Use Cases

for

ParkNow

Version 1.0 approved

Prepared by Team Overflow

Nanyang Technological University

Sept 7, 2023

Revision History

Name	Date	Reason For Changes	Version
Wang Yangming	Sept 7	First Draft	1.0

Use Case ID:	UC001		
Use Case Name:	Register		
Created By:	WANG YANGMING	Last Updated By:	
Date Created:	4th Sept	Date Last Updated:	

Actor:	User
Description:	This use case describes the process of a user registering for an account on the system, including the validation of user-provided information and the creation of a user account.
Preconditions:	1. The user has access to the system's registration functionality. 2. The user is not currently logged into the system.
Postconditions:	The user has a registered account in the system and is logged in.
Priority:	High
Frequency of Use:	This use case is typically performed once per user.
Flow of Events:	1. The user accesses the registration page. 2. The system displays the registration form, including the following text fields: 2.1. Username field 2.2. Email address field 2.3. Password field 3. The user enters their desired login details: 3.1. The user enters a username (8 to 32 alphanumeric characters inclusive). 3.2. The user enters a valid and unique email address. 3.3. The user enters a password (at least 8 characters, with at least 1 uppercase character) 4. The user clicks the “Register” button. 5. The system verified the entered information: 5.1. Ensure all text fields are filled. 5.2. Check the uniqueness and validity of the username. 5.3. Verify that the email address is in the correct format and not previously registered 5.4. Send an OTP (One-Time Password) to the user’s provided email address for verification. 6. If the verification fails for any reason (e.g., fields not filled, non-unique username, incorrect email format), the system provides error messages describing the reasons for rejection. 7. If there are errors: 7.1. The system empties all text fields. 7.2. The user corrects the errors and repeats steps 3 to 6. 8. If the verification succeeds, the system proceeds to create the user account: 8.1. The system records the user’s account in the database, including username or email and password. 9. The system logs the user into the main page of the system
Alternative Flows:	AF-S5.2. If the chosen username is not unique, the system must prompt the user to choose a different username.

	AF-S5.3. If the provided email address has been previously registered, the system must prompt the user to use a different email address. AF-S5.4: If the user does not receive the OTP, the system should provide an option to resend the OTP. If the user fails to verify their OTP within a specified time limit, the registration process may be canceled or require reinitiation.
Exceptions:	EX1. Technical issues such as database connection problems or email delivery failures may occur during the registration process. Proper error handling and notifications should be implemented.
Includes:	Sending an OTP to the user's email address for verification may involve integrating with an email service.
Special Requirements:	1. The system should implement strong security measures to protect user data including password hashing and encryption of sensitive information. 2. Compliance with data privacy regulations and policies is necessary, including obtaining user consent for data processing.
Assumptions:	1. The user has access to a valid email address for OTP verification. 2. The system follows best practices for data security and user authentication.
Notes and Issues:	It is essential to design a user-friendly interface that guides users through the registration process and provides clear error messages to facilitate a smooth registration experience.

Use Case ID:	UC002		
Use Case Name:	Login		
Created By:	WANG YANGMING	Last Updated By:	
Date Created:	4th Sept	Date Last Updated:	

Actor:	User
Description:	This use case describes the process of a user logging into the system using their login credentials, including the verification of the provided information and the ability to log out.
Preconditions:	1. The user has an existing account in the system. 2. The user has access to the system's login functionality. 3. The user is not currently logged into the system.
Postconditions:	1. The user is logged into the system and has access to the main page. 2. The user can log out of their account at any time.
Priority:	High
Frequency of Use:	This use case is typically performed each time a user wants to access the system.
Flow of Events:	1. The user accesses the login page. 2. The system displays the login form, including the following text fields:

	2.1. User ID field (accepting either a username or an email address) 2.2. Password field 3. The user enters their login credentials: 3.1. The user enters either a username or an email address into the User ID field 3.2. The user enters their password. 4. The user clicks the "Log In" button. 5. The system verifies the entered information: 5.1. Ensure both text fields are filled. 5.2. Determine whether the User ID entered corresponds to an existing username or email address in the system's database. 5.2.1. If the User ID is a username, check if it exists in the database. 5.2.2. If the User ID is an email address, check if it exists in the database 5.3. Verify that the entered password matches the password associated with the User ID in the system's database. 6. If the verification fails for any reason (e.g., fields not filled, User ID not found, incorrect password), the system provides error messages describing the reasons for rejection. 7. If there are errors: 7.1. The system allows the user to correct the errors and repeat steps 3 to 6. 8. If the verification succeeds, the system logs the user into the main page of the system. 9. The user has the option to log out of the account at any time after logging in.
Alternative Flows:	AF-S3.2. If the user forgets their password, there should be a "Forgot Password" functionality that allows them to reset their password through email verification. AF-S5.2. If the User ID is not found in the database, the system must provide an error message indicating that the User ID is invalid.
Exceptions:	EX1. Technical issues such as database connection problems may occur during the login process. Proper error handling and notifications should be implemented.
Includes:	The "Forgot Password" functionality mentioned in the alternative flows may involve integrating with an email service for password reset instructions.
Special Requirements:	1. The system should implement strong security measures to protect user data, including password hashing and encryption of sensitive information. 2. Compliance with data privacy regulations and policies is necessary, including protecting user login information.
Assumptions:	1. The user has already registered for an account in the system 2. The system follows best practices for data security and user authentication

Notes and Issues:	1. It is essential to design a user-friendly interface that guides users through the login process and provides clear error messages to facilitate a smooth login experience. 2. The option to log out should be easily accessible to the user within the system's interface

Use Case ID:	UC003		
Use Case Name:	Search for Nearest Car Parks		
Created By:	WANG YANGMING	Last Updated By:	
Date Created:	4th Sept	Date Last Updated:	

Actor:	User, Google Maps API
Description:	This use case describes the functionality of the application's home page, focusing on the search bar that allows users to find the nearest car parks based on their specified location or current location. The user case also covered the use of the Google Maps API for location identification and the display of a list of nearby car parks.
Preconditions:	1. The user has accessed the application's home page. 2. The user has the necessary permissions to access location services if selecting the current location as input.
Postconditions:	1. The user receives a list of up to 10 nearest car parks within a 5-kilometer radius based on the provided or current location. 2. The user can interact with the list of car parks or perform additional actions.
Priority:	High
Frequency of Use:	This use case is frequently used whenever a user wants to find nearby car parks.
Flow of Events:	1. The user accesses the application's home page. 2. The home page displays a search bar. 3. The user can input a location into the search bar: 3.1. The user can either choose to specify a location manually or select "current location" 3.2. If the user selects "current location", 3.2.1. If the user is using accessing the function for the first time, the system requests permission to access the user's device information. 3.2.2. If the user has previously denied permission to the system to access device information, the system requests permission to access the user's device information 4. The input location, whether manually specified or current, is sent to the Google Maps API for identification. 5. The system retrieves a list of up to 10 nearest car parks within a 5-kilometer radius of the specified or current location from the Google Maps API.

	6. The list of car parks is sorted from shortest to longest linear distance from the user's specified or current location. 7. The system displays the sorted list of car parks to the user.
Alternative Flows:	AF-S3.2. If the user denies permission to access their current location, they can manually enter a location in the search bar. AF-S4. If the input location is not identifiable by Google Maps API, the system must provide an error message indicating that the input location is invalid.
Exceptions:	EX1. Technical issues or errors related to the Google Maps API or location services may occur during location identification and retrieval. The system should handle such errors and provide appropriate feedback to the user.
Includes:	Utilization of the Google Maps API for location identification and distance calculations.
Special Requirements:	1. The application must have access to the user's location services if the "current location" option is selected. 2. The system should provide a user-friendly interface with clear instructions on how to use the search bar and what information is required.
Assumptions:	1. The Google Maps API is available and properly configured for location identification. 2. The application can access the user's device location if required. 3. Users are generally familiar with using search bars for location-based queries.
Notes and Issues:	1. To enhance user experience, consider providing additional information about each car park in the list, such as availability, pricing, and directions. 2. Implementing a user-friendly and intuitive search bar is crucial to ensure users can easily find car parks based on their preferences.

Use Case ID:	UC004		
Use Case Name:	Query Car Park Details		
Created By:	WANG YANGMING	Last Updated By:	
Date Created:	4th Sept	Date Last Updated:	

Actor:	User, Carpark Availability API
Description:	This use case outlines the functionality available to members (registered users) when they select a car park from the search results list. Members can access real-time car park details, including information about available parking spaces, parking costs, and recommended travel routes to the selected car park.
Preconditions:	1. The user is logged into their member account. 2. The user has performed a search for car parks and received a list of search results.

Postconditions:	The member has access to detailed information about the selected car park, including available parking spaces, parking costs, and recommended travel routes.
Priority:	High
Frequency of Use:	This use case is used whenever a member wants to obtain detailed information about a specific car park from the search results.
Flow of Events:	1. The user performed a search for car parks and received a list of search results. 2. The user selects a specific car park from the search results list. 3. The system accesses real-time car park details from the relevant API for the selected car park. 4. The system displays the following information about the selected car park: 4.1. The number of available car parking spaces in the selected car park. 4.2. The number of available motorcycle parking spaces in the selected car park. 4.3. Line chart of parking costs within the next six months for the selected car park. 5. The system displays a recommended route to the selected car park: 5.1. The system calculates and reports the estimated travel time by car to reach the selected car park from the member's current or specified location.
Alternative Flows:	AF-S1. If the member has not performed a search for car parks: 1. If the member is selecting car parks from the favorites folder, continue the flow from step 3. 2. If the member has not performed a search for car parks and is not selecting car parks from the favorites folder, they must initiate a search before selecting a specific car park.
Exceptions:	EX1. Technical issues or errors related to accessing real-time car park details from the API may occur. The system should handle such errors and provide appropriate feedback to the member.
Includes:	Integration with external APIs for real-time car park information and route calculations.
Special Requirements:	1. The system should display the most up-to-date information about available parking spaces and parking costs. 2. Integration with a navigation service may be required to provide recommended travel routes to the selected car park.
Assumptions:	1. Members have already performed a search for car parks and have a list of search results to choose from. 2. Real-time data from external sources (such as parking space availability) is available and reliable.
Notes and Issues:	Providing real-time information and navigation assistance can significantly enhance the user experience and help members make informed decisions about parking.

Use Case ID:	UC005		
Use Case Name:	Add Car Park to Favorites		
Created By:	WANG YANGMING	Last Updated By:	
Date Created:	4th Sept	Date Last Updated:	

Actor:	User
Description:	This use case outlines the functionality available to members (registered users) to add car parks to their favorites folder and access detailed information about car parks stored in their favorites. Members can organize and sort their favorite car parks by various criteria and view real-time information about these car parks.
Preconditions:	1. The user is logged into their member account. 2. The user has performed a search for car parks and received a list of search results.
Postconditions:	1. The member can view and access their favorite car parks, sorted and organized according to their preferences. 2. The member can access real-time car park details for the selected car park from their favorites folder.
Priority:	Medium
Frequency of Use:	This use case is used whenever a member wants to manage their favorite car parks or access detailed information about them.
Flow of Events:	1. The user clicks the "Add to Favorites" button of the specific car park the user wants to add. 2. The user navigates to their favorites folder within the application. 3. The system displays a list of car parks that the user has added to their favorites. 4. The user can perform the following actions within the favorites folder: 4.1. Sort the car parks by multiple orders, including: 4.1.1. Sorting by time of addition. 4.1.2. Sorting alphabetically. 4.1.3. Sorting by increasing or decreasing order based on specific criteria 5. The user can select a specific car park from their favorites folder. 6. The system accesses real-time car park details from the relevant API for the selected car park. 7. The system displays the following information about the selected car park: 7.1. The number of available car parking spaces in the selected car park. 7.2. The number of available motorcycle parking spaces in the selected car park. 7.3. Parking costs within the next six months for the selected car park.
Alternative Flows:	AF1. If the car park has been added to the favorites folder. Clicking again will remove the car park from the user's favorites folder.
Exceptions:	EX1. Technical issues or errors related to accessing real-time car park details from the API may occur. The system should handle such errors and provide appropriate feedback to the member.
Includes:	Integration with external APIs for real-time car park information.

Special Requirements:	1. The system should display the most up-to-date information about available parking spaces and parking costs. 2. A user-friendly interface should allow members to easily manage and sort their favorite car parks.
Assumptions:	1. Members have previously added car parks to their favorites folder. 2. Real-time data from external sources (such as parking space availability) is available and reliable.
Notes and Issues:	Providing members with the ability to organize and access detailed information about their favorite car parks can enhance the user experience and help members plan their parking more effectively.

Use Case ID:	UC006		
Use Case Name:	View Profile Page		
Created By:	WANG YANGMING	Last Updated By:	
Date Created:	13th Sept	Date Last Updated:	

Actor:	User
Description:	This use case describes the functionality related to the profile page in the system, where registered users can view and manage their personal information and settings.
Preconditions:	1. The user must be logged into the system. 2. The user must have a registered account.
Postconditions:	1. The user can view their profile information. 2. The user can make changes to their profile picture, email address, and password. 3. The user can log out from the profile page.
Priority:	High
Frequency of Use:	Regularly
Flow of Events:	1. The user logs into the system. 2. The user navigates to the profile page. 3. On the profile page: 3.1. The system displays the user's current profile picture (if available) 3.2. The system displays the user's username. 3.3. The system displays the user's email address. 3.4. The user has the option to change their profile picture. 3.5. The user has the option to change their password. 3.6. The user has the option to change their password. 3.7. The user has the option to log out. 4. If the user chooses to change their profile picture. 4.1. The user uploads a new profile picture. 4.2. The system updates the profile picture.

	4.3. The system confirms the successful update. 5. If the user chooses to change their email address: 5.1. The user enters a new email address. 5.2. The system sends a verification email to the new address. 5.3. The user receives the verification email. 5.4. The user clicks the verification link to confirm the new email address. 5.5. The system updates the email address. 5.6. The system confirms the successful update. 6. If the user chooses to change their password: 6.1. The user enters the current password and a new password. 6.2. The system validates the new password. 6.3. The system updates the password. 6.4. The system confirms the successful password change. 7. If the user chooses to log out: 7.1. The user is logged out of the system.
Alternative Flows:	AF-S3.1. If the user does not have a profile picture, the system displays a default profile picture. AF-S5.2. If the user's chosen new email address is already in use by another account, the system should prompt the user to choose a different email address. AF-S6.1. If the user enters an incorrect current password when changing their password, the system should notify the user and not make the change.
Exceptions:	EX1. If there are technical issues or errors during any of the processes (e.g., uploading a profile picture, sending a verification email), the system should display an error message and allow the user to retry.
Includes:	Resending a verification email is included in the process of changing the email address.
Special Requirements:	1. The system should enforce password complexity rules for setting a new password. 2. The system should provide clear and user-friendly error messages in case of issues.
Assumptions:	The user is familiar with their current login credentials (username and password).
Notes and Issues:	User experience and security considerations should be taken into account when implementing these functionalities.

Use Case ID:	UC007		
Use Case Name:	Share Car Parks		
Created By:	WANG YANGMING	Last Updated By:	
Date Created:	13th Sept	Date Last Updated:	

Actor:	User

Description:	This use case describes the functionality for a registered user to share information about car parks using various methods, including copying location details and generating links to navigation apps.
Preconditions:	1. The user must be logged into the system. 2. The user must have access to information about car parks.
Postconditions:	1. The user can share car park information using different methods. 2. The system copies the selected information or generates links to navigation apps.
Priority:	Medium
Frequency of Use:	Occasional
Flow of Events:	1. The user logs into the system. 2. The user accesses information about a specific car park. 3. The user taps the “Share” button. 4. The system presents a submenu with options to choose the type of information to generate or copy: 4.1. The user can choose to copy the location of the car park in plain text to the clipboard. 4.2. The user can choose to generate a link to Google Maps and copy it to the clipboard. 4.3. The user can choose to generate a link to Waze and copy it to the clipboard. 4.4. The user can choose to generate a link to Apple Maps and copy it to the clipboard. 4.5. The user can choose to generate a link to ParkNow and copy it to the clipboard. 5. If the user selects the option to generate a link to ParkNow: 5.1. If the invited user has already logged in to ParkNow, the user can access the specific car park details page. 6. The system completes the selected action, and the user can now share the information using the chosen method.
Alternative Flows:	AF-S5.1. If the user has not logged into the system, they must log in before accessing car park information.
Exceptions:	EX1. If there are technical issues or errors during any of the processes (e.g., generating links), the system should display an error message and allow the user to retry.
Includes:	Generating links and copying information to the clipboard are included as steps in the process of sharing car park information.
Special Requirements:	The system should handle different types of links and information generation based on the user’s selection.
Assumptions:	1. The user has access to car park information within the system. 2. The user is familiar with how to use the clipboard on their device.
Notes and Issues:	1. Ensure that the generated links are formatted correctly and functional for navigation apps. 2. Consider providing a confirmation message to the user after the selected action is completed successfully.

Use Case ID:	UC008		
Use Case Name:	Share Drives		
Created By:	WANG YANGMING	Last Updated By:	
Date Created:	13th Sept	Date Last Updated:	

Actor:	User
Description:	This use case outlines the functionality for a registered user to share drive information, specifically generating a link to ParkNow that allows other users to monitor the user's progress on their current route and query details about car parks.
Preconditions:	1. The user must be logged into the system 2. The user must be actively using navigation or driving within the system. 3. The user must have access to a route and car park information within the system.
Postconditions:	1. The user can share drive information with other users by generating a link. 2. Other users who have access to the link can monitor the user's route progress and query car park details.
Priority:	Medium
Frequency of Use:	Occasional
Flow of Events:	1. The user logs into the system. 2. The user starts a navigation session or is actively driving within the system 3. The user taps the "Share Drive Information" button. 4. The system generates a unique link to ParkNow and copies it to the clipboard. 5. The user can share this link with other users. 6. Other users who receive the link can access it. 7. When other users access the link: 7.1. They can view the user's progress on their current route. 7.2. They can interact with the map and select car parks to query their details.
Alternative Flows:	AF-S2. If the user is not actively using navigation or driving, they cannot access this feature. AF-S3. If the user is not logged into the system, they must log in before accessing the "Share Drive Information" button. AF-S5. The user may choose to share the link through various communication channels (e.g., messaging apps, email).
Exceptions:	EX1. If there are technical issues or errors during the link generation process, the system should display an error message and allow the user to retry.
Includes:	Generating a link to ParkNow and copying it to the clipboard is included as a step in the process of sharing drive information.
Special Requirements:	1. The generated link should provide real-time or near-real-time updates on the user's route progress. 2. The map and car park details should be interactive for users who access the link. 3. Privacy settings and permissions may be needed to control who can access the drive information link.

Assumptions:	1. The user has access to a navigation or driving feature within the system. 2. Other users who receive the link have access to the same map and car park information within the system.
Notes and Issues:	1. Consider implementing security measures to protect the privacy of the user sharing their drive information. 2. Ensure that the generated link is functional and provides a seamless experience for users accessing it. 3. Consider providing an option for the user to revoke or expire the shared drive information link.

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 labeled 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 Determineds) that must be resolved. Identify who will resolve each issue, the due date, and what the resolution ultimately is.