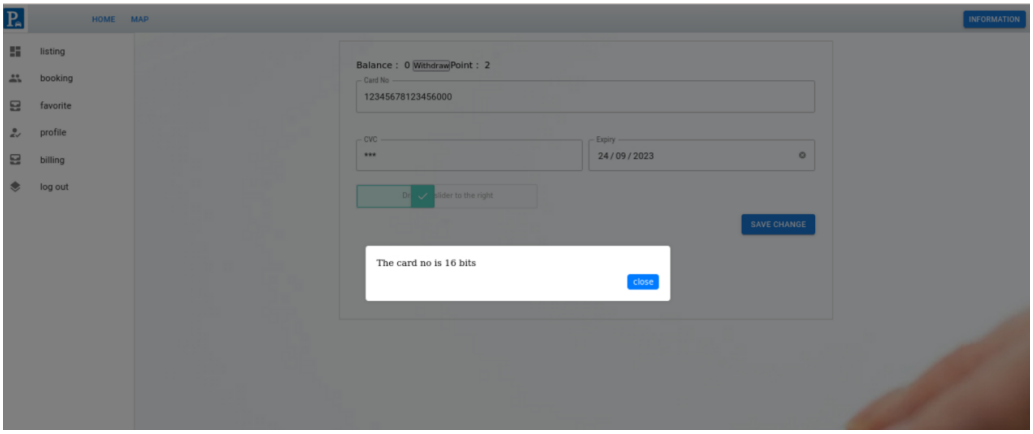


Figure 2.10.2 Warning for the wrong format of the bank account number, the bank account number should be 16 digit.

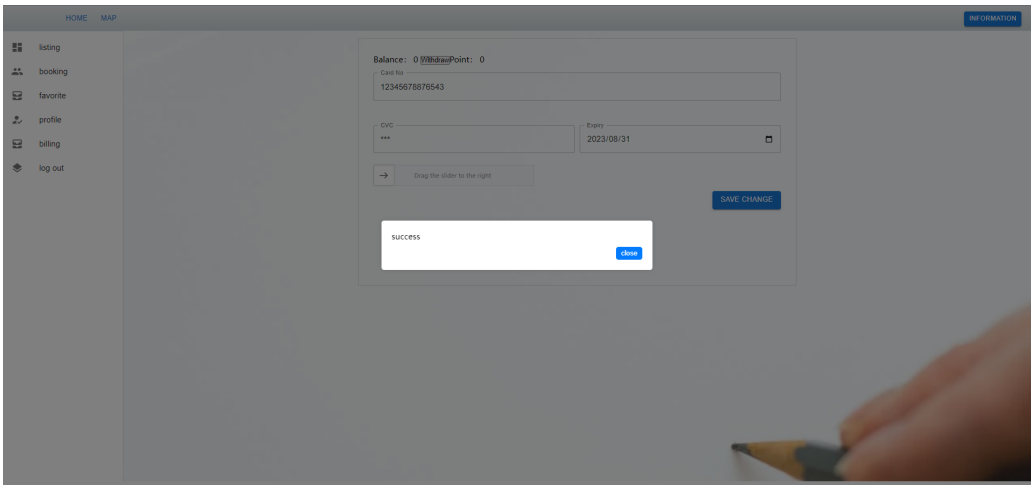


Figure 2.10.3 Withdraw the money successfully

2.11 The authority of Admin.

This part includes the **Admin1** in the proposal.

The Admin has the highest level authority in the system, he can manage the listings created by other users like updating the information and delete the listing, and he can also manage the booking order, he can delete the booking order of any user and he can also finish the order manually.

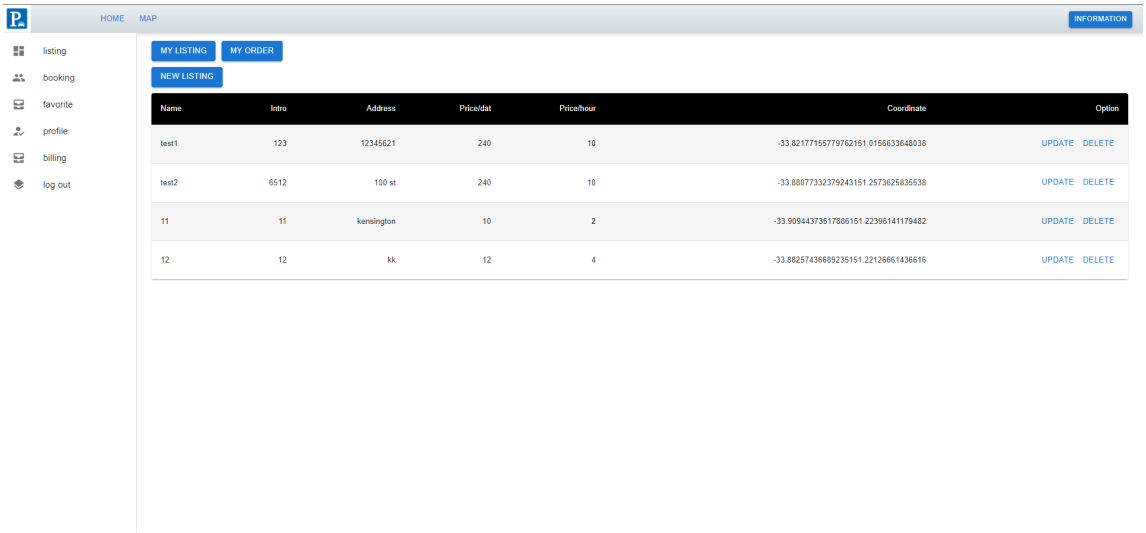


Figure 2.11.1 The listing page

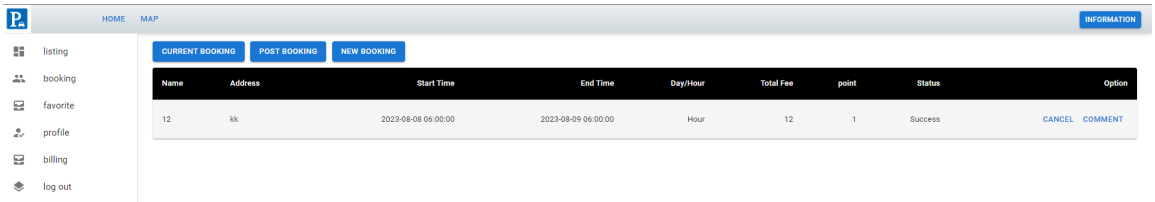


Figure 2.11.2 The booking page

Condition	Result

UPDATE	Update the information of the parking space like what the provider does.
DELETE	Delete any one of the listing on the website.
CANCEL	Cancel the order.
COMMENT	Can finish the order with no rating and comment manually.

3. Third-party functionalities

3.1 Frontend

3.1.1 UI-react

React is an open source JavaScript library developed and maintained by Facebook for building user interfaces, especially single page applications. It allows developers to build complex UI interfaces with independently reusable components through the idea of componentisation.

In our car parking space rental website, we use React to create dynamic front-end pages, including but not limited to the following sections:

Car Parking Spaces Listing Page: this page shows all the available car parking spaces for renting. Each parking space is a separate component that displays basic information about the parking space, such as location, size, price, and so on. Users can use search and filter functions to find the car park they want.

Parking Space Details Page: When a user clicks on a parking space, they are navigated to this page. This page shows the details of the car park, including details of location, price, size, pictures, etc. Users can book a parking space on this page.

User Login and Registration Page: These two pages are used for user registration and login. We use React's form processing and validation features to implement these two pages.

User Personal Centre Page: On this page, users can view and manage their personal information such as username, password, contact details, and so on. They can also view and manage their rental information, such as booked spaces, rental time, and so on.

React's component-based design makes our code more modular and easier to maintain and extend. For example, we can create a "parking space" component and reuse it on the parking space listing page and the parking space details page. When we need to change the way the parking spaces are displayed, we only need to modify this component, not on each page separately.

In addition, React has a powerful state management feature where we can use React's states and props to manage and pass data. For example, we can use state to hold a user's login information and then pass that information to the components that need it via props.

Overall, React is a powerful front-end framework that allows us to quickly create and manage complex user interfaces.

3.1.2 Google map API

Google Maps API is a set of interfaces provided by Google that allows developers to embed Google Maps in their websites or applications. In our car parking space rental website, we use Google Maps API to provide mapping services.

Using Google Maps API, we can display the location of car parking spaces on a map, and users can use the map to find and select car parking spaces. In addition, we can also use other features provided by Google Maps API, such as map markers, information windows, etc., to provide richer functionality and better user experience.

Here are some of the main features we use with Google Maps API:

Display Maps: We display Google Maps on the parking space listing page and the parking space details page. users can see the location of the parking spaces on the map, which can help them better understand the location of the parking spaces.
Map Markers: We use markers on the map to indicate the location of the parking spaces. When a user clicks on a marker, they can see detailed information about the parking space, such as location, size, price, and more.
Information Window: When a user clicks on a map marker, we display an information window showing the details of the parking space. This provides a more intuitive display of information and improves the user experience.
Overall, Google Maps API is a powerful interface for map services, which enables us to embed maps in our website, provide rich map functionality and improve user experience.

3.2 Backend

3.2.1 SpringBoot

SpringBoot is a Java library based on the Spring Framework for creating standalone, production-grade Spring applications. In the development of our car parking space rental website, we used SpringBoot to create the back-end services. The main advantage of SpringBoot is its "convention over configuration" design philosophy, which provides a set of default configurations that allow us to quickly start the project without the need for tedious configuration. configuration. In addition, SpringBoot also has a series of built-in services, such as embedded servers, security controls, data manipulation, etc., so that we can focus on the development of business logic. In our project, we use SpringBoot to handle business logic such as parking query, parking reservation, user registration and login, and interaction with the database.

3.2.2 maven

Maven is a powerful project management and build tool that is primarily used for Java projects. During the development of our car parking space rental website, we used Maven to manage project dependencies and builds.
The main advantage of Maven is its unified project structure and dependency management. By using Maven, we can easily manage all the dependencies of the project, including SpringBoot, Spring Data JPA, JUnit, etc. Maven automatically handles the relationships between dependencies, downloads and installs the required libraries, allowing us to focus on writing the code.
In addition, Maven provides a range of build and packaging tools. We can use Maven to compile and test code, package projects as executable JAR files, and even deploy projects to servers.
Here are some of the main features we use Maven for:
Dependency management: We declare all the dependencies of our project in the pom.xml file and Maven automatically downloads and installs them. This allows our project to run consistently in any environment.
Project build: We use the mvn clean install command to clean up the results of previous builds, compile and test the code, and then install the packaged project to the local repository.

This command is part of the Maven lifecycle and performs a series of tasks in a predefined order.

Project Run: We use the mvn spring-boot:run command to run the project. This command will start the SpringBoot application and we can visit the application URL in the browser to see the results of the run.

Overall, Maven is a powerful project management and build tool that enables us to quickly manage and build projects, providing a stable and efficient development environment.

4. Implementation challenges

4.1 Frontend

4.1.1 Front-end and back-end interface docking

Front-end and back-end interface docking: The development of front-end and back-end is usually carried out in parallel, which requires a clear definition of the interface at the early stage of development, including the path of the request, method, parameters, return value and so on. Unclear interface definition or frequent changes in the development process, these will have difficulties in the docking work. Even serious refactoring may occur. So we should try to avoid this situation, in the beginning of the project to develop a clear interface document, and in the development process to update.

4.1.2 Data consistency

In this project, both the front-end and the back-end need to handle multiple types of data such as users, parking spaces, reservations, etc. This requires ensuring that the front-end and the back-end handle these data in the same way. For example, the back-end in the database of the parking space information, user information processing, the front-end in the display of these data need to be processed accordingly. If the processing of data is inconsistent, it may lead to incorrect display of data and may even affect the functionality of the system. In order to ensure the consistency of data, we need to take this into account when designing the database and interface, and conduct rigorous testing during the development process.

4.1.3 Security

When dealing with user data, we need to consider the security of the data, such as the user's password, user's bank card number and security code. Security is a very important issue and any security breach can lead to serious consequences. Firstly, in the interface design, we let the user to log in using a complex password structure, including upper and lower case and special symbols. In fact, we added a human-machine authentication to the user's personal information screen. On the user's bank card screen, we have hidden the display to ensure the security of the user's information.

4.1.4 Interface Optimisation

Let's start by discussing the importance of optimising the UI: Optimising the UI has a high impact on the overall website experience. A clear and intuitive interface can help users quickly find the information they need, for example, on the home page users can quickly view the useful information about parking spaces, view the map, etc. An optimised UI reduces the cognitive load on users and increases their satisfaction. The way information is presented in the interface is critical. A good UI should be able to present parking information clearly and effectively so that users can quickly find the information they need. For example, a map view might be a good way to show the location of a parking space, the price, have a list view, and have a search function that allows users to filter and sort the spaces according to their needs (e.g., price, time of day, etc.). Finally, to completely cover the operation logic, users have two main operations on the site: viewing parking spaces and uploading parking spaces. An

optimised UI should be able to provide a simple, intuitive flow that allows users to complete these operations easily.

Next discuss the challenges.The point we want to focus on is that this website needs to manage a large amount of information about car parking spaces. It can be a challenge for us to organise and present this information effectively to meet the needs of different users. We designed filters, sorters, and map markers to help users filter parking spaces and quickly find the spaces they need. Uploading parking space information requires a form. Designing a good form can also be a challenge. When uploading a parking space, we consider that a parking space requires a lot of information to describe, so the uploader is required to refine the form content, and any one item left out is not allowed in our opinion. We provide feedback to users to let them know that their information has been successfully uploaded and what information has been missed.

4.1.5 Google Maps API

In our project, Google maps API is a challenge, mainly based on its complexity, performance and customisation, and data privacy. The use of Google Maps API is mainly focused on the following aspects: map display (Google Map component), markers (Marker component), and information windows (Info Window component). In the interface Google Maps API is mainly used to display the map. This requires us to read the user's latitude and longitude, initialise the Google map's centroid latitude and longitude and zoom level, followed by the corresponding user actions, including dragging the map or zooming the map. When we add new parking space information, we need to place markers on the map to mark the location of the parking space on the map, which also need to get the coordinates latitude and longitude information. All these operations add complexity to the project. On both the home page and the parking space page, we have map information windows that show the location and price of the parking space on the map, and we can also click on the map to get detailed information about the parking space. Creating and managing these information windows, as well as choosing to show and hide them when handling user responses, also introduced complexity. If you need to display a large number of markers and information windows on the map, this can have an impact on performance. The Google Maps API provides a number of customisation options, such as changing the style of the map (colours, features, etc.), changing the icons of markers, creating custom information windows, and so on. However, these tasks require an in-depth understanding of the API, adding to the workload and complexity. In order to recommend a parking space to a user, we need to have access to the user's location, so you need to ensure the privacy and security of this data. This means that you need to get explicit permission from the user and also do some encryption of the data.

4.2 Backend

4.2.1 Understanding and Using Spring Boot and Spring Data JPA

Spring Boot is a framework that, through its "convention over configuration" design philosophy, automatically configures many of the underlying details, allowing us to focus on writing business logic. When a user wants to upload their parking space information, there's no need to go through the effort of configuring it - Spring Boot already handles that

automatically. But when we use custom configurations, we have to understand how they are auto-configured and how they work, so the whole process becomes complicated. For example, the @EnableAutoConfiguration annotation lets Spring Boot automatically configure an application, but how does this "automatically" work? Spring Boot has a policy for determining which autoconfigurations should be applied, based on dependencies in pom.xml. Understanding this process, and how to override the default behaviour of autoconfiguration, is a challenge.

Spring Data JPA, on the other hand, greatly simplifies the implementation of the data access layer. Developers can define methods directly in the repository interface and Spring Data JPA will automatically generate the corresponding database queries for them. For example, we have a number of repositories that use Spring Data JPA. This provides a number of methods for accessing the database without having to write implementation code. The advantage of this approach is that you can quickly create a repository with basic CRUD functionality. However, understanding how these methods work and how to use Spring Data JPA to perform more complex queries can be somewhat difficult.

Overall, Spring Boot and Spring Data JPA have greatly simplified the development process, and we have been able to learn and understand the applications and principles through multiple sessions where the group has helped each other.

4.2.2 Error processing

In the back-end, it is necessary to deal with a variety of possible errors, such as database operation failure, file read/write errors, user upload failure, operation failure and so on. This requires a well-designed error handling mechanism that can provide users with clear error messages without leaking the internal information of the system.

5. User Manual

5.1 Load the backend

5.1.1 Clone or download the frontend folder

5.1.2 Open a terminal on the frontend directory and execute the following command:

$ mvn spring-boot:run

If it executes successfully, the terminal will show the following messages in the end:

$ mvn spring-boot:run

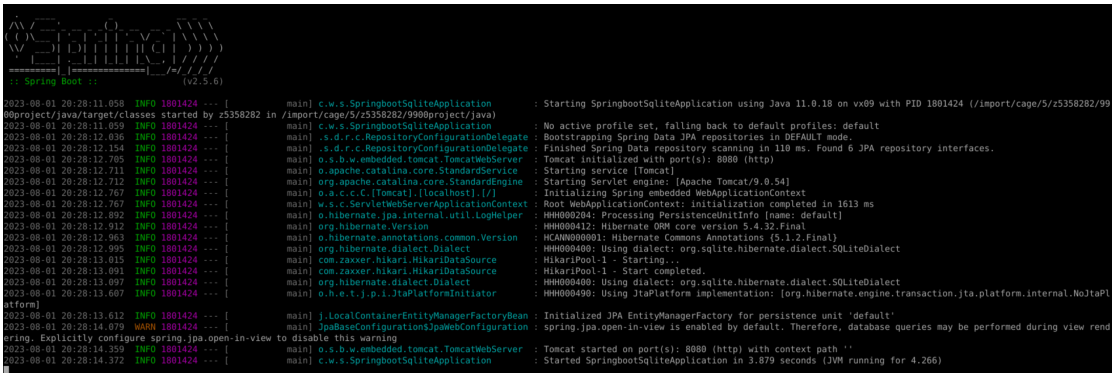


Figure 5.1.2

5.2 Load the frontend

5.2.1 Clone or download the frontend folder

5.2.2 Open a terminal on the frontend directory and execute the following command:

$ npm install

$ npm i sass --save-dev

$ npm run start

If it executes successfully, the browser will pop up as shown below:

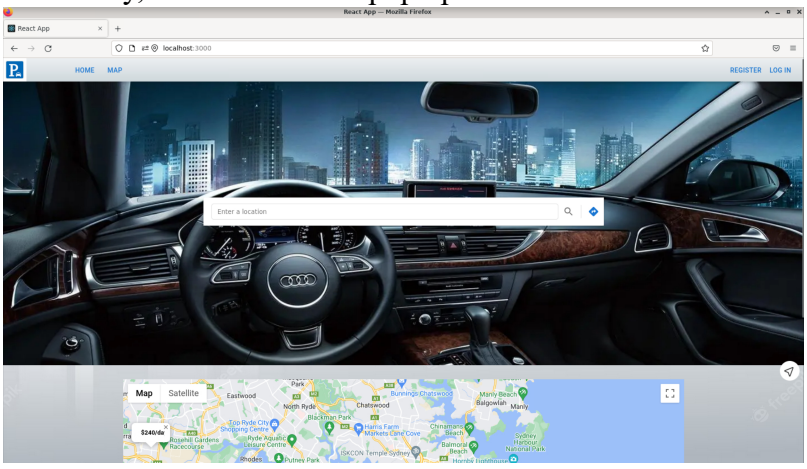


Figure 5.2.2

6. References

1. Building Java Projects with Maven. (n.d.). Retrieved from https://spring.io/guides/gs/maven/
2. Spring integrates React for front-end development----using webpack to package React-related components in a Maven project. (n.d.). Retrieved from https://blog.csdn.net/linfujian1999/article/details/72639194
3. How to Use Sass with CSS. (n.d.). Retrieved from https://www.freecodecamp.org/news/how-to-use-sass-with-css/
4. @react-google-maps/api. (n.d.). Retrieved from https://www.npmjs.com/package/@react-google-maps/api
5. React Router. (n.d.). w3schools. Retrieved from https://www.w3schools.com/react/react_router.asp
6. Material UI. (n.d.). react.school. Retrieved from https://react.school/material-ui
7. JPA - Quick Guide. (n.d.). tutorialspoint. Retrieved from https://www.tutorialspoint.com/jpa/index.htm
8. SQLite - Java. (n.d.). tutorialspoint. Retrieved from https://www.tutorialspoint.com/sqlite/sqlite_java.htm

7. Appendix

User stories

User type

- Provider – Users who want to rent out their parking space through the website.
- Admin - Users with high level privileges and the ability to modify the database.
- Consumer – Users who want to rent a parking space through the website.
- System

User Stories Table

The table below shows a table of the user stories presented in this project.

User	User Stories	Acceptance Criteria
Provider 1	As a provider, I want to register new parking spaces and upload them to the website so that they are available for users to view and book.	Providers can register new parking spaces and enter parking space information: ● Title ● Address ● Photos ● Space type ● Way to access ● Description ● Size of parking space ● Vehicle type ● Price ● Availability
Provider 2	As a provider, I want to be able to manage my parking information and fix existing parking information so that I have control over the accuracy and updates of the parking details.	Users can view their parking space information, change the price, address, and tags of existing spaces, or add new spaces and delete existing spaces.
Provider 3	As a provider, I want to be able to receive the rent that is due to me so that I can ensure timely and consistent payment for my property.	The system can pay the provider's bank account (minus 15% service fee).
Provider 4	As a provider, I want the system to automatically accept available reservations instead of requiring me to manually confirm, so that it would be very convenient for me.	The system can accept reservations for parking spaces based on availability.
Provider 5	As a provider, I want the system to automatically terminate the user's reservation if the consumer doesn't pay in time after booking, so that competitors cannot maliciously take my reservation.	When a user pays for an order during the window period, there is a 5-minute countdown, after which the system must be able to automatically cancel the booking.

Provider 6	As a provider, I want the system to recommend my parking space to potential consumers, so that it can increase the chances of attracting interested users.	Consumers can fill in two addresses they frequent in their personal information, and when the user logs in, the corresponding parking space will be recommended on the homepage according to the location they frequent, or randomly if the user does not fill in the item.
Admin 1	As an administrator, I want to be able to view, edit, and delete existing parking spaces in the database, so that I have full control over managing the available parking options.	The system provides the admin with the ability to manage existing parking spaces
Admin 2	As an administrator, I want the system to authenticate providers, consumers, including administrators, before any sensitive information is updated, so that it ensures security and prevents unauthorized access to sensitive data.	The system can authenticate administrators, providers and consumers before a user can update any sensitive information. Authentication is carried out by human verification. Sensitive information: ● Email Bank card details
Consumer 1	As a consumer, I want to be able to pay my bills online, so that I can conveniently manage my payments without the need for physical transactions.	Consumers can pay for their orders online
Consumer 2	As a consumer, I want to register for the service and input my personal and vehicle details so that I can create an account and provide the necessary information for a smooth user experience.	Consumers can register for the service and enter personal information and vehicle details: ● Name ● Email ● Password ● Vehicle type ● Bank card detail ● Common activity areas
Consumer 3	As a consumer, I want to be able to find a car space from the list of registered parking spaces in the system, so that I can easily locate and choose a suitable parking spot.	The consumer can search or filter from the list of car spaces in the system to find a car space: Filter: ● Price ● Space type ● Distance to the search place ● Way to access ● Vehicle type
Consumer 4	As a consumer, I want to be able to view the details of the selected parking space, so that I can gather relevant information about the	Consumers can view the details of the parking spaces by clicking: ● Title

	parking spot such as location, availability, and pricing.	• Address • Photos • Space type • Way to access • Description • Size of parking space • Vehicle type • Price • Availability • Rate and comments
Consumer 5	As a consumer, I want to be able to book an available car space and specify the duration of the booking, so that I can secure a parking spot for a specific period of time according to my needs.	The consumer can freely select the booking date from the available booking times and the system calculates the price
Consumer 6	As a consumer, I want to be able to view the list of orders and have the option to cancel the reserved parking space, so that I can manage my bookings and make changes if needed.	Consumers can view their order history and cancel orders from their order history
Consumer 7	As a consumer, I want to be able to browse my own bookings, rate and review the spaces I have booked by providing ratings on the platform, so that I can share my experiences and feedback with others in the community.	After completing an order, consumers can view the order, give it a rating of one to five and write a review for the parking space. The system can count the number of individual ratings and display the comments
Consumer 8 (novel)	As a consumer, I want to be able to bookmark my favorite parking spaces, so that I can easily access and find them more quickly whenever needed.	Consumers can bookmark a parking space at any time while browsing the list of spaces and can manage their own list of favorites.
Consumer 9 (novel)	As a consumer, I want to be able to earn points for each of my historical orders, which will grant me discounts on subsequent orders, so that I can enjoy benefits and savings as a loyal customer.	On the checkout page, consumers can see their current accumulated points and choose whether to use them to pay off a part of their order.
Consumer 10	As a consumer, I want to be able to know the exact cost of the reservation I want to make, so that I can have full transparency and clarity regarding the pricing before confirming my booking.	On the checkout page, the total cost of parking can be calculated based on the duration of the booking.
Consumer 11	As a consumer, I want to be able to pay online directly after selecting the desired booking date, so that I can conveniently complete the	The consumer can successfully jump to the payment page after selecting the scheduled date.

	payment process without any additional steps or delays.	