COMP3900 Computer Science Project

Term 1 2022

Project Report

Park It - Car Space Sharing System

Name	E-Mail Address	zID	Role
Tobias Wong	z5255663@unsw.edu.au	z5255663	Scrum Master
Muhammad Abubakar	z5165087@unsw.edu.au	z5165087	Developer
Eu Chieu Ang	z5245630@ad.unsw.edu.au	z5245630	Developer
Younggil Tak	z5192259@ad.unsw.edu.au	z5192259	Developer

Submitted on 21/04/2022 by 3900-W14B-Blue Man Group

Table of Contents

1 User Documentation

1.1 Prerequisites

We recommend that a user who wishes to setup and run our project from their personal device should have the following:

- The Lubuntu 20.4.1 LTS virtual machine image as described here: https://webcms3.cse.unsw.edu.au/COMP9900/22T1/resources/71190

1.2 Initial Setup

1.2.1 Obtaining project from GitHub

To begin setting up our project, please head on over to our project's GitHub repository located at this link here (https://github.com/unsw-cse-comp3900-9900-22T1/capstone-project-3900-w14b-blue-man-group).

Once there, you can make a clone of the repository by clicking on the green "Code" button as shown in the image below (Diagram 1.1)

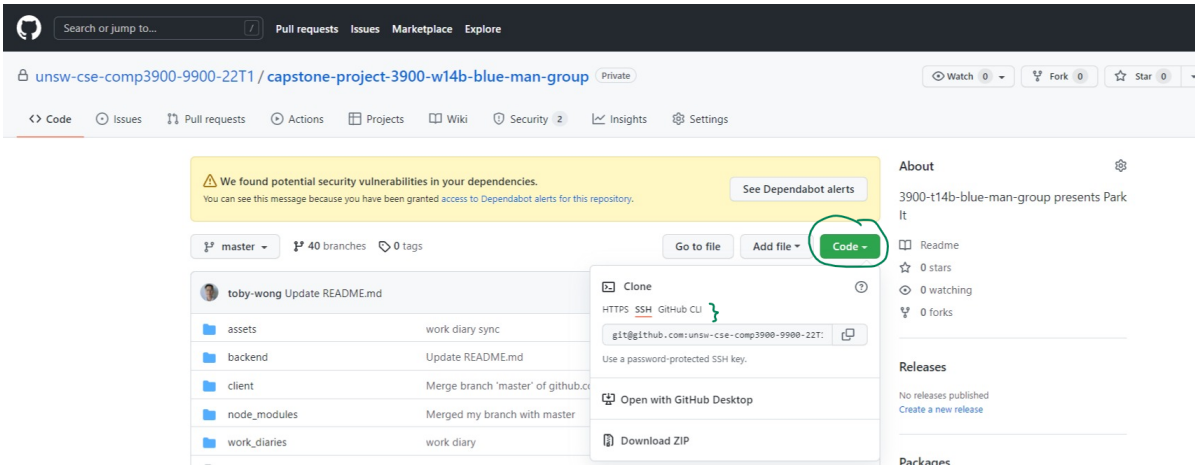


Diagram 1.1 - "Code" button to clone project's GitHub repository

Once you have selected a cloning option (we recommend either HTML or SSH), run the following command in a terminal along with the link that you have obtained from the cloning option from just now:

Command: `git clone <insert link here>`

Note that this command should be run in the root directory of your terminal, so that it would be easier to access the project's directory later on.

1.2.2 Obtaining project from Zip File (Via `give` on CSE)

On the Lubuntu 20.4.1 LTS virtual machine image as described here:
https://webcms3.cse.unsw.edu.au/COMP9900/22T1/resources/71190

Unzip the given file and you should see a folder structure like this:

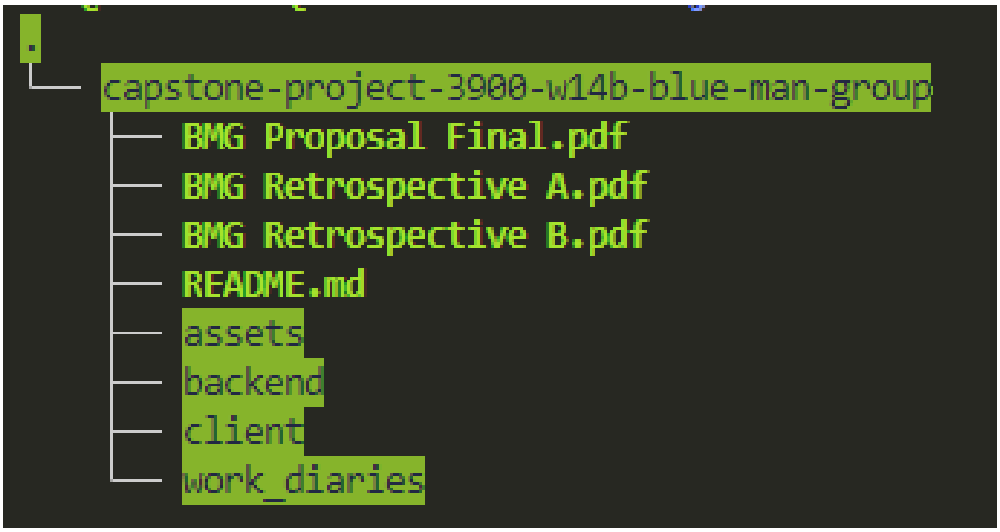


Diagram 1.2 - File structure of unzipped file

1.3 Setting up the Frontend and Backend

This section focuses on setting up the frontend and backend of the project. For an overview of what these components are and how they work, refer to the **Overview** section down below.

1.3.1 Frontend

These instructions will help you run the React-based frontend on the Lubuntu image.

Note that the frontend and backend must be run simultaneously in different terminals.

1 Open a Terminal and run these commands

```
$ sudo apt update  // update package lists from repositories
$ sudo apt install --only-upgrade firefox // upgrade firefox for compatibility
$ sudo apt install curl // install curl to install nvm
$ curl https://raw.githubusercontent.com/creationix/nvm/master/install.sh | bash
```

2 Close that Terminal and open a new Terminal, and then run these commands

```
$ nvm install 14.17.0 // install node 14.17.0 via nvm
$ sudo apt install npm
```

3 In the same terminal navigate to the `client` directory

```
$ cd capstone-project-3900-w14b-blue-man-group/client
$ npm install // please be patient with this. It takes a while to install
$ npm start
```

The frontend will be accessible at `localhost:3000`, but it should open automatically

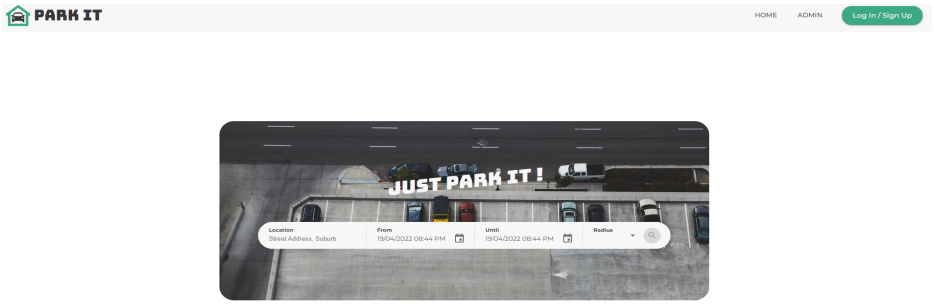


Diagram 1.2 - Expected page after following the instructions above

1.3.2 Backend

These instructions will help you run the Django-based backend on the Lubuntu image.

Note that the frontend and backend must be run simultaneously in different terminals.

Open a new Terminal and run these commands

```
$ sudo apt install python3-pip
$ sudo -H pip install -U pipenv
```

In the same terminal navigate to the backend directory

```
$ cd capstone-project-3900-w14b-blue-man-group/backend
$ pipenv install
$ pipenv shell
$ python3 manage.py runserver
```

```
(backend-dlWwoQX9) bruhbook@chieu.com
→ ~/capstone-project-3900-w14b-blue-man-group/backend python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 19, 2022 - 20:49:29
Django version 4.0.4, using settings 'backend_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Diagram 1.3 - Output from the terminal after setting up backend

1.3.3 Admin Account

```
/*
  This account has staff privileges to test admin and parking spaces
  already inserted, but feel free to make your own account!
*/
email: jzdoog@gmail.com
username: exampleman
password: Dragon123*@
```

1.4 Using the Project

This section contains a guide that will help you navigate the different pages that are available on our web app. Each section will introduce to you the features that we have developed for that page, as well as the intended ways to use it and demonstrate safety nets to prevent unintended usage patterns.

1.4.1 Registering and Logging In

Welcome to Park It's homepage, the main page where all of our users will land when you first arrive at the web app.

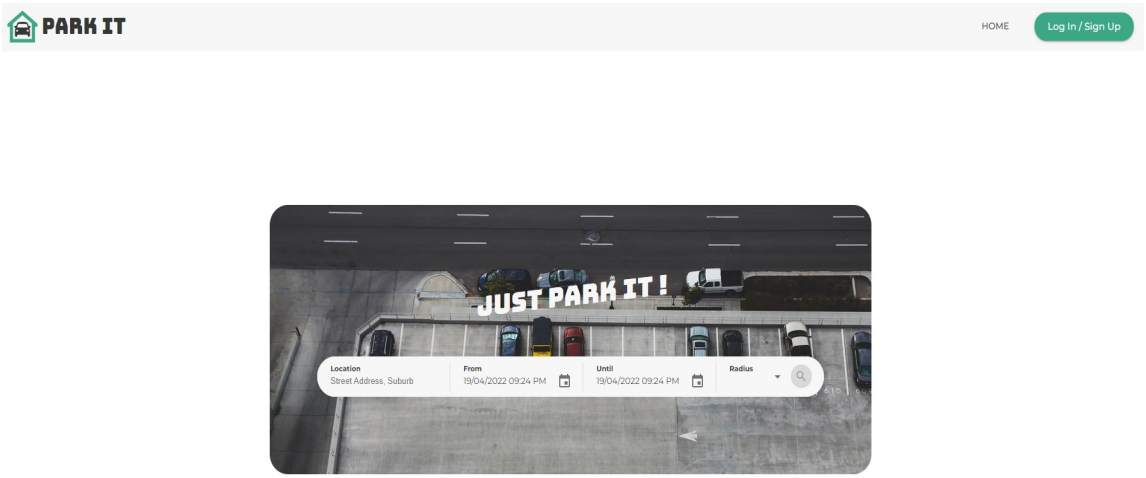


Diagram 1.4 - Home Page

Here we see a search bar that can be used by any non-logged-in/non-registered user to see what car parking spaces are available on our platform prior to committing to a full account. The search bar allows us to enter an address, potential booking start and end dates, and finally a radius around that address to look for available car spaces. If any of these fields are not fully filled in, the search will not proceed. When these fields are fully completed, it will redirect you to a page that will show all parking spaces available in the period you searched for, and within the radius you selected.

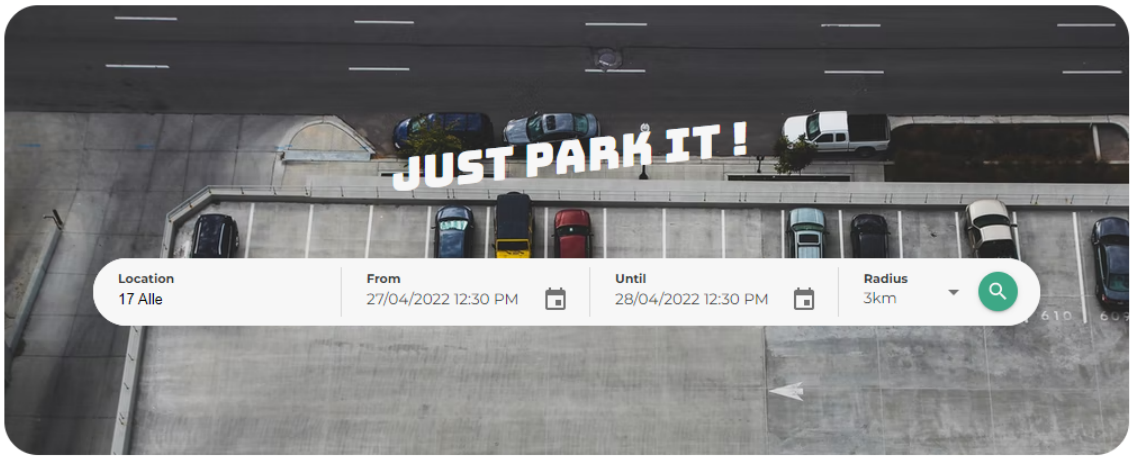


Diagram 1.5 - Home Page search bar in use

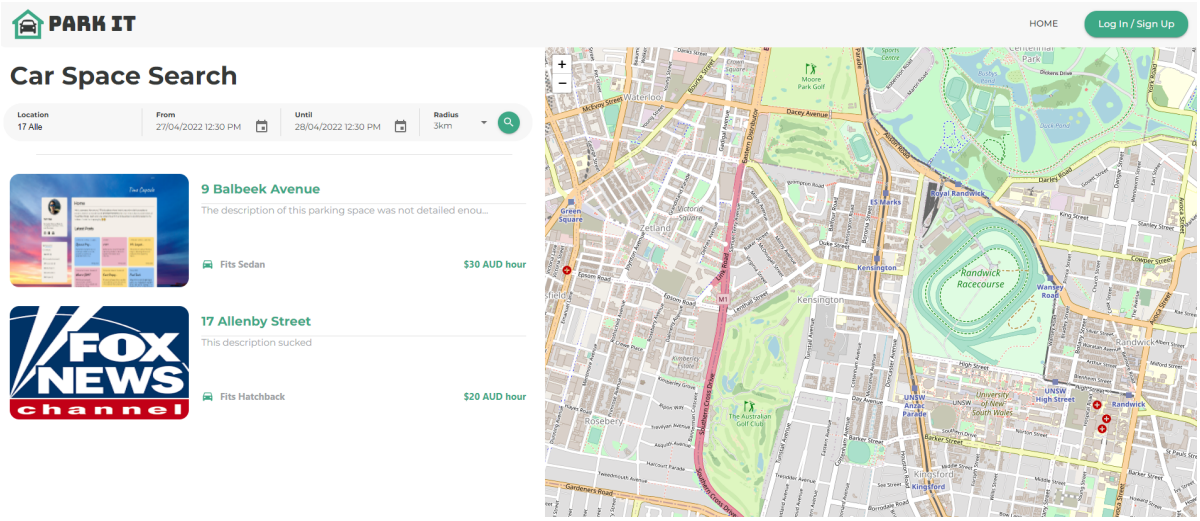


Diagram 1.6 - Results from Search whilst not being logged in

From the Home Page, we will see a login/sign up button that can be used by registered/non-registered users to login into their existing account or sign up for a new account. When clicked on, it will display our login/sign up modal as shown below.

From this modal, we are able to log in using our account credentials. All fields must be completed or an appropriate error message will be shown within the modal letting users know about the issue.

New users can register for a new account via the sign-up hyperlink within the modal. From here, a new user is able to register for a new account with their preferred username, password, email and phone number. All fields are required to be filled in with valid information for the registration form to be successfully submitted for the registration process. The form will check to see if a user's email has been used before. If so, an appropriate message will be shown to notify the user that the email address is already in use and that they can reset the password with the forgotten password option in the same menu. Otherwise, the form will be successfully accepted and the application will display a message indicating that a verification email has been sent to your email to verify its authenticity. The verification email will look like this:

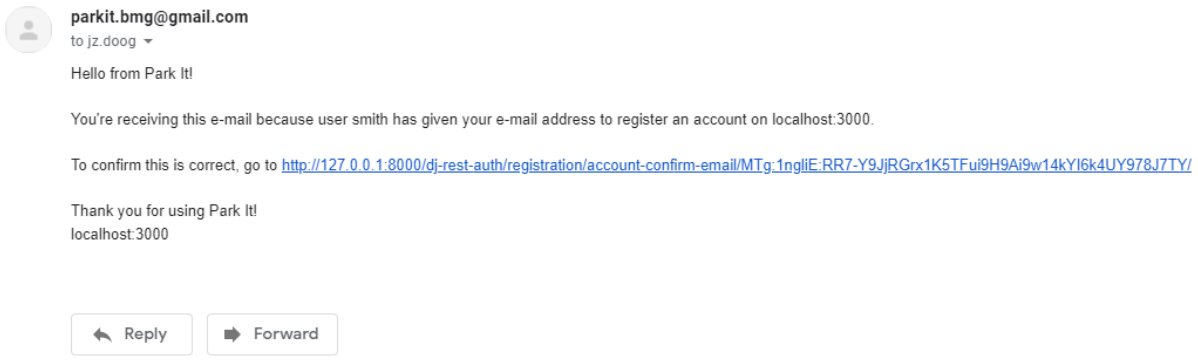


Diagram 1.7 - Example of the confirmation email sent after signing up

Once the link in the email has been clicked, it will redirect you to a new tab of the same website, except that we can now login using our new account details.

To reset your password, simply select the Forgot Password link in the login modal, which will accept an email address and send a password reset email to that address if it exists in the database. Clicking on the confirmation email will display a form to reset your password.

NOTE: All email links must be clicked in the Lubuntu environment so the backend receives the confirmation.

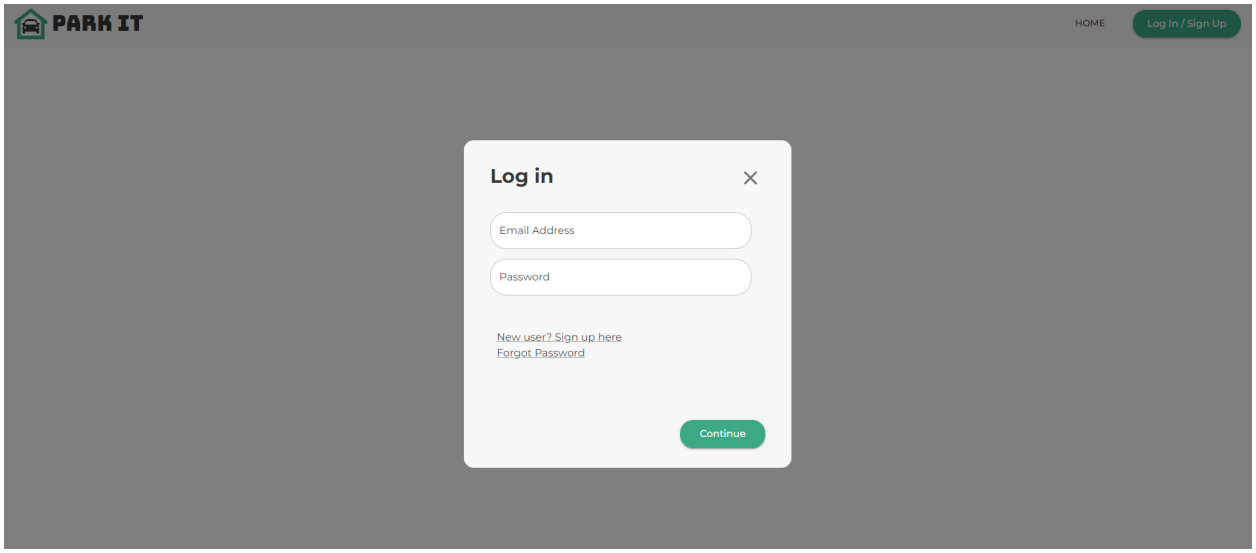


Diagram 1.8 - Log in modal

1.4.2 How To: Consumer

Setting up as a Consumer

Once logged in you should be presented with the homepage, except with the login/sign up button replaced with a new suite of options to choose from. Basically, any account on ParkIt is considered as both a consumer and a provider, meaning that you can both rent car spaces as well as list some of your own ones on the same account. This allows users to both search for car spaces to rent and check on those that they have on offer on the platform.

However, this part of the tutorial will focus on setting up our accounts as a Consumer. To do so, we need to head on to the accounts tab denoted by the green head icon on the far right of the screen.

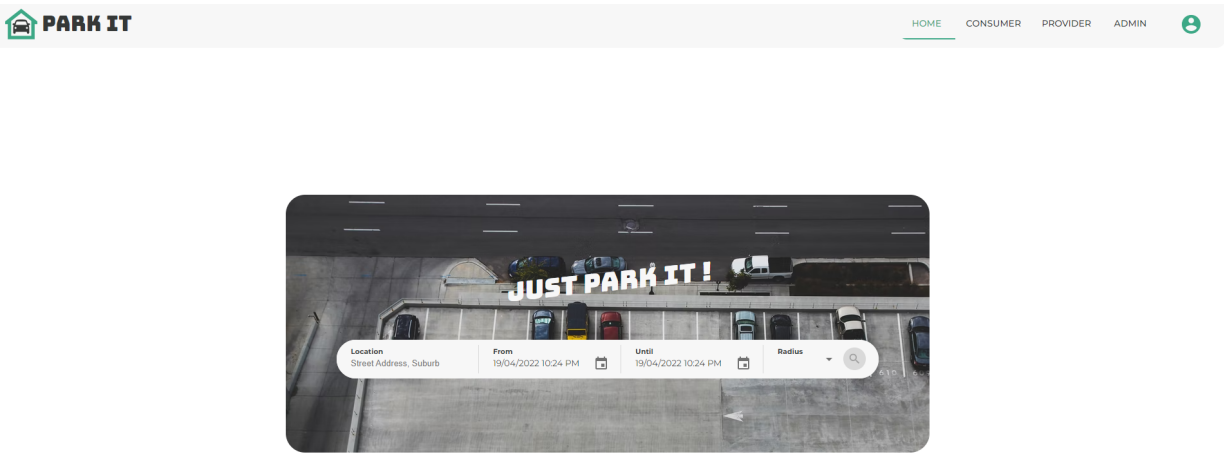


Diagram 2.1 - Home screen is shown when logged in

Account Details

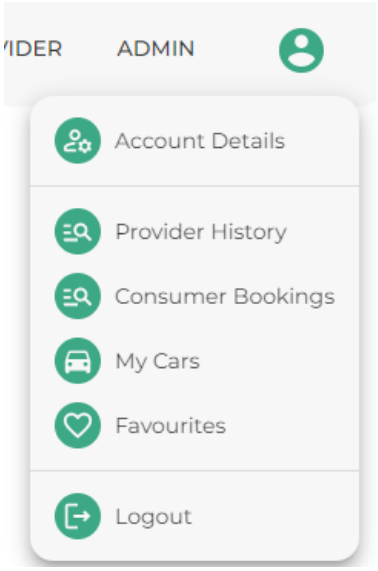


Diagram 2.2 - Account settings selection menu

Here we can see our dropdown menu upon clicking on the green head icon. The dropdown menu shows us what we can do with our account, such as accessing account details, as well as other options that we can use as either a consumer or provider. For setup flow purposes, we would proceed to the “Account Details” tab by simply clicking on the corresponding tab in the menu.

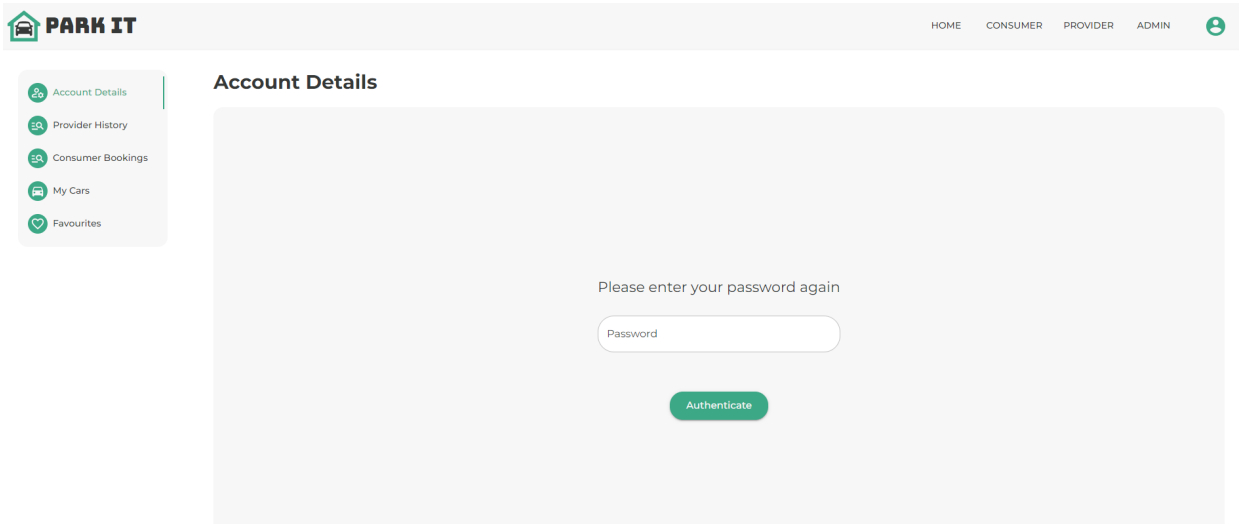


Diagram 2.3 - Account details before authenticating

Once you have clicked on the tab, you will be redirected to the page as shown above. We see that the main account details tab is locked off by a password page. This is done so that we can confirm your identity prior to revealing your personal details to the person accessing your account (which hopefully is you!). Therefore, we just need to re-enter our password into the input field, and we should be able to see our account details.

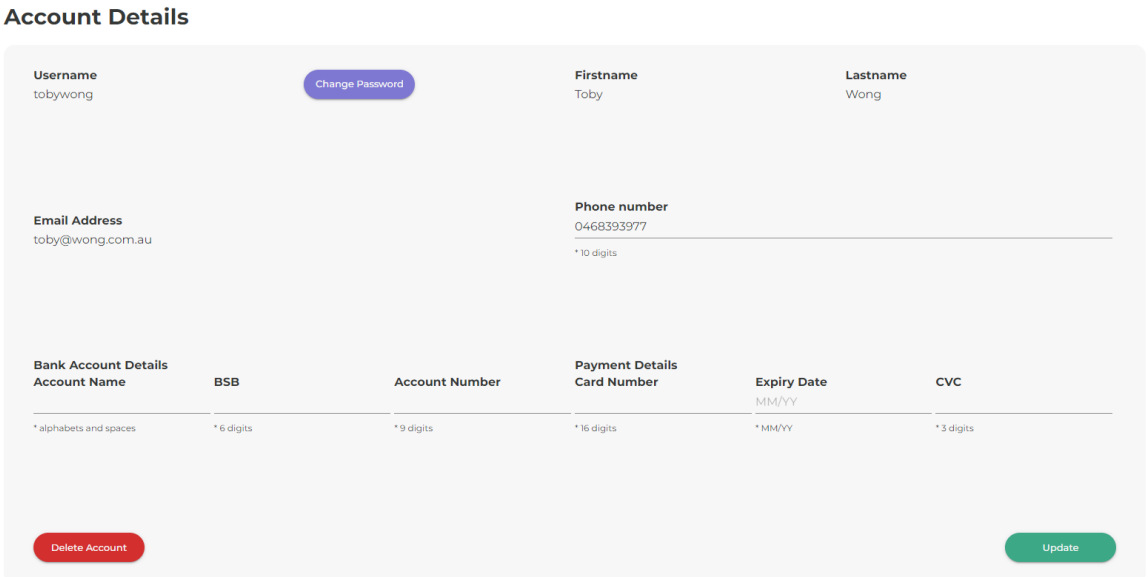


Diagram 2.4 - Account details after successful authentication

Over here we can finally see our Account details, such as name, phone number, email as well as Payment Details. For Payment Details, we have split up the section into 2 different sections, one called Bank Account Details and one called Payment Details. In short, you would only need Bank Account details if you are a provider, only need Payment Details if you are a consumer, and both if you are a client. For setup flow purposes, we would only key in our Payment details. Once we have fully keyed in our Payment details, we can save our changes by clicking on the “Update” button. Once clicked on, the details are saved and we can move on to the next stage of the setup.

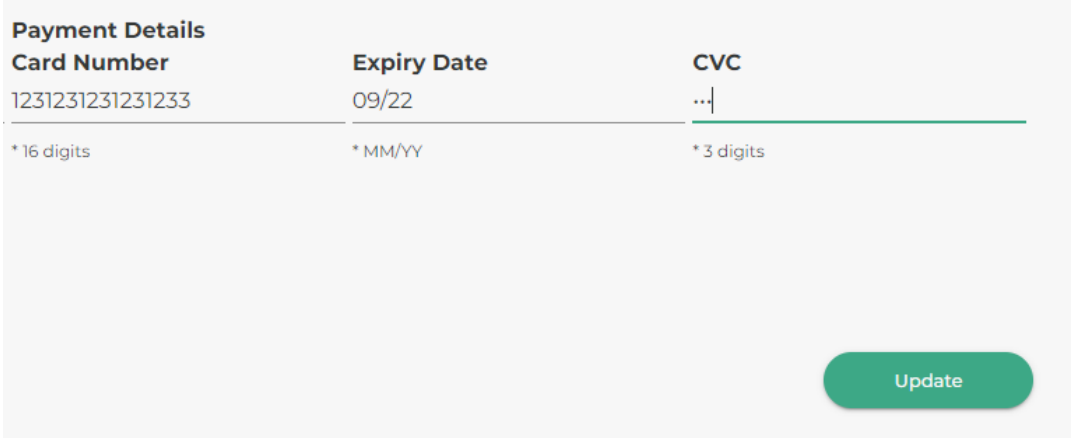


Diagram 2.5 - Adding card payment details

My Cars

Now that we have our payment details stored, we can now register a car as a consumer. Each consumer can have one or multiple cars under their account. Since we are a new user, let us add our imaginary car.

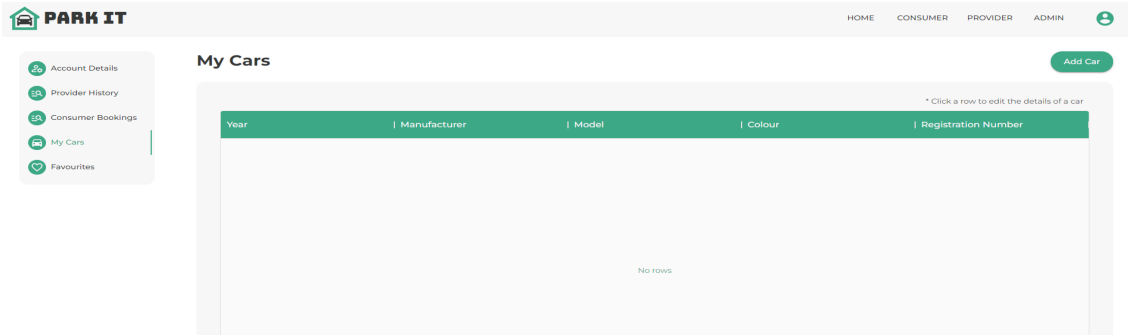


Diagram 2.6 - List of cars added to user

We click on the appropriately named “Add Car” Button on the top right of the page followed by Adding our car’s details into the form, as shown below

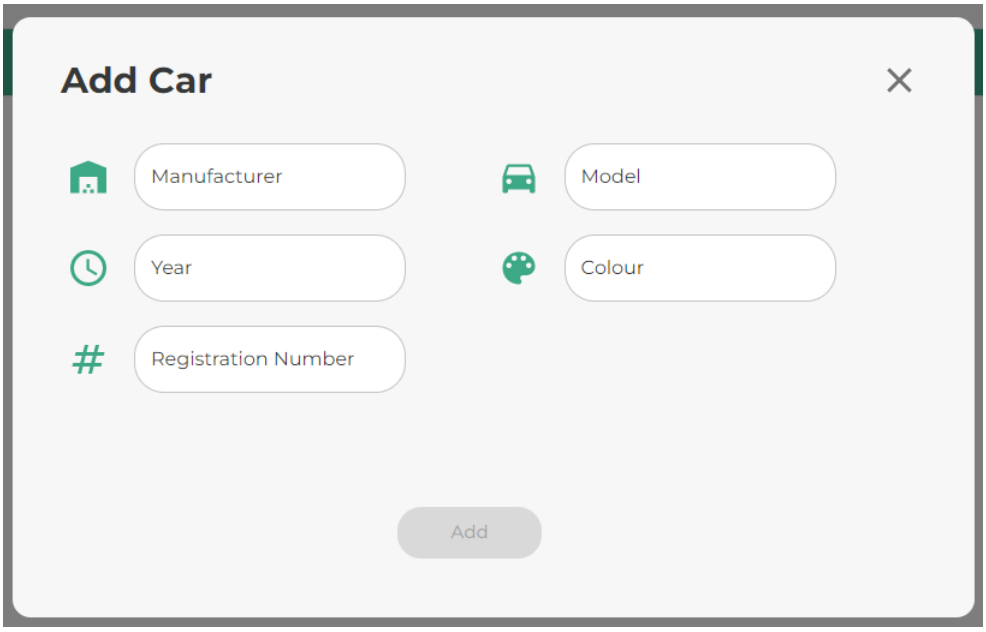


Diagram 2.7 - Adding car modal

Once that is done, we should now see our car listed on the My Cars table on the page. Now that we have both our car registered, and our payment details in place, we are ready to book a space as a consumer.

Making A Booking

We are now ready to make a booking as a Consumer.
First, we head back over to the Consumers tab, located on the top right of the page.

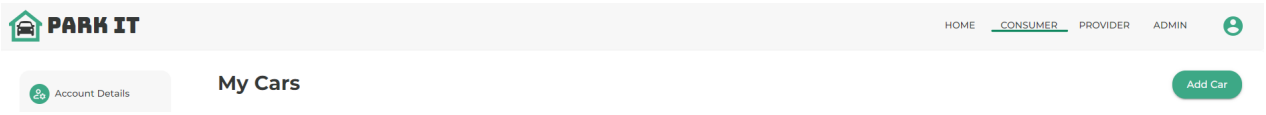


Diagram 2.8 - Highlighted consumer tab

Once we are back on the consumer’s page, we can begin searching for a car parking space.
To do so, we can utilise the search bar on the left-hand side of the page.

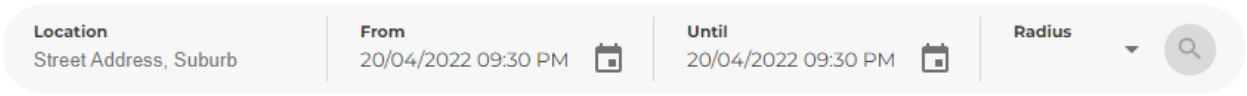


Diagram 2.9 - Search bar

To use the page, we need to put in a location that we would want to search from, our parking duration (start and end time of our intended booking), and the search radius from the location that we have placed. We need to ensure that all fields are filled before the search bar will be toggled for searching.

Once we have searched for a location, it will return all results that satisfy our search criteria. Here we see that we have inputted an example search query and that the results are displayed in the once empty void that was underneath the search bar.

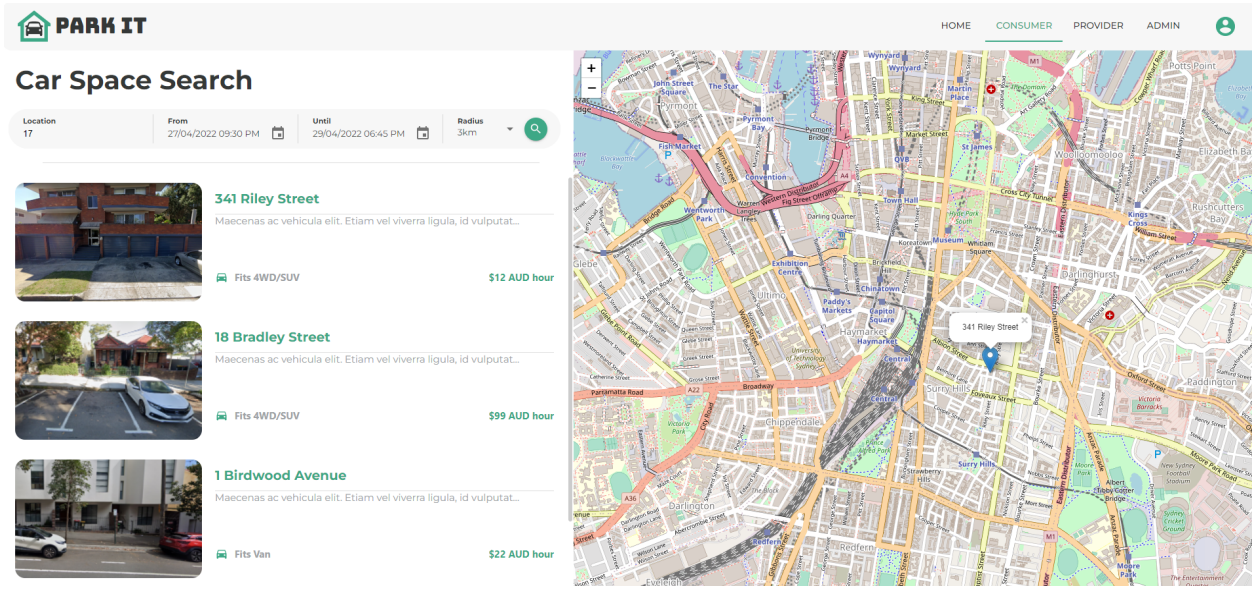


Diagram 2.10 - Search results displayed with Map View

The list on the left shows us the car space listing that can accommodate our booking criteria (location, start and end time), while the map on the right shows us a better representation as to where our car space is rather than just showing a plain text address.

To see where each listing is on the map, we can do either one of two actions:

1. Clicking on the listing on the left-hand side, would then open the car space listing AND move the map to where the point is.
2. Moving the map on the right-hand side, and clicking on the blue points on the map to view which point corresponds to which listing in the search results,

Either way, both methods should allow you to get a sense of where the car space is.

Once you have selected your preferred parking space from either of the 2 methods above, you can view more about the car space by clicking on the car space listing. It should look something like this:

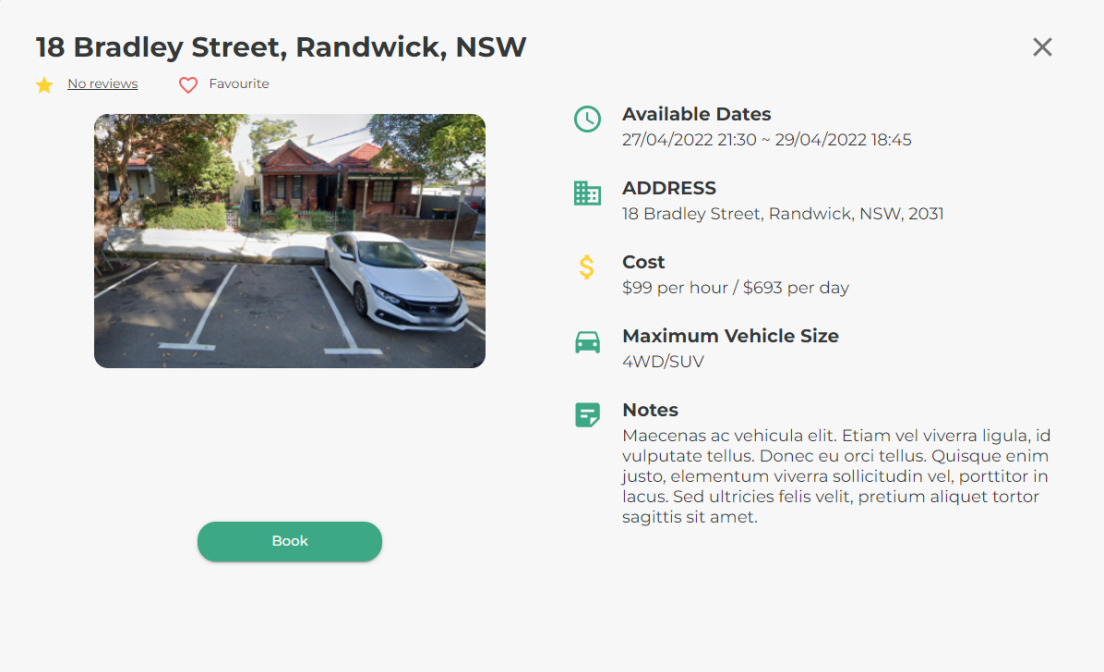


Diagram 2.11 - Parking space information

In the listing, we can see further details of the listing, such as the address of the car space again, hourly/daily cost of the car space, maximum vehicle size that this ar space can accommodate, and some notes and images that allow consumers to further visualise and understand the space that they are looking at.

If you want to see the reviews on the booking, you can do so by clicking on the Reviews button on the top left corner of the listing, right next to the logo of the stars. There, you would be able to see all reviews left by previous users of the car space.

If you wish to keep this listing for future use, you can do so by adding it to your Favourites by clicking on the heart icon. It would be stored in the Favourites list which is mentioned in Managing Bookings - Favourites, Cancellation and Review down below

However if this space is the one for you, you can proceed with booking by clicking on the “Book” button in the listing, and you’ll be redirected to the booking form, as shown below.

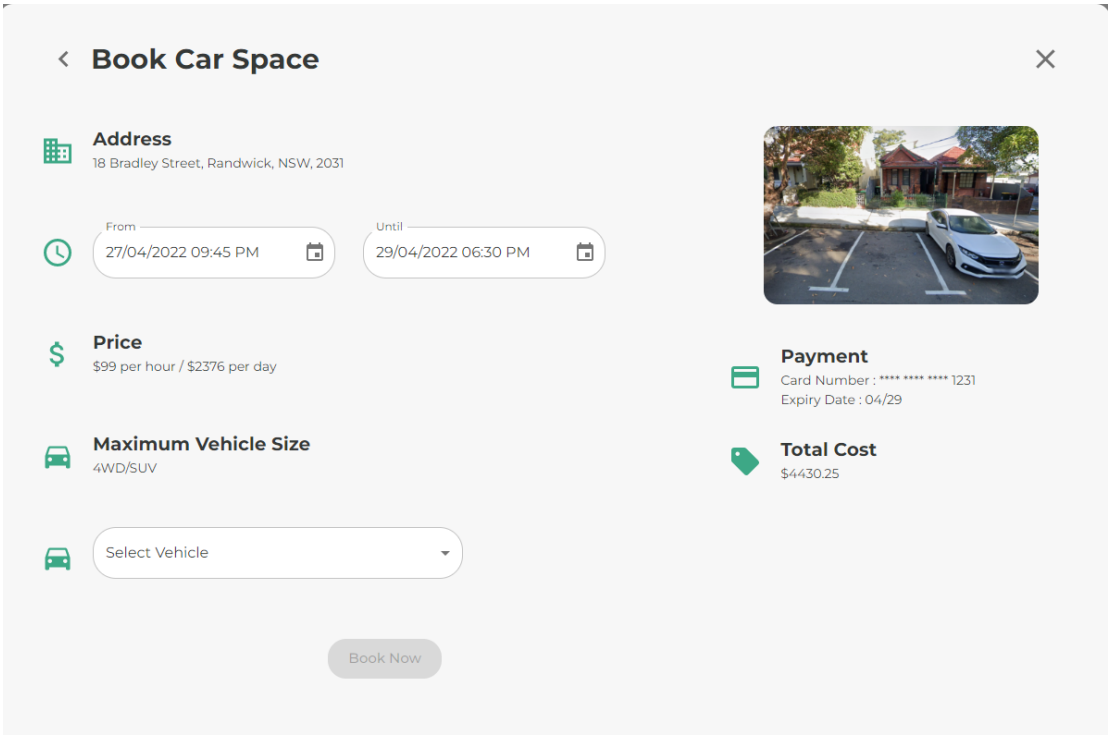


Diagram 2.12 - Car Space Booking Modal

Once here, we should reconfirm our booking start and end date, agree to the total cost charged, and add the car that we want to use in the car space before finally placing a booking on the space. Once we have booked our space, there would be a confirmation message like the one shown below, letting us know that we cannot cancel bookings 7

days before the booking is meant to start. Once we accept the confirmation, only then would our booking be placed

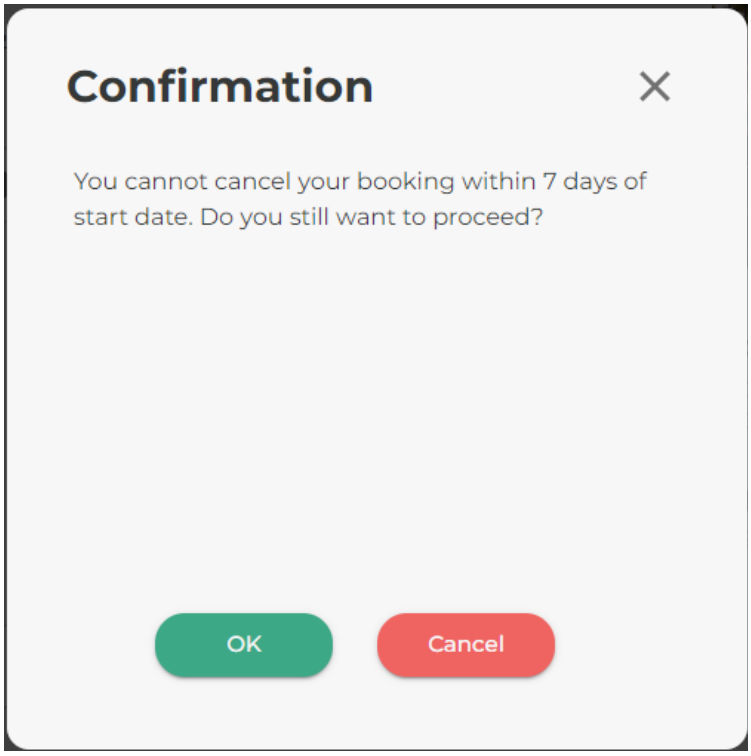


Diagram 2.13 - Booking confirmation modal

If no results are returned, it could mean that there is no available parking space in the area you are searching for, or that there are no available slots for a booking to be made within the car spaces that are within your search location.

Managing Bookings - Favourites, Cancellation and Review

Once we have made a booking, we can manage our bookings by heading to the top right corner, clicking on the Accounts Tab and selecting the Consumer Bookings Tab here

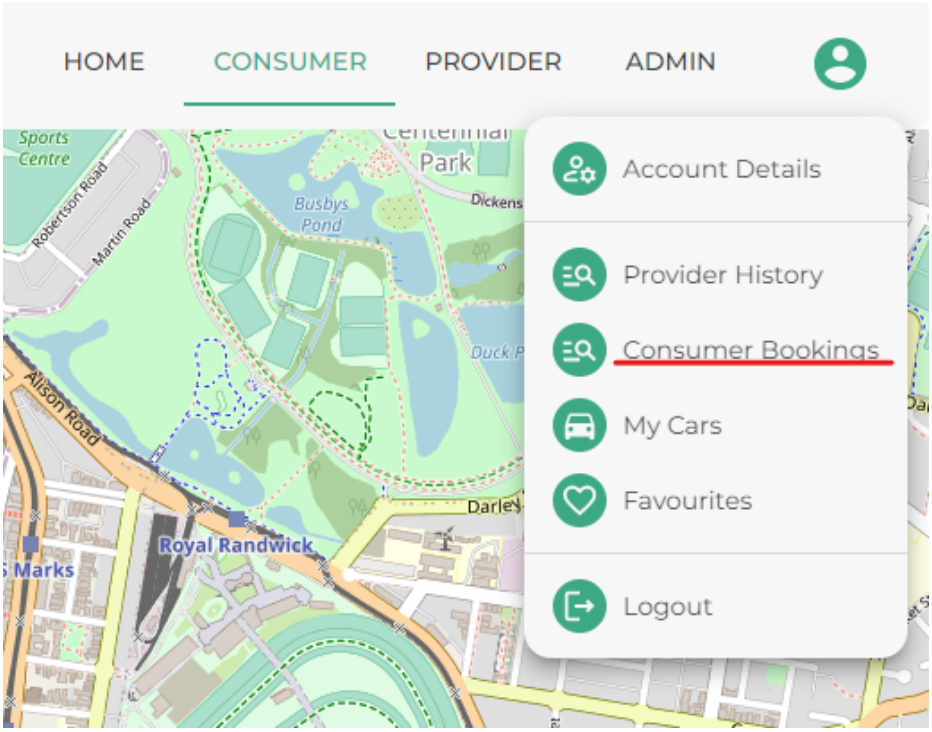


Diagram 2.14 - Consumer bookings option under user menu

We would then be redirected to the Consumer Booking page, which would look something like this

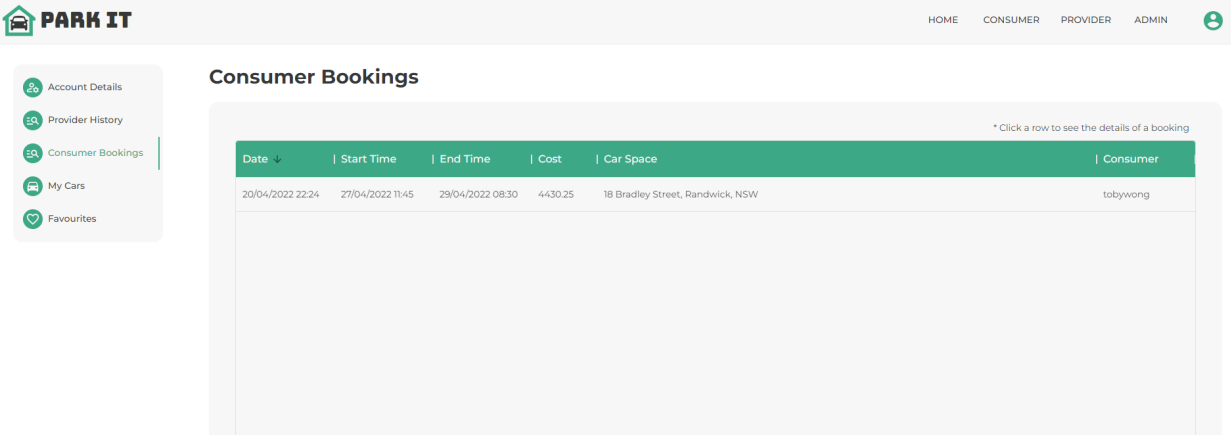


Diagram 2.15 - Consumer bookings

We can see that our latest booking has shown up here. If we click on it, we can see the booking receipt that shows us all the details of the booking we just made. Down at the

bottom, we see options for us to write a review for the booking, and also a button to cancel the booking. These buttons allow themselves to be used based on the time that we access the booking. In short, we can only leave a review after the booking end date. As for the delete button, we can only delete a booking 7 days before the booking start date.

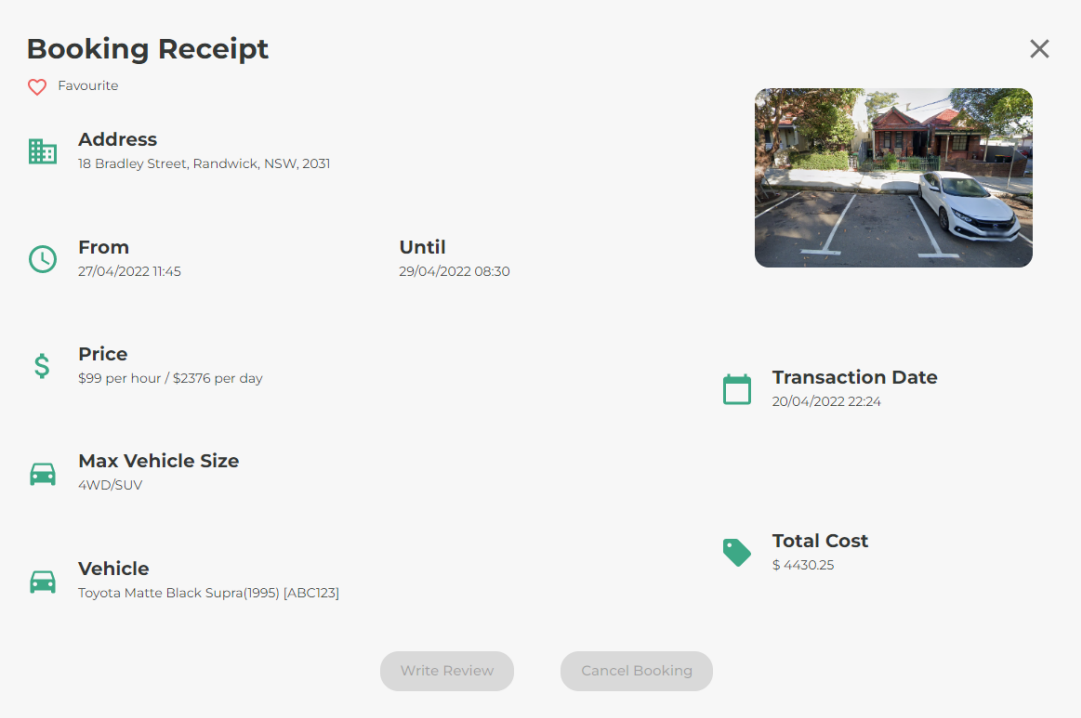


Diagram 2.16 - Booking receipt

From the same page as before, we can also view all of the listings that we have favourited by clicking on the Favourites tab shown in Diagram 2.15.

1.4.3 How To: Provider

Setting up as a Provider

Setting up our account is similar to the process of setting up our account as a consumer, except we now need to fill up our Banking Details instead of our Payment

Details in our account. To do so, we first click on the head icon on the top right of the page.

Account Details

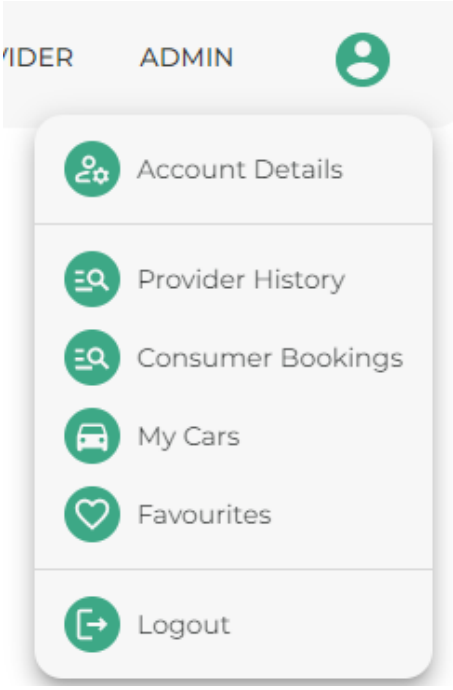


Diagram 3.1 - User settings menu

Here we can see our dropdown menu upon clicking on the green head icon. Then, we click on the Account Details section which would bring us back to this familiar screen below.

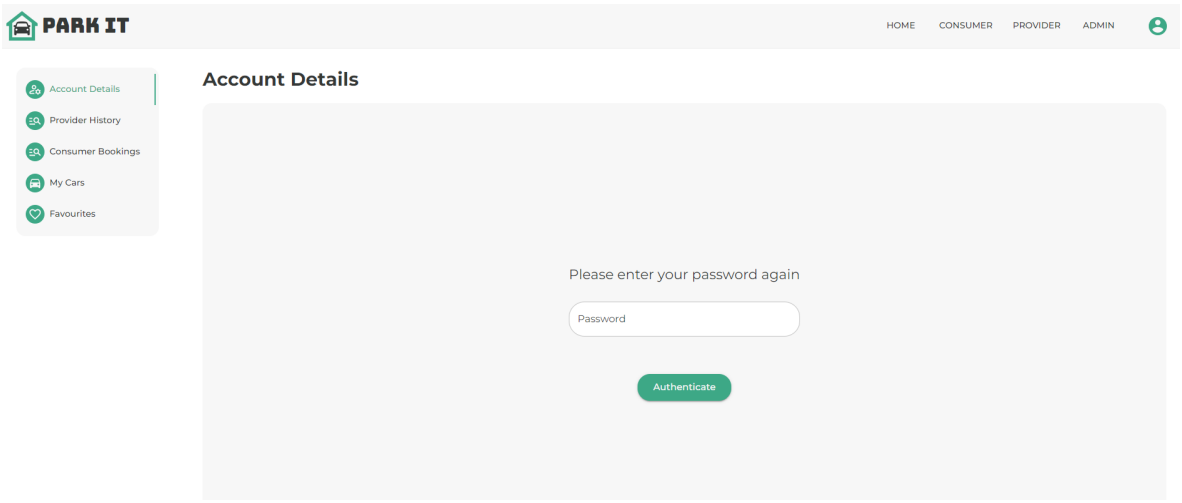


Diagram 3.2 - Account details before user authentication

Once here, all we need to do is to re-input our password into the input field, and we should be able to see our account details now, as shown below.

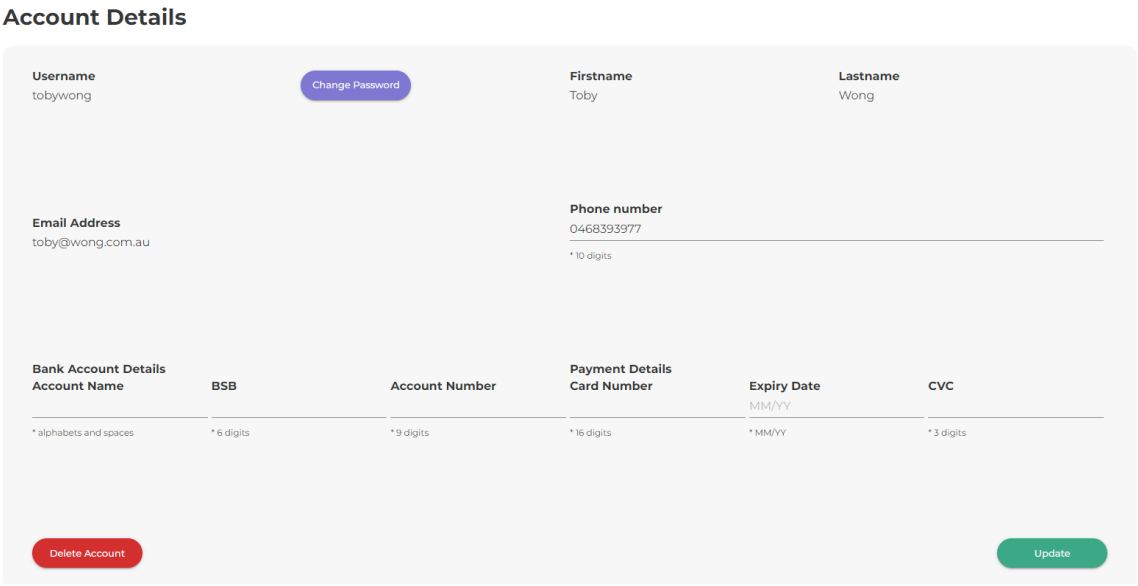


Diagram 3.3 - User account details

Instead of filling up our Payment Details, we would instead fill in our Bank Account Details section on the bottom right of the account details page. Once we have fully keyed in our Bank Account details, we can save our changes by clicking on the

“Update” button. Once clicked on, the details are saved and we can move on to the next stage of the setup.

Adding a Space in Provider

Now that we have our Banking Details in our account, we can begin providing/advertising car spaces from our account. To do so, click on the Providers tab on the top right corner of the screen. It should look something like this

Diagram 3.4 - Results from Search whilst not being logged in

From here, we would be redirected to the Providers page which would look something like this

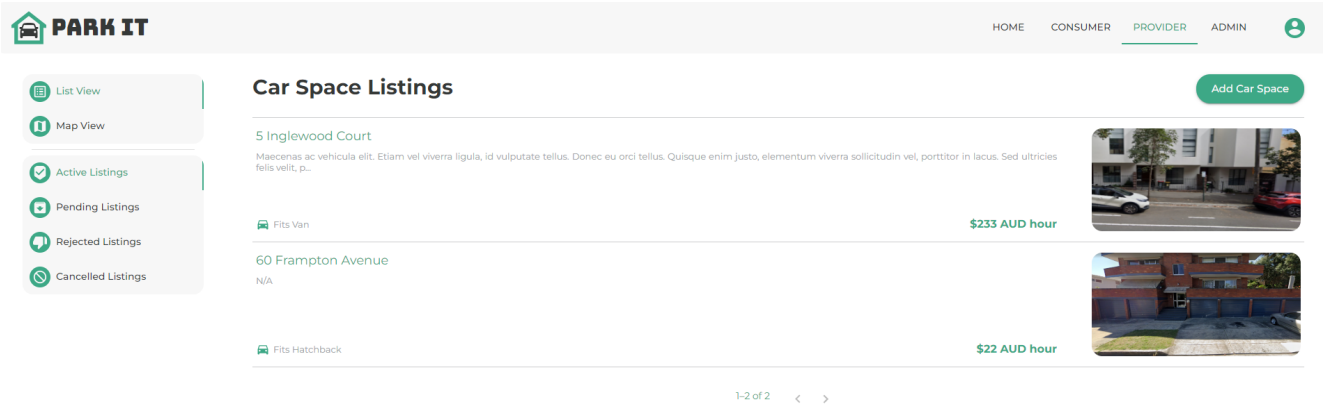


Diagram 3.5 - Current car space listings belonging to the user

To add a car space, we would click on the Add Car Space button on the top right, followed by filling in the Car Space Registration form as shown below.

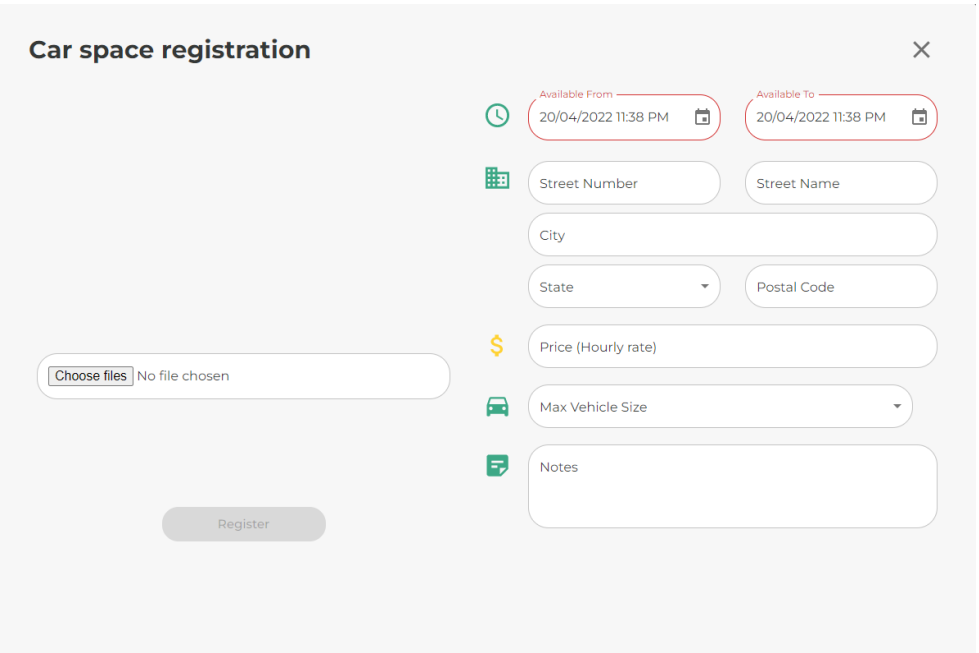


Diagram 3.6 - Car space registration modal

Once that is done and we have filled in all the details we need, we can submit the form, which would bring up a notice letting us know that as providers, we would have to fulfil all bookings that have been made even though our spot may be cancelled in the future.

Once that is done, we can find our newly registered car space located in the pending listings tab on the right side of Diagram 3.5. This is because all newly registered car spaces must be reviewed and approved by BMG staff members.

View, Edit and Delete a Car Space

Once a car space has been approved by BMG staff, it will now show up in the Active Listings page by clicking on the Active Listings tab, while Rejected Listings will show up in the Rejected Listings page by clicking on the Rejected Listings tab.

A provider can view their listing in more detail by clicking on the listing in the Active Listings page.

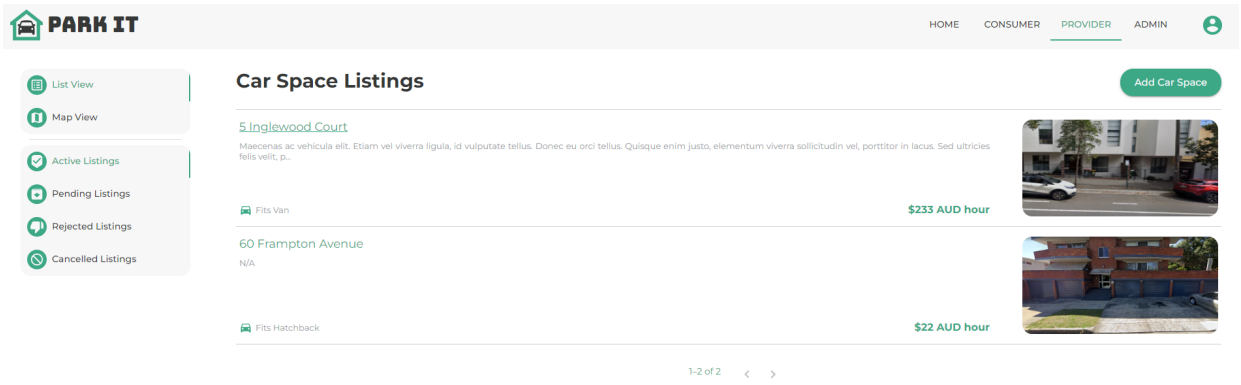


Diagram 3.7 - Car space listings belonging to user

A modal would show up just like the one below

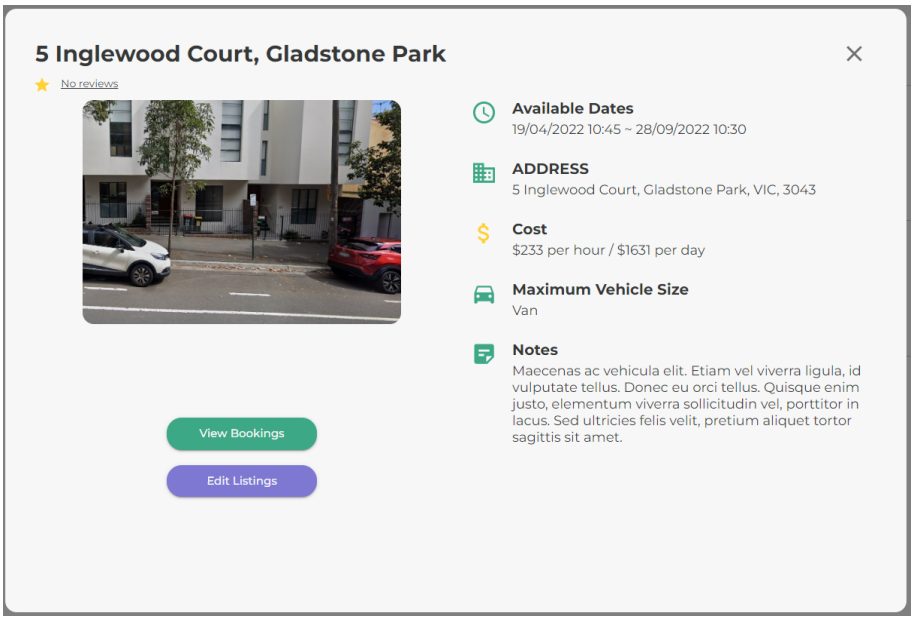


Diagram 3.8 - Parking space details

In this modal, we are able to either view all bookings made under that space by clicking on the View Bookings button, while we can edit/delete the listing by clicking on the Edit Listings button.

Here we can see that the modal changes to an editable form, whereby we can change the fields that were in our listing, such as the one below.

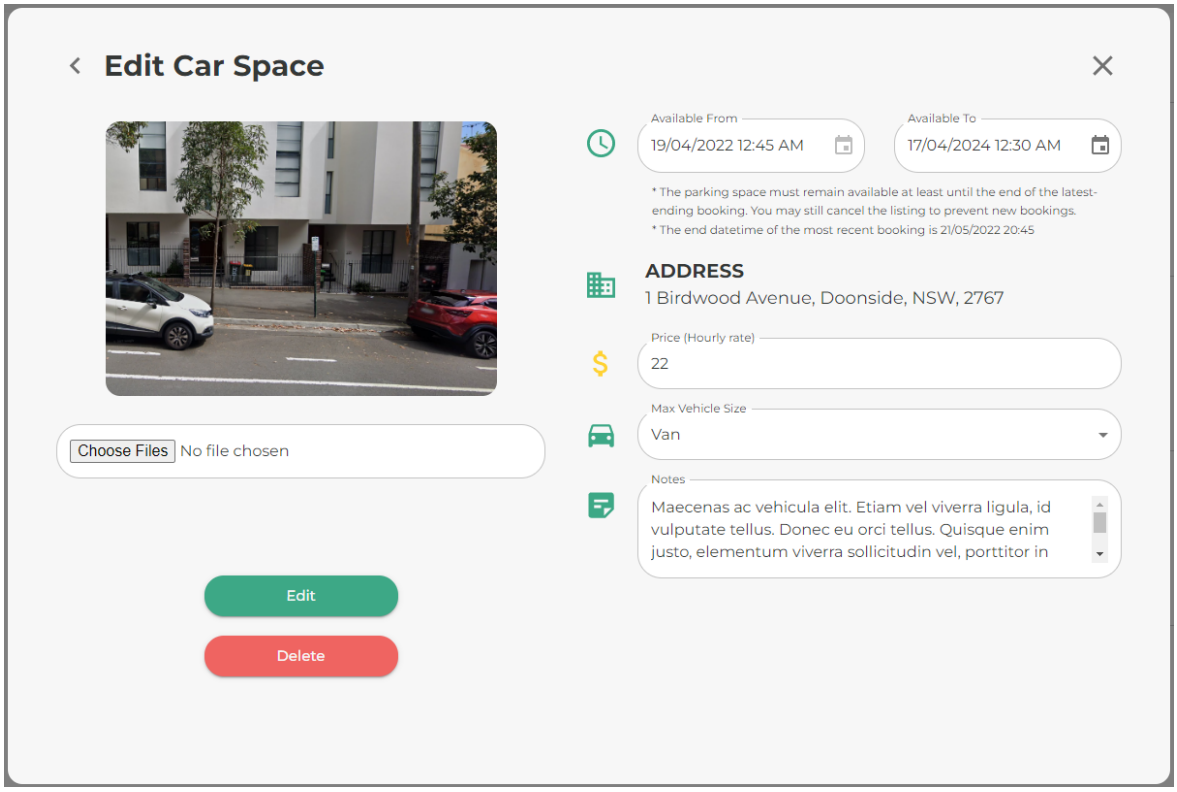


Diagram 3.9 - Editing car space listing details

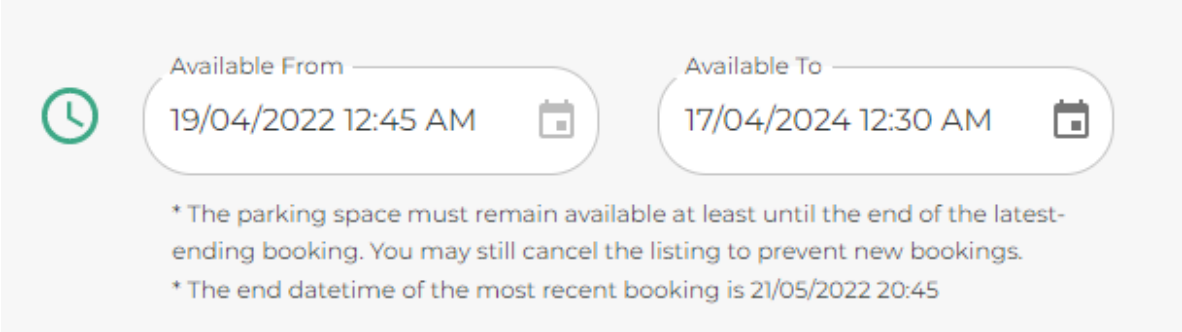


Diagram 3.10 - Footnote preventing the user from setting the availability end date before the end of the last booking in that space

Here we can see that the last booking end date for the space is 21/05/2022 at 20:45. Since a provider should fulfil all the booking which has already been made from the customers, the end date cannot be edited to earlier than 21/05/2022 20:45. And you can check it in the image below.

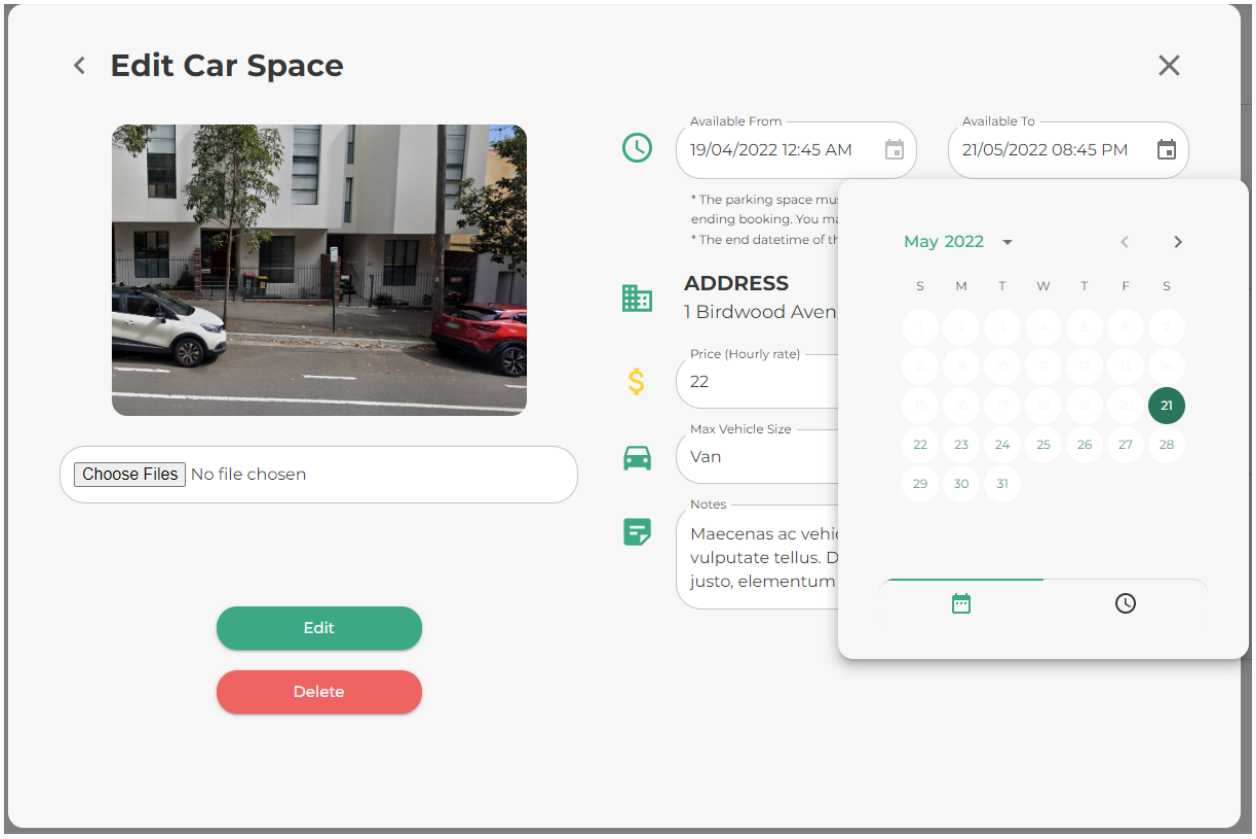


Diagram 3.11 - Date selection modal shown whilst editing car space details

1.4.4 How To: Admin

Setting up as an Admin

To login as an Admin, we recommend using the provided Admin account details located in Account Details. Using those account details, you can then login on to the admin page that looks something like this:

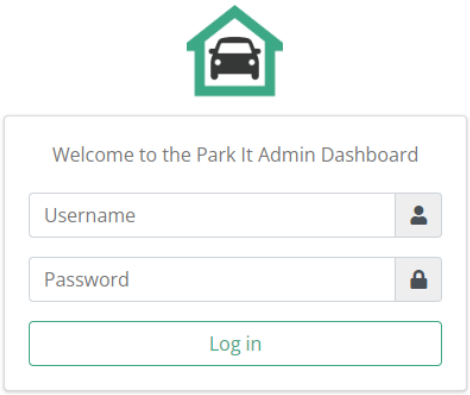


Diagram 4.1 - Admin Landing Page

Once logged in, the admin panel would look something like the image below. Here, you can access all the different settings that an admin would use. However, the main key parts that we would use are the Users tab, located at the bottom left of the page.

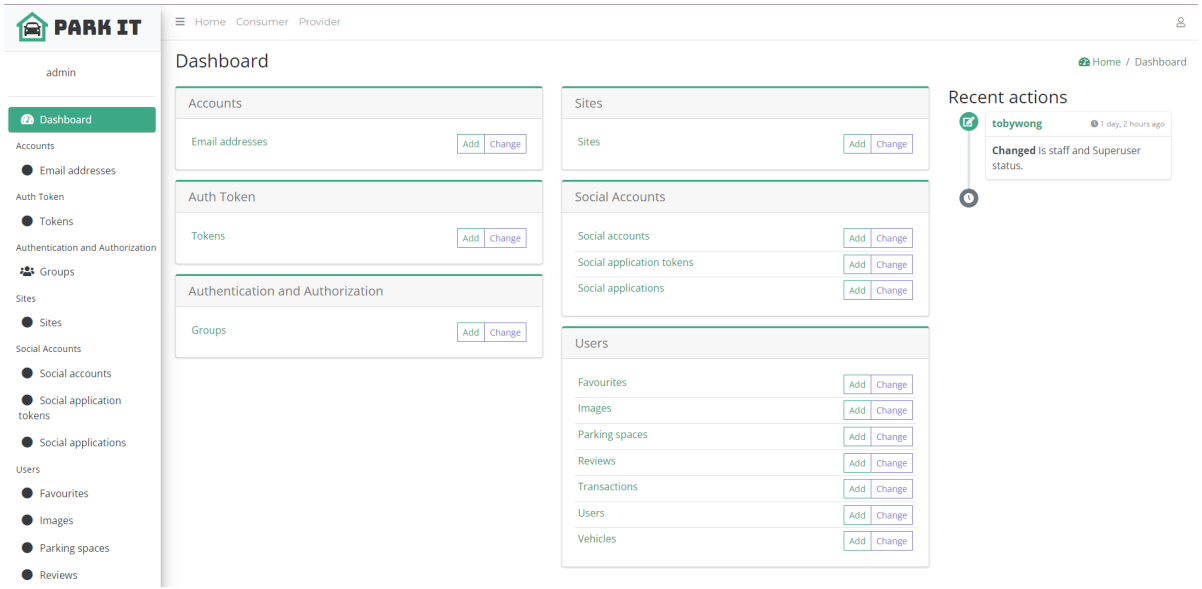


Diagram 4.2 - Admin page after logging in

Add someone to the staff account

To add a new account to be an admin/staff account, we first need the user to register for an account like normal. This can be done by following the instructions shown in Registering and Logging In. Once you have made an account, notify an admin (which in this is us) to convert an account into an admin account. (For admin account details please see backend set-up instructions at the top of this document).

For admins, to convert an account, go to the bottom left of the page where you would find the Users tab menu. Once there, click on the Users tab under the Users menu. An example is shown here

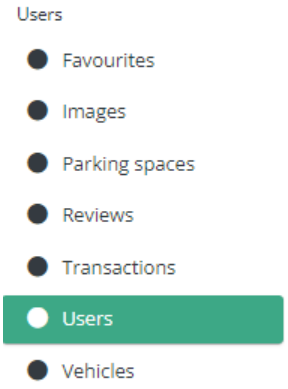


Diagram 4.3 - Users tab under Users menu

Once here, we should see a page that contains all the users on the platform

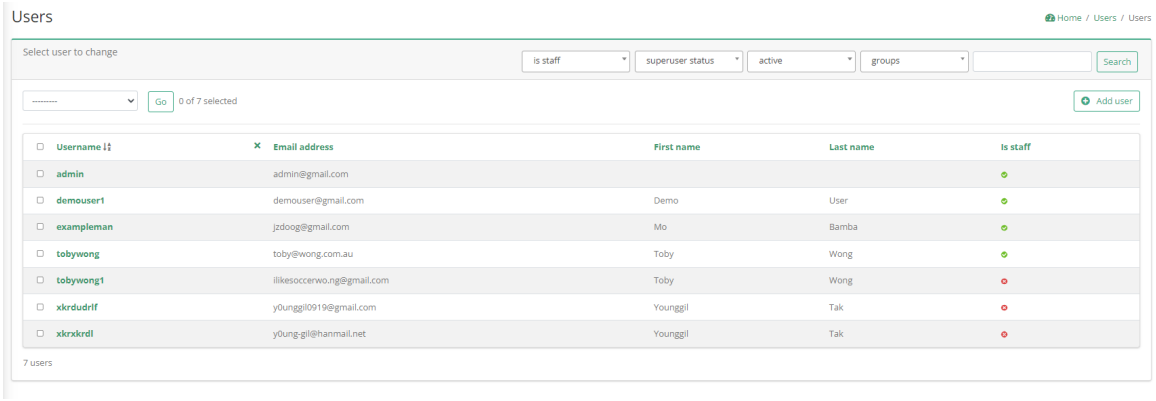


Diagram 4.4 - Users page showing all users

Once here, we can select the User-based on their email on the page, and then check the is_staff flag under the Permissions page, as well as selecting the Staff group under the “Change Groups” menu. An example is shown below.

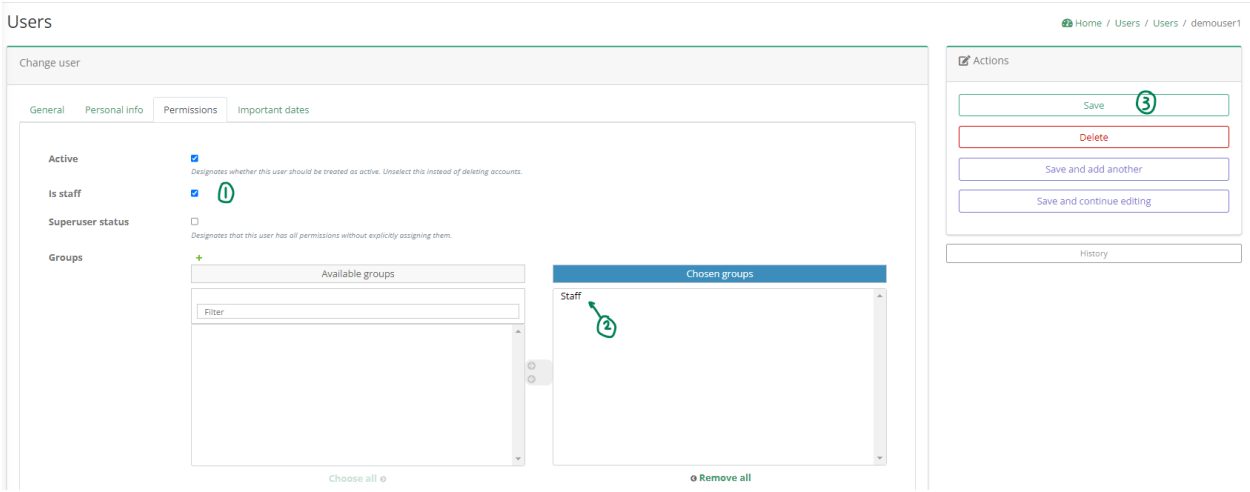


Diagram 4.5 - Permission Tab to toggle is_staff

After that, the person should have staff privileges on their account. This can be seen by the new Admin's tab whenever they log in like a normal user. An example is shown here:

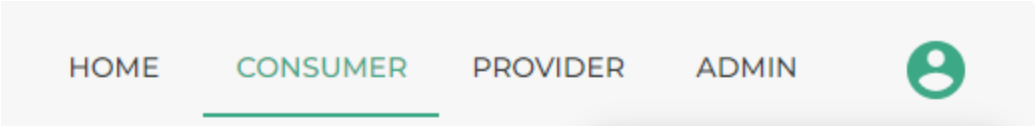


Diagram 4.6 - New Admin's tab after logging in

Approving Car Spaces

We would now focus on one of the staff's responsibilities on this admin platform, and that is to approve new car space registrations. To do so, we first go to the Parking Spaces tab within the User menu as shown below

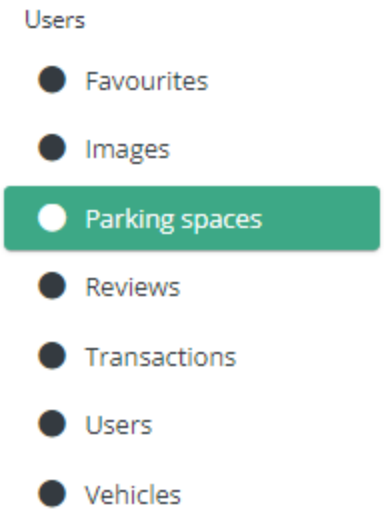


Diagram 5.1 - Parking Spaces tab under the Users menu

Then, we land on this page which shows us all parking spaces on the website.

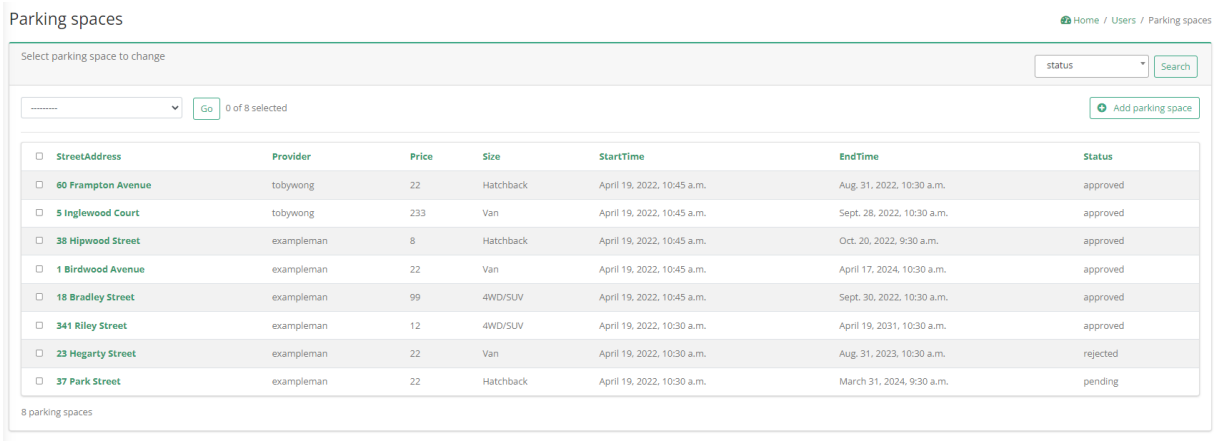


Diagram 5.2 - Parking Spaces page

To make things easier to navigate, we can filter our car parking spaces into their different statuses (eg. Active, Pending, Rejected, Cancelled) via the dropdown menu on the top right of Diagram 5.2. For our case, we would filter based on Pending car spaces.

To filter our car parking spaces into its different statuses (eg. Active, Pending, Rejected, Cancelled), we can use the dropdown menu on the top to filter. Filtered parking spaces should look something like this

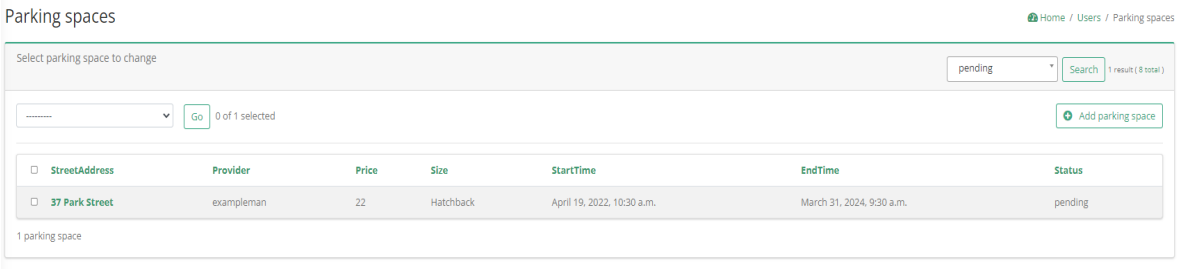


Diagram 5.3 - Pending parking spaces

Now that we have our Pending parking space, we can select it to view its contents. Once we feel that we are ready to change its status, we can do so by selecting a status in the dropdown menu, and clicking on save.

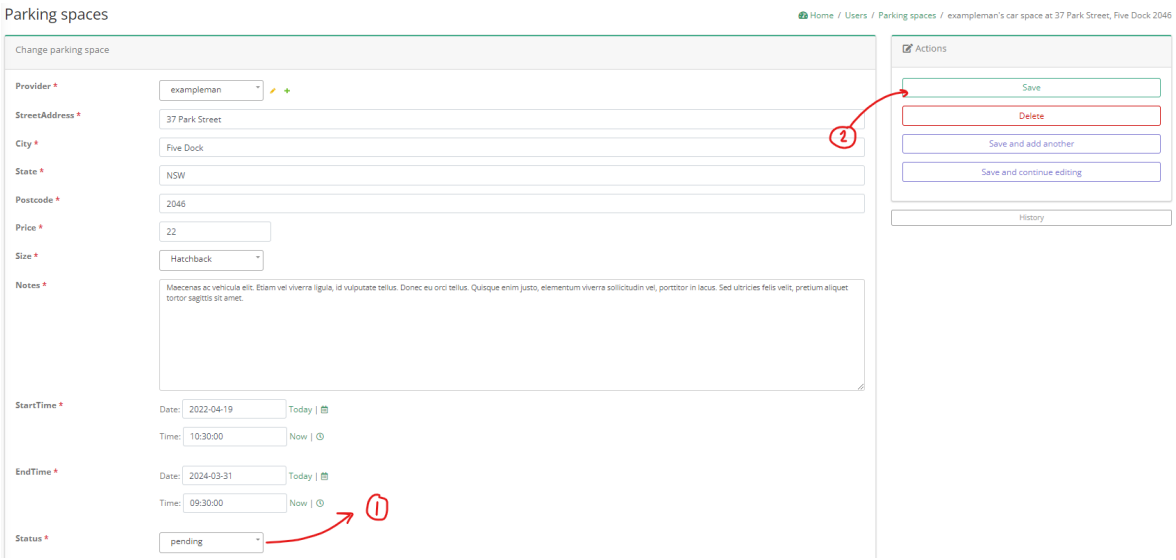


Diagram 5.4 - How-to change the status for a parking space and save changes

2 Overview

2.1 System Architecture

Park It is a car space sharing system. It is a web-application built on a stack including a React frontend and a Django backend, connected through a REST API powered by

Django REST Framework. (Facebook, 2022; Django Software Foundation, 2022; Encode OSS, 2021). Data is stored in an SQLite3 database (SQLite Consortium, 2022). The goal of the application is to provide an interface for the offering and renting of car spaces, and associated functionality such as authentication, reviews, favouriting, searching and platform administration.

The technical architecture can be split into three layers - the presentation layer, the business layer and the data layer. Each will be described in detail below.

2.2 The presentation layer

The presentation layer of the application consists of the front-end which has been built using the Javascript framework React and written in HTML (Facebook, 2022). The application has been styled using CSS and Material UI to maintain a consistent look and feel across the platform (MUI, 2022).

2.3 The business layer

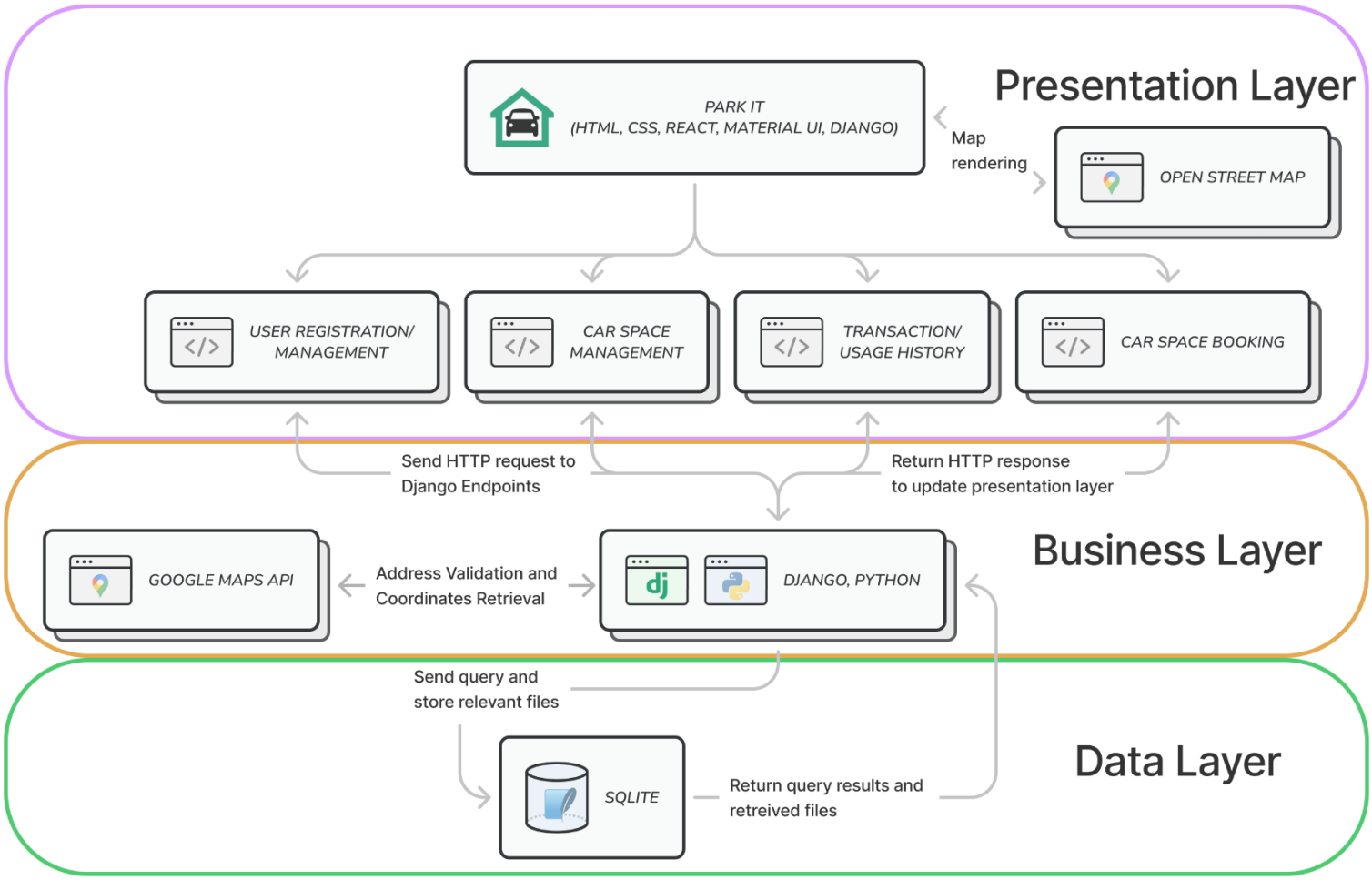
The business layer of the application maintains the business logic and is responsible for the validation of incoming inputs, and any manipulation that needs to be done in order for the core functionalities of the Park It platform to be carried out. The backend is written in Python, leveraging the Django web framework and the Django REST Framework to create a REST API in order to provide a modular, scalable and clean interface with the frontend (Django Software Foundation, 2022; Encode OSS, 2021). The business layer also interacts with GMail as our email server and Google Maps/OpenStreetMap to serve interactive maps that identify the geolocation of all registered parking spaces (Google, 2022; OpenStreetMap Foundation, 2022).

2.4 The data layer

The data layer of our application takes the form of an SQLite3 database that is able to handle and process queries efficiently (SQLite Consortium, 2022). It stores the user

data, parking space data, booking data and much more. It is seamlessly integrated with the Django backend (Django Software Foundation, 2022).

System architecture diagram



3 Functionalities of Application

3.1 Core functionalities

Park It has eight core functionalities that make up the car space sharing platform, and together provide a holistic and compelling experience which differentiates Park It from competing for car space sharing platforms.

3.2 Account registration and management

Park It allows users to register and manage their accounts to ensure the persistence of important data, including registration of payment details, car details, booking history and keep track of any car spaces currently or previously being listed. Park, It adheres to industry standards for security and authentication, making use of JWT-based authentication for ensuring all login information and any associated account information is safely transmitted. On registration, an email confirmation is required to ensure that the account is valid. There is also a ‘Forgot your password’ feature which provides a password reset link to the user’s email address in case they have forgotten their password. Certain account details are able to be changed, but require re-authentication for security purposes.

A unique aspect of the account management system is the hybridised design where users are able to both search and book car spaces, while also listing their own car spaces for rent. Put simply, the user is able to act as both a consumer and a provider. The hybridisation means that only one account and set of login information is needed, saving the end-user the trouble of having to maintain two separate accounts to use the full potential of Park It.

3.3 Transaction and system usage history

Park It allows users to monitor their usage history both from the Provider and Consumer point-of-view. They are able to view past bookings and obtain details such as when the booking was made, between what times and how much was paid among other details.

While Consumer History allows users to keep track of their purchases, Provider History is particularly useful as it allows the provider to monitor changes in length and frequency of bookings for the particular parking space. With this information, they are able to adjust the pricing of their space to attract more customers to their listing.

3.4 Car space searching

Park It makes finding car spaces to be rented easy with the search feature which allows consumers to find parking spaces near their desired location. The search system uses Google Maps to identify the geolocation of the desired spot, and utilises a bounding box to efficiently select parking spaces within a geographical distance from that desired spot.

Filters are included with the search feature to narrow down results based on criteria including desired dates and desired distance away from the given address. The search only shows available car spaces that are not booked throughout the period between the given dates.

3.5 Inspecting and booking car spaces

Consumers are able to view the parking spaces after searching for them and determine whether they are interested in booking the parking space based on the images provided or any of the details in the listing such as price, size or availability.

After selecting an appropriate parking space, consumers must register their vehicle in the Park It system, and then will be able to undertake the booking process. Once the duration of the booking and relevant dates are confirmed, the total cost will be calculated and upon consumer approval, the booking will be entered into the system. Consumers may cancel bookings up until one week prior to the start of the booking. As part of the platform policy, consumers are ensured that upon successfully making a booking, they will be guaranteed that spot and timeslot by the provider.

A consumer may cancel their booking, but only if there is more than a week until their booking starts. A consumer may not edit their booking, and if required, would need to cancel and re-book.

3.6 Providing and managing car spaces for rent

Providers are able to easily list their car spaces for rent by filling out the online form in the Provider tab. To list a car space, several details such as an address, size and pricing must be filled out as well as the start and end dates for the period on which the car space is made available to rent. The provider is required to upload at least one image of the car space.

Parking space listings must be manually approved by a Staff member before being added to the publicly visible pool. This is to ensure high-quality listings that accurately

display the parking space and its features. Providers are able to view the status of their listings in the Providers tab.

Providers are able to edit their listings at any time, however, they may not change their availability such that it would violate any existing bookings. However, if a provider wishes to stop providing their parking space, they may cancel the space and no new bookings will be allowed. As a condition of use, providers must still honour any pre-existing bookings made and should set the availability of their parking space appropriately.

3.7 Platform administration

Staff members (also referred to as ‘admins’) are able to moderate and review all activity on Park It. Through the use of the Admin Dashboard, they are able to modify and delete any listings, reviews, bookings etc if it proves to be necessary. In order to account for large teams, staff members are able to grant other users' staff permissions, making it scalable to the needs of the company.

The primary function of the admins other than general maintenance is to manually approve car space applications to ensure that they meet BMG standards for quality. This is easily done through the admin dashboard.

3.8 Bookmarking favourite car spaces

A unique feature of Park It is the ability for the consumer to bookmark certain car space listings so that a consumer may easily find them again. This feature might be used to short-list desirable spaces, or just to rebook a space that was previously rented by the consumer. Each car space that is bookmarked or “favourited” is added to a list that is accessible on the Account page.

3.9 Reviewing car spaces

The second unique feature of Park It is the ability to leave written reviews for car spaces. These reviews include both written feedback and a review score out of 5, which makes it easier for future customers to find a high quality parking space. This ensures a high level of consumer satisfaction as they are able to both provide feedback to the provider, and ensure that their future car space rentals have been vetted by other fellow consumers.

Consumers may only leave a review for car spaces they have booked, and only after their booking elapses.

4 Third-Party Functionality

4.1 Google Maps

Google Maps API was used to support the backend's validation of parking space functionalities. Utilising Google's i18n address metadata allowed for the backend to validate and clean addresses passed from the presentation layer. These cleaned addresses can then be passed to the Google Maps endpoint to retrieve the respective latitude and longitude of locations. It was necessary to use the Google Maps API as no other free service provided the necessary level of accuracy when determining the geolocation of given addresses. We are using the free-tier of the Maps API, and the licence can be found here https://cloud.google.com/maps-platform/terms (Google, 2020). In production, a higher tier to guarantee operation under heavy load may be necessary.
It can be found at https://developers.google.com/maps (Google, 2022).

4.2 Django

We used the Django web framework to power our backend to save time and effort in producing a robust and well-tested backend with appropriate security features built in. The framework is open-source and written in Python. Using Django gave us more time to build important features rather than spending time re-inventing the wheel. The inbuilt ORM meant a large amount of saved effort in writing and duplicating raw SQL queries to the SQLite3 database and would have made the backend very difficult to maintain and document. The framework is open-source and per the licence (found at https://github.com/django/django/blob/main/LICENSE) has certain conditions for display of copyright, conditions and disclaimer but is otherwise freely usable (Django Software Foundation, 2013).

It can be found at https://www.djangoproject.com/ (Django Software Foundation, 2022).

4.3 Django Rest Framework

We used Django Rest Framework to provide a REST API so that our backend and frontend could communicate. This gave us the ability to choose and switch our frontend freely, and decoupled the frontend from the backend providing more flexibility and modularity. This had the added benefit of making the backend codebase much easier to maintain. The framework is open source and per the licence (found at https://github.com/encode/django-rest-framework/blob/master/LICENSE.md) has certain conditions for display of copyright, conditions and disclaimer but is otherwise freely usable (Encode OSS, 2018).
It can be found at https://github.com/encode/django-rest-framework (Encode OSS, 2021).

4.4 dj-rest-auth

We used the DJ-rest-auth library to manage account authentication as it ensured account registration and authentication were secure and seamless as it used industry-standard JWT authentication. It also allowed for email verification functionality which increased the security of the application. The library is open source and under the MIT licence (Github, 2022).
It can be found at https://github.com/iMerica/dj-rest-auth (iMerica, 2022).

4.5 Jazzmin

We used Jazzmin to theme the admin dashboard that comes inbuilt with Django in order to provide a modern UX and UI for that interface and match the look and feel of the rest of the web application. The library is open source and under the MIT licence (Github, 2022).
It can be found at https://github.com/farridav/django-jazzmin (Farrington, 2022).

4.6 React

We used the React library to create our frontend. This allowed us to produce a modern UX and UI, and maintain a consistent look and feel across Park It. Using React made the process of creating the frontend much faster and allowed us to effectively meet deadlines and deliver Park It. Using an industry standard framework also makes the codebase much more maintainable. React is open source and under the MIT licence (Github, 2022).
It can be found at https://github.com/facebook/react (Facebook, 2022).

4.7 react-custom-scrollbars-2

We used a custom scroll bar component instead of the default React scrollbar because we felt like it improved the visual appeal of our web application. The component is open source and under the MIT licence (Github, 2022).
It can be found at https://github.com/RobPethick/react-custom-scrollbars-2 (Pethick, 2022).

4.8 react-material-ui-carousel

We used a custom Carousel component to allow for a nice looking gallery feature for car space images. The component is open source and under the MIT licence (Github, 2022).
It can be found at https://github.com/Learus/react-material-ui-carousel (Maliaras, 2021).

4.9 drf-writable-nested

We used a library to add nested object functionality to handle parking spaces with multiple images, which made it significantly easier for the front end team. This was because SQLite3 does not support storing lists so another solution was necessary to

provide the correct endpoint responses. The library is open source and per the licence (found at https://github.com/beda-software/drf-writable-nested/blob/master/LICENSE.md) has certain conditions for display of copyright, conditions and disclaimer but is otherwise freely usable (Beda Software, 2022).

It can be found at https://github.com/beda-software/drf-writable-nested/ (Beda Software, 2022).

4.10 Leaflet

Leaflet was used to create interactive maps that allowed users to pan and zoom in on the map. Furthermore, it allowed for the placing of map markers corresponding to the geolocation of parking spaces and made it easy to see where their potential parking spaces were located on the map. Leaflet is open source and under the BSD-2 Licence (Github, 2022).
It can be found at https://github.com/Leaflet/Leaflet (Agafonkin, 2022).

4.11 Material UI

Material UI was used for styling the frontend and providing React components that made building an attractive and compelling UI and UX relatively less time consuming. Material UI is an industry standard React library, and using it helped us ensure a consistent look and feel across the platform. Material UI is open source and under the MIT licence (Github, 2022).
It can be found at https://github.com/mui/material-ui (MUI, 2022).

4.12 SQLite3

We used SQLite3 as our database as it was already built into Django and therefore needed no further configuration (Django Software Foundation, 2022).

It can be found at https://www.sqlite.org (SQLite Consortium, 2022).

4.13 Gmail

We used Gmail as our email server because it was free and easy to set up. It provided a secure way to send email address verification emails.

It can be found at https://mail.google.com/ (Google, 2022).

5 Implementation Challenges

5.1 Maps API

We were originally using the OpenStreetMap API for address validation and determining the geolocation of the parking spaces (OpenStreetMap Foundation, 2022). We required the coordinates to show map markers for the relevant car spaces on the interactive maps. We discovered midway during the development that the database the OpenStreetMap API relies upon does not have street number specific coordinates for most Australian addresses, and in most cases would default to the same coordinates for a given street, no matter what street number was given.

For example, the API returned the same coordinates for 1 High Street and 100 High Street. This meant the map markers on a given street all pointed to the same spot, even if in reality they were hundreds of metres away. This level of accuracy was unsuitable for our purposes so we explored other options and eventually chose Google Maps API as a replacement as they were the only free service that gave us the geolocation accuracy needed for accurate map markers (Google, 2022).

5.2 Django - full stack vs REST API

We had the option of using Django as the framework of our entire application, from the frontend to backend, and this design was how Django was originally envisioned (Django Software Foundation, 2022). However we chose to add Django Rest Framework which allowed us to build a REST API that accepted and retrieved data as needed (Encode OSS, 2018). This allowed our system to be much more modular, and gave us the freedom and flexibility to select a modern frontend framework like React (Facebook, 2022). In theory, we could switch out the entire front end without ever having to touch the backend, and likewise we could switch out the backend without needing to touch the

front end. This separation made the code base much more maintainable and it was easier to isolate any problems to one half of the web application.

5.3 Search algorithm

As Park It is essentially a rental system, the most fundamental functionality was the ability for users to book car spaces and a robust search algorithm is key to showing users car spaces relevant to their needs. After discussions on what users want to see when searching for car spaces, we decided on several distinguishing features.

1. With the inclusion of the desired start and end time from the user when searching, we decided that only car spaces that were available for that period would be shown to the user. This was done to show spaces that aligned with consumers' needs.
2. By including a radius option, users are able to vary the distance from their destination they want their parking spot to be. This allows for flexibility in situations where the final section of the journey can be completed by walking or taking public transport.
3. By excluding a user's own spots from the results, this prevents abuse from providers making fake bookings and reviews to change the perception of their space.

5.4 Booking algorithm

Another crucial component to allow Park It to function as a platform was creating an efficient booking algorithm to support users' needs. There were many variables that had to be taken into account, some of which were added late into the process on the advice of our tutors. Those variables include car space availability, booking duration, avoiding booking overlap, and appropriately handling certain cases like the editing and cancellation of car spaces and bookings and appropriate validation logic for those things.

We discussed as a group the best way to handle these situations and determined that:

1. Providers agree that when they set their availability, they guarantee to honour all bookings made.
2. Providers may edit their availability but only so long as it doesn't violate any existing bookings.
3. Providers may cancel their listing, but all pre-existing bookings must be honoured. No new bookings will be allowed by the system, and the parking space will no longer appear in the search.
4. Consumers may only cancel bookings if there is more than a week until their booking.
5. Consumers may not edit their bookings, and are required to cancel and rebook if necessary.

5.5 Material UI (MUI)

We used Material-UI for having a consistent style across all components in the system and setting up the initial environment to use Material-UI was a bit challenging since it was my first time to use Material-UI (MUI, 2022). For example, we set up a "ThemeProvider" component to use the common theme for all Material-UI components and also set up "StyledEngineProvider" to style Material-UI components with css modules.

Also, there were some challenges in how to customise each Material-UI component to our needs. Since most Material-UI components are composed of many other Material-UI components, we often had to explore Material-UI official document from the top-level component to the base-level component and had to search up for the actual use case for each styling attribute since often there was not a sample usage case for a styling attribute.

6 References

Agafonkin, V. (2022). *Leaflet/Leaflet: JavaScript library for mobile-friendly interactive maps*. GitHub. Retrieved April 20, 2022, from https://github.com/Leaflet/Leaflet

Beda Software. (2022). *beda-software/drf-writable-nested: Writable nested model serializer for Django REST Framework*. GitHub. Retrieved April 20, 2022, from https://github.com/beda-software/drf-writable-nested/

Beda Software. (2022). *drf-writable-nested/LICENSE.md at master · beda-software/drf-writable-nested*. GitHub. Retrieved April 20, 2022, from https://github.com/beda-software/drf-writable-nested/blob/master/LICENSE.md

Django Software Foundation. (2013). *django/LICENSE at main · django/django · GitHub*. GitHub. Retrieved April 20, 2022, from https://github.com/django/django/blob/main/LICENSE

Django Software Foundation. (2022). Django: The web framework for perfectionists with deadlines. Retrieved April 20, 2022, from https://www.djangoproject.com/

Encode OSS. (2018). *django-rest-framework/LICENSE.md at master · encode/django-rest-framework*. GitHub. Retrieved April 20, 2022, from https://github.com/encode/django-rest-framework/blob/master/LICENSE.md

Encode OSS. (2021). *encode/django-rest-framework: Web APIs for Django.* GitHub. Retrieved April 20, 2022, from https://github.com/encode/django-rest-framework

Facebook. (2022). *facebook/react: A declarative, efficient, and flexible JavaScript library for building user interfaces.* GitHub. Retrieved April 20, 2022, from https://github.com/facebook/react

Farrington, D. (2022). *farridav/django-jazzmin: Jazzy theme for Django*. GitHub. Retrieved April 20, 2022, from https://github.com/farridav/django-jazzmin

Github. (2022). *BSD 2-Clause "Simplified" License*. Choose a License. Retrieved April 20, 2022, from https://choosealicense.com/licenses/bsd-2-clause/

Github. (2022). *MIT License*. Choose a License. Retrieved April 20, 2022, from https://choosealicense.com/licenses/mit/

Google. (2020). *Google Maps Platform Terms Of Service*. Google Cloud. Retrieved April 20, 2022, from https://cloud.google.com/maps-platform/terms

Google. (2022). *Gmail*. Google. Retrieved April 20, 2022, from https://mail.google.com/

Google. (2022). *Google Maps Platform*. Google Developers. Retrieved April 20, 2022, from https://developers.google.com/maps

iMerica. (2022). *iMerica/dj-rest-auth: Authentication for Django Rest Framework*. GitHub. Retrieved April 20, 2022, from https://github.com/iMerica/dj-rest-auth

Maliaras, Y. (2021). *Learus/react-material-ui-carousel: A Generic carousel UI component for React using Material UI.* GitHub. Retrieved April 20, 2022, from https://github.com/Learus/react-material-ui-carousel

MUI. (2022). *mui/material-ui: MUI Core (formerly Material-UI) is the React UI library you always wanted. Follow your own design system, or start with Material Design.* GitHub. Retrieved April 20, 2022, from https://github.com/mui/material-ui

OpenStreetMap Foundation. (2022). OpenStreetMap.org. Retrieved April 20, 2022, from https://www.openstreetmap.org/

Pethick, R. (2022). *RobPethick/react-custom-scrollbars-2*. GitHub. Retrieved April 20, 2022, from https://github.com/RobPethick/react-custom-scrollbars-2

SQLite Consortium. (2022). SQLite Home Page. Retrieved April 20, 2022, from https://www.sqlite.org/