Use Cases

for

ParkWhere

Version 1.0 approved

Prepared by
Aaron Lim Wee Him
Goh Yi Heng
Kaung Set @John Xie
Ong Shun Ping

Nanyang Technological University

24/01/2024

Revision History

Name	Date	Reason For Changes	Version
Aaron Lim	04/02/2024	First draft	1.0
Shun Ping Ong	24/3/2024	Change Reset Password use case	1.1

Guidance for Use Case Template
Document each use case using the template shown in the Appendix. This section provides a description of each section in the use case template.

1. **Use Case Identification**

1.1. Use Case ID

Give each use case a unique numeric identifier, in hierarchical form: X.Y. Related use cases can be grouped in the hierarchy. Functional requirements can be traced back to a labelled use case.

1.2. Use Case Name

State a concise, results-oriented name for the use case. These reflect the tasks the user needs to be able to accomplish using the system. Include an action verb and a noun. Some examples:

- View part number information.
- Manually mark hypertext source and establish link to target.
- Place an order for a CD with the updated software version.

1.3. Use Case History

1.3.1 Created By

Supply the name of the person who initially documented this use case.

1.3.2 Date Created

Enter the date on which the use case was initially documented.

1.3.3 Last Updated By

Supply the name of the person who performed the most recent update to the use case description.

1.3.4 Date Last Updated

Enter the date on which the use case was most recently updated.

2. **Use Case Definition**

2.1. Actor

An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks. Different actors often correspond to different user classes, or roles, identified from the customer community that will use the product. Name the actor(s) that will be performing this use case.

2.2. Description

Provide a brief description of the reason for and outcome of this use case, or a high-level description of the sequence of actions and the outcome of executing the use case.

2.3. Preconditions

List any activities that must take place, or any conditions that must be true, before the use case can be started. Number each precondition. Examples:

1. User's identity has been authenticated.
2. User's computer has sufficient free memory available to launch task.

2.4. Postconditions

Describe the state of the system at the conclusion of the use case execution. Number each postcondition. Examples:

1. Document contains only valid SGML tags.
2. Price of item in database has been updated with new value.

2.5. Priority

Indicate the relative priority of implementing the functionality required to allow this use case to be executed. The priority scheme used must be the same as that used in the software requirements specification.

2.6. Frequency of Use

Estimate the number of times this use case will be performed by the actors per some appropriate unit of time.

2.7. Flow of Events

Provide a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description. This description may be written as an answer to the hypothetical question, "How do I <accomplish the task stated in the use case name>?" This is best done as a numbered list of actions performed by the actor, alternating with responses provided by the system.

2.8. Alternative Flows

Document other, legitimate usage scenarios that can take place within this use case separately in this section. State the alternative course, and describe any differences in the sequence of steps that take place. Number each alternative course using the Use Case ID as a prefix, followed by "AC" to indicate "Alternative Course". Example: X.Y.AC.1.

2.9. Exceptions

Describe any anticipated error conditions that could occur during execution of the use case, and define how the system is to respond to those conditions. Also, describe how the system is to respond if the use case execution fails for some unanticipated reason. Number each exception using the Use Case ID as a prefix, followed by "EX" to indicate "Exception". Example: X.Y.EX.1.

2.10. Includes

List any other use cases that are included ("called") by this use case. Common functionality that appears in multiple use cases can be split out into a separate use case that is included by the ones that need that common functionality.

2.11. Special Requirements

Identify any additional requirements, such as nonfunctional requirements, for the use case that may need to be addressed during design or implementation. These may include performance requirements or other quality attributes.

2.12. Assumptions

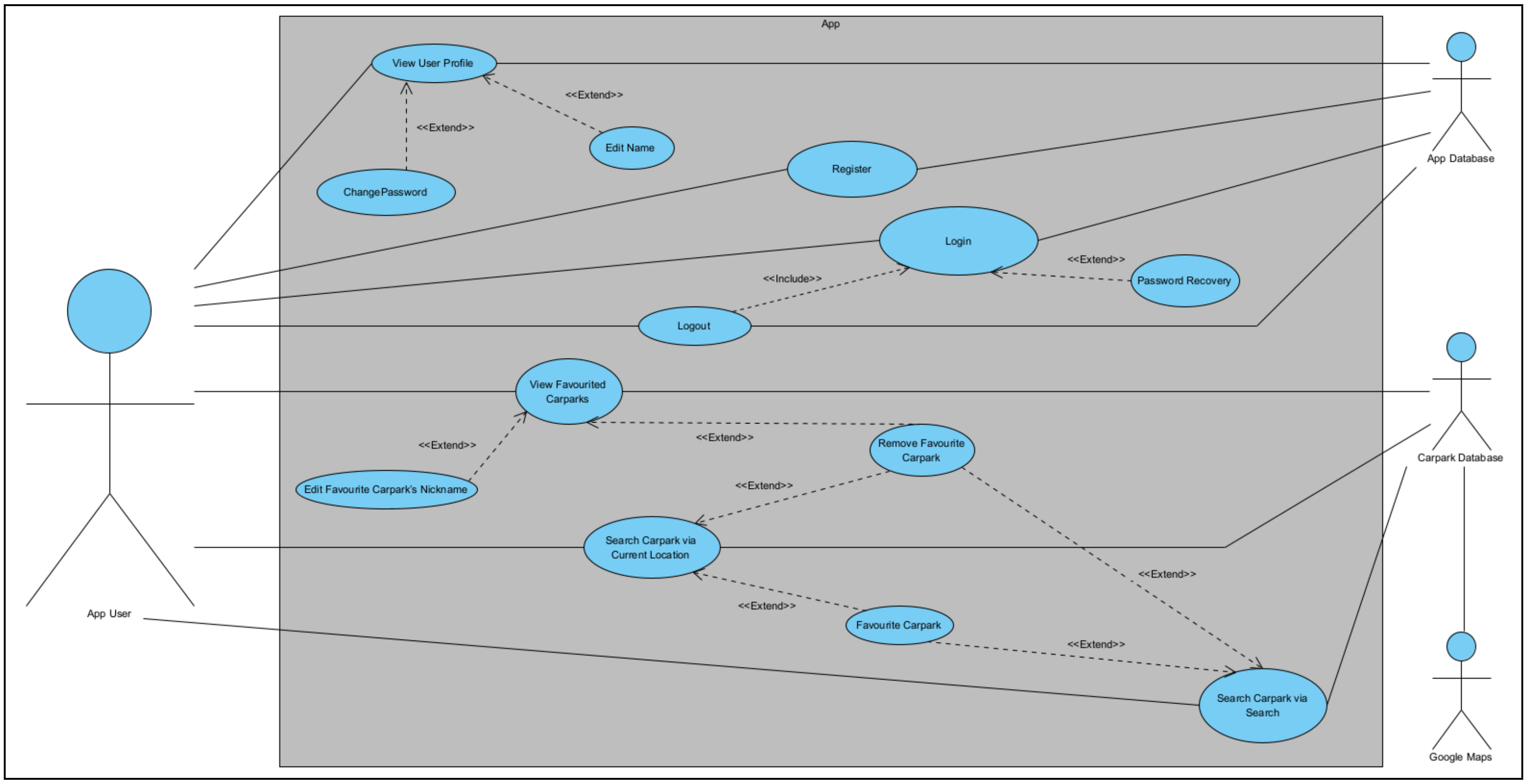
List any assumptions that were made in the analysis that led to accepting this use case into the product description and writing the use case description.

2.13. Notes and Issues

List any additional comments about this use case or any remaining open issues or TBDs (To Be Determined) that must be resolved. Identify who will resolve each issue, the due date, and what the resolution ultimately is.

USE CASE MODEL

USE CASE DIAGRAM



* Please refer to the Use Case Diagram on GitHub for better visibility

USE CASE TEMPLATE

Use Case ID:	01		
Use Case Name:	Register		
Created By:	Yi Heng	Last Updated By:	Aaron
Date Created:	30/01/2024	Date Last Updated:	03/02/2024

Actor:	User, System, Google API
Description:	Register for a new account (locally and Google)
Preconditions:	User had downloaded and opened the application
Postconditions:	1) User account created 2) App displays login page
Priority:	High
Frequency of Use:	Low
Flow of Events:	Local: 1) User enters login page 2) User clicks on 'Create Account' 3) User inputs email as user ID 4) User inputs password 5) App verifies user details 6) App saves user details in database 7) App displays successful account creation Google: a) User enters Google login page b) User clicks on 'Create Account' c) User clicks on 'Link Account' d) User inputs Google email address e) User inputs password f) Google API verifies user details g) App displays successful account creation
Alternative Flows:	01-AF-S3 User inputs invalid email 1) App displays 'Error, please input valid email' 2) User returns to step 3) 01-AF-S3 User inputs email that already exists 1) App displays 'Error, email already exists' 2) User returns to step 3) 01-AF-S4 User inputs invalid password 1) App displays 'Error, password does not meet requirements' 2) User returns to step 3) 01-AF-Sd User inputs invalid Google email

	a) App displays 'Error, please input valid email' b) User returns to step d) 01-AF-Sd User inputs email that already exists a) App displays 'Error, email already exists' b) User returns to step d) 01-AF-Se User inputs invalid password a) App displays 'Error, password does not meet requirements' b) User returns to step e)
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. Users must be able to register for an account on the system manually or using their Google Account.
   - 1.1. The system must display text fields for the user to enter his/her information on account creation for manual creation.
     - 1.1.1. The text fields must consist of Full name.
     - 1.1.2. The text fields must consist of Email.
     - 1.1.3. The text fields must consist of Password.
     - 1.1.4. The text fields must consist of Confirm password.
     - 1.1.5. The user must fill in all the text fields before clicking "Create your account."
       - 1.1.5.1. The system must verify the information when the user clicks the button.
         - 1.1.5.1.1. The email given has never been registered in the system.
         - 1.1.5.1.2. The email given must be in the correct format.
         - 1.1.5.1.3. The password given must contain at least 1 uppercase character.
         - 1.1.5.1.4. The password given must contain at least 1 lowercase character.
         - 1.1.5.1.5. The password given must contain at least 8 characters.
         - 1.1.5.1.6. The password given must contain at least 1 number.
         - 1.1.5.1.7. The system must provide error messages describing the reason the account creation is rejected.
   - 1.2. The system must display text fields for the user to enter his/her information on account creation for Google creation.
     - 1.2.1. The text fields must consist of Google Email.
     - 1.2.2. The text fields must consist of Google Password.
     - 1.2.3. The user must fill in all the text fields before clicking "Sign Up."
       - 1.2.3.1. The Google Firebase system must verify the information when the user clicks the button.
   - 1.3. The system must create an account for the user upon verification.
   - 1.4. The system must log the user into the main page of the system.

Use Case ID:	02		
Use Case Name:	Login		
Created By:	Yi Heng	Last Updated By:	Aaron
Date Created:	30/01/2024	Date Last Updated:	03/02/2024

Actor:	User, System
Description:	User must log in to the system
Preconditions:	User must already be registered in the application
Postconditions:	1) User logged in 2) App displays main menu
Priority:	High
Frequency of Use:	Medium
Flow of Events:	Local: 1) User enters login page 2) User inputs email as user id 3) User inputs password 4) User clicks on 'Login' 5) App verifies user details 6) App goes to main menu Google: a) User enters Google login page b) User inputs Google email as user id c) User inputs password d) User clicks on 'Login' e) App verifies user details f) App goes to main menu
Alternative Flows:	02-AF-S4 User inputs invalid email 1) App displays 'Error, please input valid email' 2) User returns to step 2) 02-AF-S4 User inputs invalid password 1) App displays 'Error, incorrect password' 2) User returns to step 3) 02-AF-S4 User inputs wrong password 5 times 1) App displays 'Account has been locked, please contact admin' 2) User returns to step 3) 02-AF-S4 User inputs invalid Google email a) App displays 'Error, please input valid Google email' b) User returns to step b) 02-AF-S4 User inputs invalid password a) App displays 'Error, incorrect password' b) User returns to step c) 02-AF-S4 User inputs wrong password 5 times

	a) App displays 'Account has been locked, please contact admin' b) User returns to step c)
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to login to the system manually or using their Google Account.
   - 1.1.1. The system must display text fields for the user to enter his information to login manually.
     - 1.1.1.1. The text fields must consist of Email.
     - 1.1.1.2. The text fields must consist of Password.
   - 1.1.2. The system must display text fields for the user to enter his information to login via the Google account.
     - 1.1.2.1. The text fields must consist of Google email.
     - 1.1.2.2. The text fields must consist of Google password.
   - 1.1.3. The user must fill in all of the text fields before clicking 'Sign In' button
     - 1.1.3.1. The system must verify the fields filled in by the user before logging the user into the system.
     - 1.1.3.2. The system must display an error message if the account does not exist.
   - 1.1.4. The system must log in the user if the information obtained from the text fields are verified.
     - 1.1.4.1. The email must be found in the database of the system.
       - 1.1.4.1.1. The password matches the corresponding email of the user.
     - 1.1.4.2. The Google account must be found in the database of the System.
   - 1.1.5. The user must be able to see the main menu interface upon login.

Use Case ID:	03		
Use Case Name:	Logout		
Created By:	Yi Heng	Last Updated By:	
Date Created:	30/01/2024	Date Last Updated:	

Actor:	User, System
Description:	User logs out of system
Preconditions:	User must be logged in to the App
Postconditions:	User is logged out of the App
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1) User opens main menu 2) User selected 'Log Out' 3) User is logged out of App 4) App returns to log in page
Alternative Flows:	NIL
Exceptions:	NIL
Includes:	Login
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to logout from the system.
   - 1.1. The system main menu must display a button "Log Out" to exit the system.
     - 1.1.1. Upon clicking "Log Out", the user must be navigated back to the Login page.
   - 1.2. The user's information must still remain in the system for relogin.
   - 1.3. The system must display the same screen shown at the login page to the user upon Logout.

Use Case ID:	04		
Use Case Name:	Search Carpark via Current Location		
Created By:	Yi Heng	Last Updated By:	
Date Created:	30/01/2024	Date Last Updated:	

Actor:	User, Carpark API
Description:	User searches carparks nearby with current location
Preconditions:	User has logged into the App
Postconditions:	App displays nearby carparks
Priority:	High
Frequency of Use:	High
Flow of Events:	1) User selects 'Carparks near me' 2) Carpark API locates current location 3) Carpark API returns carpark locations nearby 4) App displays nearby carpark information
Alternative Flows:	04-AF-S3 Carpark API unable to find carparks nearby 1) App displays 'unable to find carparks nearby' 2) App returns to main menu
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to search carpark via current location.
    - 1.1. The system main menu must display carparks near the users.
        - 1.1.1. The system must show the carparks nearby the user's current location in a circle of 1 kilometer radius.
        - 1.1.2. Carparks displayed on the map must be clickable, allowing users to access more information
            - 1.1.2.1. Clicking on a carpark marker must lead to a page displaying detailed information.
                - 1.1.2.1.1. The detailed information must consist of Carpark name
                - 1.1.2.1.2. The detailed information must consist of Live availability
                - 1.1.2.1.3. The detailed information must consist of Distance
                - 1.1.2.1.4. The detailed information page must include a "Directions" button
                    - 1.1.2.1.4.1. Clicking the "Directions" button must initiate the process of providing directions
                    - 1.1.2.1.4.2. The system must display an outlined route from the user's current location to the selected carpark
                    - 1.1.2.1.4.3. The system must allow the user to cancel navigation at anytime by clicking the "X".

Use Case ID:	05		
Use Case Name:	View User Profile		
Created By:	Shun Ping	Last Updated By:	Aaron
Date Created:	30/1/2024	Date Last Updated:	03/02/2024

Actor:	User, System
Description:	User view profile related to self
Preconditions:	User has logged into the App
Postconditions:	App displays information regarding user
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1) User selects 'View User Profile" 2) System navigates to page with all the information of the user 3) System display all user information 4) If the user selects the back button, the system returns to the selection screen. 5) If the user chooses to edit profile, user can choose between "Edit Name" or "Change Password". 6) If user chooses "Edit Name", then he uses the included use case Edit Name to edit the user name. 7) If user chooses "Change Password", then he uses the included use case Change Password to change the password. 8) System updates user profile based on option carried out.
Alternative Flows:	NIL
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to view their current profile.
   - 1.1. The system must display a button "View User Profile" to view the user's profile.
     - 1.1.1. The system must display Name Registered.
     - 1.1.2. The system must display the Registered Email Address.
   - 1.2. User must be able to able to edit name.
   - 1.3. User must be able to change passwords.

Use Case ID:	06		
Use Case Name:	Search Carpark via Search		
Created By:	Yi Heng	Last Updated By:	Aaron
Date Created:	30/01/2024	Date Last Updated:	03/02/2024

Actor:	User, Carpark API
Description:	User searches carparks nearby with postal code or name
Preconditions:	User has logged into App
Postconditions:	App displays nearby carparks
Priority:	High
Frequency of Use:	High
Flow of Events:	1) User selects 'Search For Carparks' 2) User enters postal code or name of location 3) Carpark API searches for carparks nearby based on entered location 4) Carpark API returns carpark locations nearby 5) App displays nearby carpark information
Alternative Flows:	06-AF-S3 Carpark API unable to find carparks nearby 1) App displays 'unable to find carparks nearby' 2) Go to step 2)
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to search carpark via search bar.
    - 1.1. The system main menu must display "Search" for user to enter his/her information.
        - 1.1.1. After selecting "Carparks via search bar", the system must display a text field for users to input their information.
            - 1.1.1.1. The text field must consist of "Enter Postal Code/ Enter Street Name"
                - 1.1.1.1.1. User must key in either postal code or street name to locate the carpark
                    - 1.1.1.1.1.1. The system must verify the information filled in the field.
                    - 1.1.1.1.1.2. The system must display "No results found. Try again" if not found.
                - 1.1.1.1.2. The system must show the carparks nearby the address in a circle of 1km radius if input is found on the map.
        - 1.1.2. Carparks displayed on the map must be clickable, allowing users to access more information
            - 1.1.2.1. Clicking on a carpark marker must lead to a page displaying detailed information.
                - 1.1.2.1.1. The detailed information must consist of Carpark name

1.1.2.1.2. The detailed information must consist of Live availability
1.1.2.1.3. The detailed information must consist of Distance
1.1.2.1.4. The detailed information page must include a "Directions" button
    1.1.2.1.4.1. Clicking the "Directions" button must initiate the process of providing directions
    1.1.2.1.4.2. The system must display an outlined route from the user's current location to the selected carpark
    1.1.2.1.4.3. The system must allow the user to cancel navigation at anytime by clicking the “X”

Use Case ID:	07		
Use Case Name:	Favourite Carpark		
Created By:	Yi Heng	Last Updated By:	Aaron
Date Created:	30/01/2024	Date Last Updated:	03/02/2024

Actor:	User, System, Carpark API
Description:	User favourites carparks
Preconditions:	User has logged into App
Postconditions:	Carparks has been favourited to their profile.
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1) App displays nearby carpark information after 'Search Carpark via Current Location' or 'Search Carpark via Search' 2) User clicks on carpark they wish to favourite 3) User clicks on 'Favourite Carpark' 4) App adds carpark to 'Favourite Carparks' under Favourite Carparks in main menu 5) App display "successfully favourited carpark"
Alternative Flows:	07-AF-S3 'Favourite Carparks' exceeds the limit of 5 1) App displays "Favourite Carparks list is full" 2) Go to step 1)
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to favourite carparks in the app
    - 1.1. Clicking on a carpark marker on the map must lead to a page displaying detailed information.
        - 1.1.1. The detailed information page must include a "Favourite" button if the carpark does not exist in the User's list of favourite carparks.
            - 1.1.1.1. Clicking on "Favourite" button must add the carpark to the User's list of Favourite Carparks

Use Case ID:	08		
Use Case Name:	Remove Favourite Carpark		
Created By:	John	Last Updated By:	Aaron
Date Created:	30/01/2024	Date Last Updated:	03/02/2024

Actor:	User, System, Carpark API
Description:	User removes a favourited carpark (by favourite list or by map)
Preconditions:	User has logged into App, User has favourite carparks
Postconditions:	Favourited carpark has been removed from their profile
Priority:	Low
Frequency of Use:	Low
Flow of Events:	Favourite List 1) User clicks on ‘View Favourite Carparks’ in main menu 2) App displays list of favourite carparks 3) User clicks on a favourited carpark they wish to unfavourite 4) User clicks “Unfavourite Carpark” 5) App removes favourited carpark from user’s favourite list 6) App displays successful removal of favourited carpark Search a) App displays nearby carpark information after ‘Search Carpark via Current Location’ or ‘Search Carpark via Search’ b) User clicks on a favourited carpark they wish to unfavourite c) User clicks on ‘Unfavourite Carpark’ d) App removes carpark from User profile e) App display successful removal of favourite carpark
Alternative Flows:	08-AF-S3 User has no favourite carpark 1) App displays ‘No favourite carparks found’ 2) Go to main menu
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to remove favourited carparks in the app
    1.1. System must display carpark information after clicking on carpark on map.
        1.1.1. The detailed information page must include a “Unfavourite” button if the carpark already exists in the User’s list of favourite carparks.
            1.1.1.1. Clicking on “Unfavourite” button must remove the carpark from the User’s list of Favourite Carparks
    1.2. Clicking on favourited carpark in View Favourite Carparks in main menu must display favourited carparks information.
        1.2.1. The information page must include a “Unfavourite” button if the carpark already exists in the User’s list of favourite carparks.
            1.2.1.1. Clicking on “Unfavourite” button must remove the carpark from the User’s list of Favourite Carparks

Use Case ID:	09		
Use Case Name:	View Favourited Carparks		
Created By:	Yi Heng	Last Updated By:	Aaron
Date Created:	30/01/2024	Date Last Updated:	03/02/2024

Actor:	User, System, Carpark API
Description:	User view favourited carparks
Preconditions:	User has logged into App
Postconditions:	App displays list of favourited carparks
Priority:	High
Frequency of Use:	High
Flow of Events:	1) User clicks on 'View Favourite Carparks' in main menu 2) App displays list of favourite carparks 3) User clicks on a favourited carpark 4) App searches for carpark information in Carpark API 5) App displays carpark information of favourite carparks
Alternative Flows:	08-AF-S1 User has no favourite carpark 1) App displays 'No favourite carparks found' 2) Go to main menu
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to view their favourite carparks.
    - 1.1. The system must have a "View Favourite Carparks" for the user to click on.
    - 1.2. The system must display the lists of favourite carparks once they clicked on the "View Favourite Carparks".
    - 1.3. The user must be able to select a carpark they have favourited before.
        - 1.3.1. The system must display carpark information about the selected favourite carpark.
            - 1.3.1.1. The information must consist of carpark availability.
            - 1.3.1.2. The information must consist of distance from current location.
            - 1.3.1.3. The information must consist of name of carpark saved as.
        - 1.3.2. The system must provide a "Directions" button next to each favourited carpark
            - 1.3.2.1. The system must initiate a navigation service to guide the user from their current location to the selected carpark upon clicking on the "Directions" button.
        - 1.3.3. The system must allow user to cancel navigation at any time.
            - 1.3.3.1. The system must provide a "X" button to cancel the current navigation.

Use Case ID:	10		
Use Case Name:	Password Recovery		
Created By:	Aaron	Last Updated By:	Shun Ping
Date Created:	01/02/2024	Date Last Updated:	24/03/2024

Actor:	User, System
Description:	App resets user password
Preconditions:	User must already be registered in the application
Postconditions:	Password of the user in the database has been updated with the new password.
Priority:	High
Frequency of Use:	Low
Flow of Events:	1) User clicks on forgot password in the login page 2) App will take user to reset password page 3) App will prompt user for their email 4) User enters email 5) User clicks send email 6) System will send reset link to the email 7) User check email for link 8) User follow steps from email link 9) User enters new password 10) System changes password of user
Alternative Flows:	09-AF-S3 Email not in database 1) App outputs ‘invalid email’ 2) App returns to step 2
Exceptions:	NIL
Includes:	NIL
Special Requirements:	User is able to return back to login page at anytime
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to reset his/her password.
    1.1. The system must have a “Forgot password” on the login page.
        1.1.1. Upon clicking “Forgot password”, the user must be navigated to the reset password page.
            1.1.1.1. The system must display text fields for the user to enter his/her information to reset the password
                1.1.1.1.1. The text fields must consist of email.
                1.1.1.1.2. The system must display a send password link button.
    1.2. The user must fill in the email text field before clicking the send password link.
        1.2.1. The system will check the app database to see if an account with the email exists.
        1.2.2. The system will then send a reset password link to the email.
    1.3. The user must go to the email and click on the link.
    1.4. The user must fill in the new password.

1.5. The system must verify the fields when the user clicks the Reset Password button.
    - 1.5.1.1. The new password must consist of at least 1 uppercase character.
    - 1.5.1.2. The new password must consist of at least 1 lowercase character.
    - 1.5.1.3. The new password must consist of at least 8 characters.
    - 1.5.1.4. The new password must consist of at least 1 number.
    - 1.5.1.5. The Confirm New Password must match the New Password.
    - 1.5.1.6. The inputted OTP must match the OTP sent by the system to the user's email
  - 1.5.2. The system must provide error messages describing the reason the user's password is rejected.

1.6. The system must reset the password upon successful verification.

.

Use Case ID:	11		
Use Case Name:	Edit Name		
Created By:	Shun Ping	Last Updated By:	
Date Created:	01/02/2024	Date Last Updated:	

Actor:	User, System
Description:	User change the name that is stored on the system.
Preconditions:	User has clicked on “View User Profile”
Postconditions:	Name of user in database has been updated with the new name.
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1) User clicks on “Edit Name” on the profile page 2) System displays the current “First name” and “Last name” to the user. 3) System prompts the user to enter the new first name and last name. 4) User enters the information and clicks on “Save”. 5) System stores and overwrites the new details into the database. 6) System changes the name of the user. 7) App will display the new name for the user.
Alternative Flows:	NIL
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to edit their own name on the app.
    - 1.1. The system must display the user's current name as a reference.
    - 1.2. The system must display text fields for the user to enter the name they want to change to.
        - 1.2.1. The text field must consist of “First name”
        - 1.2.2. The text field must consist of “Last name”
    - 1.3. The system must have a “Save” button to confirm the new name.
    - 1.4. The system must have a “<-” button to return to the previous state.
    - 1.5. The system must verify the text field before updating the name of the user..
        - 1.5.1. The text fields must not be empty.
    - 1.6. The system must update the name upon successful verification.

Use Case ID:	12		
Use Case Name:	Edit Favourite Carpark's Nickname		
Created By:	Shun Ping	Last Updated By:	
Date Created:	01/02/2024	Date Last Updated:	

Actor:	User, System, Carpark API
Description:	User edit the name of the carparks he/she favourited
Preconditions:	1) User has clicked on "View User Profile" 2) User has clicked on "View Favourited Carparks"
Postconditions:	Nicknames of carparks in database has been updated with the new value.
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1) User clicks on "View Favourited Carparks" on the main menu, then he uses the included use case View Favourite Carparks. 2) User selects the favourited carpark to rename. 3) User selects "Rename" from the pop up menu 4) System prompts the user to enter the new carpark's nickname 5) User enters the information and clicks on "Save". 6) System stores and overwrites the new details into the database. 7) System changes the nickname of the carpark for the user. 8) App will display the new nickname for the favourited carparks.
Alternative Flows:	NIL
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to edit the nickname of favourited carparks.
    - 1.1. The system must provide a user interface that allows users to access and edit the nickname of their favourited carparks.
    - 1.2. The editing interface must display a list of favourited carparks.
        - 1.2.1. The interface must include the current nicknames.
        - 1.2.2. The interface must provide an option to edit each carparks.
    - 1.3. The system must allow users to modify the existing nickname.
        - 1.3.1. The system must offer a "Rename" button for each favourited carpark.
        - 1.3.2. The system must provide a text field within the interface for users to input the new nickname for the favourite carpark once "Rename" is pressed.
            - 1.3.2.1. The system must prompt the user to enter a nickname for the carpark.
            - 1.3.2.2. The system must provide a button "Save" to update the nickname.

1.3.2.3. The system must provide a button “Cancel” to return to previous state.

1.4. The system must verify the text field before updating the nickname of the carpark.

1.4.1. The system must check the text field is not empty.

1.4.2. The system must check the nickname has not been used before.

1.5. The system must update the nickname upon successful verification.

Use Case ID:	13		
Use Case Name:	Change Password		
Created By:	Shun Ping	Last Updated By:	
Date Created:	01/02/2024	Date Last Updated:	

Actor:	User, System
Description:	User change his/her current password
Preconditions:	1) User has clicked on "View User Profile" 2) User has clicked on "Change Password"
Postconditions:	Password of user in database has been updated with the new value.
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1) User clicks on "Change Password" on the profile page. 2) System prompts the user to enter current password and new password. 3) User enters the information and clicks on "Save". 4) System checks whether the information satisfy the requirements and are valid. 5) System overwrites and stores the new password into the database. 6) App will display the "Password successfully changed".
Alternative Flows:	13-AF-S3 Current password incorrect 1) System displays "Current password do not match database". 2) System returns to Step 2. 13-AF-S3 If new password does not meet the requirements 1) System displays "Password does not meet requirements". 2) System returns to Step 2.
Exceptions:	NIL
Includes:	NIL
Special Requirements:	NIL
Assumptions:	User has internet connection
Notes and Issues:	NIL

1. User must be able to change password.
    - 1.1. The system must provide an interface for the user to change their password.
    - 1.2. The interface must display text fields for the user to enter the necessary information
        - 1.2.1. The text fields must consist of Current Password.
        - 1.2.2. The text fields must consist of New Password.
        - 1.2.3. The text fields must consist of Confirm New Password.
    - 1.3. The user must fill in all of the text fields (Current Password, New Password, Confirm New Password) before clicking the 'Save' button
        - 1.3.1. The system must verify the fields when the user clicks the ''Save' button.
    - 1.4. The system must verify the information filled in by the user before changing the password

1.4.1. The current password entered must match the password on the database.
1.4.2. The new password must meet the requirements.
1.4.2.1. Old password must match user's current password
1.4.2.2. The new password must consist of at least 1 uppercase character.
1.4.2.3. The new password must consist of at least 1 lowercase character.
1.4.2.4. The new password must consist of at least 8 characters.
1.4.2.5. The new password must consist of at least 1 number.
1.4.2.6. The new password must be different from the old password
1.4.2.7. The confirm new password must match new password
1.4.3. The system must provide error messages describing the reason the user's password is rejected.
1.5. The system must change the password upon successful verification.

NON-FUNCTIONAL REQUIREMENTS

Usability	1.	At least 80% of first time users must be able to enter a simple search query for a car park within 2 minutes after logging in.
	2.	At least 80% of new users should not take longer than 5 minutes to register for an account.
Reliability	1.	The app should be available and responsive, ensuring >99% uptime.
	2.	It should handle user inputs and requests reliably, with an error rate and crash probability not exceeding 1% under normal operating conditions.
	3.	Data synchronization between the app and server should be reliable and consistent, demonstrating a reliability rate of at least 99% across all synchronized datasets.
Supportability	1.	The app should be compatible with a range of mobile devices running on android, being compatible with at least 80% of android device models released in the past 5 years.
	2.	A comprehensive help and support system should be available within the app, providing guidance and assistance to users. At least 60% of users must be able to resolve their issue within 10 minutes of consulting the help and support system.
Scalability	1.	The app should be designed to handle a growing user base and increased traffic, with the capacity to accommodate a minimum of 10% annual user growth.
	2.	The app should perform efficiently even as the number of parked vehicles and users increases, demonstrating a response time of under 1000ms for standard user interactions, even with a concurrent user increase of up to 50% from current baselines.
Security	1.	User accounts should maintain a high level of security to prevent unauthorized access. Security testing should confirm that user accounts are resistant to common hacking techniques.
Performance	1.	The app should be responsive to user actions, maintaining a response time of under 1000ms for standard user interactions under normal operating conditions.
Maintainability	1.	Documentation for the app's codebase must be comprehensive, providing clear guidelines for code structure, dependencies, and system components.
	2.	The system should be modular, allowing for easier updates and modifications without impacting the entire application.
	3.	A version control system must be in place to track changes.