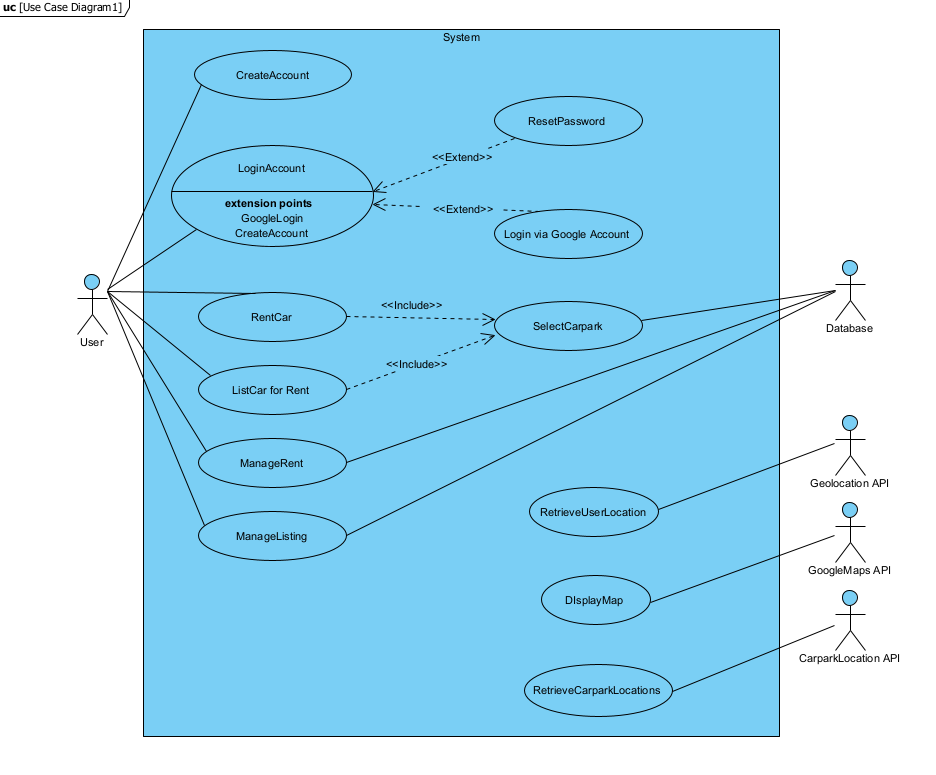
**Complete Use Case model (use case diagram + use case description)**

****

**Use Cases**

**for**

**<SC2006 Project>**

**Version 2.0 approved**

**Prepared by <A27>**

**<Nanyang Technological University>**

**<20 March 2023>**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| A27 | 29 Jan 2023 | - | 1.0 |
| A27 | 12 Feb 2023 | Refine the use case descriptions | 2.0 |
| A27 | 19 March 2023 | Finalise the use case descriptions | 3.0 |

# **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.

# **Use Case Identification**

## **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.

## **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.

## **Use Case History**

### **Created By**

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

### **Date Created**

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

### **Last Updated By**

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

### **Date Last Updated**

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

# **Use Case Definition**

## **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.

## **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.

## **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.

## **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.

## **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.

## **Frequency of Use**

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

## **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.

## **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 flow using “AF-S” followed by the step number where the alternative flow diverts from the normal flow. For example, if an alternative flow differs from the normal flow starting from step 5, then the alternative flow should be titled as “AF-S5”.

## **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 “EX” to indicate “Exception”, followed by the exception number for that use case. For example, if a use case has 2 exceptions, the first one should be titled as “EX1” and the second one titled as “EX2”.

## **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.

## **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.

## **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.

## **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.

| Use Case ID: | 1.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | CreateAccount | | |
| Created By: | Aki | Last Updated By: | Kavithran |
| Date Created: | 29 Jan 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | User |
| --- | --- |
| Description: | Allow user to create a new account. |
| Preconditions: | 1. An internet connection must be established 2. User selects to create a new account. 3. User’s account must not exist in the database. |
| Postconditions: | 1. New user account is successfully registered in the system database. 2. The user can now log in and start using the services offered by our application. |
| Priority: | Medium |
| Frequency of Use: | Medium - used only by first time users. |
| Flow of Events: | 1. User taps on ‘Create an Account’. 2. User will be redirected to the ‘Create an Account’ page with 5 input fields (First name, Last name, Email address, Password and Confirm Password). 3. User clicks the ‘Register Account’ button. 4. System adds the new user record to the database. 5. System displays the success message ‘Registration Successful! Please proceed to login’. 6. User clicks on the ‘Login’ button to return to login page. |
| Alternative Flows: | AF-S3: User leaves any of the 5 input fields empty.   1. System displays an error message, ‘Field cannot be empty’. 2. User cannot proceed and System returns to step 2.   AF-S3: User enters invalid email address   1. System displays an error message, ‘Invalid email’. 2. User cannot proceed and System returns to step 2.   AF-S3: System detects a mismatch between the ‘Password’ and ‘Repeat Password’.   1. System displays an error message, ‘Password not match’. 2. User cannot proceed and System returns to step 2.   AF-S3: System detects a weak password which is less than 8 characters.   1. System displays an error message, ‘Weak password’. 2. User cannot proceed and System returns to step 2. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 2.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | LoginAccount | | |
| Created By: | Kavithran | Last Updated By: | Kavithran |
| Date Created: | 29 Jan 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | User |
| --- | --- |
| Description: | Allow user to login into the application using username and password. |
| Preconditions: | 1. An internet connection must be established 2. User’s account must exist in the database. |
| Postconditions: | 1. User successfully log in into the application. |
| Priority: | High |
| Frequency of Use: | High |
| Flow of Events: | 1. User enters username and password in the appropriate input fields. 2. User selects the ‘Login’ button. 3. System checks the database for an existing account of user. 4. System allows user to login successfully |
| Alternative Flows: | AF-S1: User taps login via Google account.   1. System returns to step 4.   AF-S1: User taps ‘Forgot Password’.   1. User enters email address into the input field and taps on the ‘Reset Password’ button. 2. An email is sent to the user with instructions to reset the password. 3. User opens a link in the email and System prompts the user to enter a new password. 4. User can return to step 1.   AF-S2: User enters wrong username or password.   1. System displays ‘User not found’ or ‘Wrong password’ respectively. 2. System returns to step 1. |
| Exceptions: | EX-AF-S1: User not logged in their Google account.   1. System prompts user to login to their Google account. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: | 1. Need to address empty input fields when login. |

| Use Case ID: | 2.1 | | |
| --- | --- | --- | --- |
| Use Case Name: | ResetPassword | | |
| Created By: | Kavithran | Last Updated By: | Kavithran |
| Date Created: | 19 March 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | User |
| --- | --- |
| Description: | Allow users to reset their passwords. |
| Preconditions: | 1. An internet connection must be established |
| Postconditions: | 1. User successfully resets to a new password and login into the application. |
| Priority: | High |
| Frequency of Use: | Medium |
| Flow of Events: | 1. User taps on ‘Forgot Password’. 2. User enters email address into the input field. 3. User taps on the ‘Reset Password’ button. 4. An email is sent to the user with instructions to reset the password. 5. User opens a link in the email and System prompts the user to enter a new password. 6. System checks if the new password is valid (not less than 8 characters). 7. System stores the new password in the database and displays a success message. 8. System prompts the user to return to the login page. |
| Alternative Flows: | AF-S3: User enters invalid email or email not found in database.   1. System displays an error message ‘Invalid email’. 2. System returns to step 2. |
| Exceptions: | EX-S4: User attempts to re-open the link sent to email after using it once to reset password.   1. System displays an error message, ‘Your request to reset your password has expired or the link has already been used’. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 2.2 | | |
| --- | --- | --- | --- |
| Use Case Name: | Login via Google Account | | |
| Created By: | Kavithran | Last Updated By: | Kavithran |
| Date Created: | 19 March 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | User |
| --- | --- |
| Description: | Allow users to login via Google account. |
| Preconditions: | 1. An internet connection must be established |
| Postconditions: | 1. User successfully login into the application using their Google account. |
| Priority: | High |
| Frequency of Use: | Medium |
| Flow of Events: | 1. User taps login via Google account. 2. System stores user record to the database. 3. System allows user to login successfully. |
| Alternative Flows: | AF-S1: User not logged in to their Google account.   1. System prompts user to log in to their Google account. 2. User logs in to the Google account. 3. System returns to step 2. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 3.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | RentCar | | |
| Created By: | Kavithran | Last Updated By: | Kavithran |
| Date Created: | 19 March 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | User, Database |
| --- | --- |
| Description: | A user can rent a car that is listed for rent. |
| Preconditions: | 1. User has a verified account in the system. |
| Postconditions: | 1. User successfully rents a car from the car park. (AND) 2. User successfully returns the rented car to the car park. |
| Priority: | High |
| Frequency of Use: | High |
| Flow of Events: | 1. User logs in to the application. 2. System displays the map with carpark locations available. 3. User picks a preferred carpark location on the map. 4. System displays a list of available cars for rent at the selected car park. 5. User selects preferred car and timing for rent. 6. User taps on the“Rent” button. 7. System sends a rental request to the car’s owner. 8. Car owner accepts the user’s rental request and the user can rent the car. 9. User returns the car back to the same car park after renting it. 10. User taps on ‘Complete Rental’ in the application upon returning the rented car. |
| Alternative Flows: | AF-S4: If there are no cars available for rent at the selected car park.   1. The system displays “Cars unavailable. Please try another car park”. 2. System returns to step 2.   AF-S7: Car owner rejects the user’s rent request.   1. The system sends a notification to the user, “Your request has been rejected”. 2. System returns to step 2. |
| Exceptions: |  |
| Includes: | SelectCarpark |
| Special Requirements: |  |
| Assumptions: | 1. Users have a valid driving license. 2. User returns the rented car to the same carpark. 3. Payment is handled by renter and owner, outside the application. |
| Notes and Issues: | 1. Need to address what happens if renter does not tap ‘Complete Rental’ within the timeframe booked. |

| Use Case ID: | 4.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | ManageRent | | |
| Created By: | Wai Man | Last Updated By: | Kavithran |
| Date Created: | 19 March 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | User (Renter and Owner) and Database |
| --- | --- |
| Description: | This use case handles actions including Cancel Rental and Complete Rental. |
| Preconditions: | 1. User must have an account and be able to login into the system successfully 2. If a renter wants to cancel rental or complete rental, he/she must have at least one accepted rental request in the system |
| Postconditions: | 1. A message (notification) must be shown to both renter and owner if the rental request has been canceled or completed successfully. 2. Listing status must be updated    1. Rented    2. Pending (wait for approve from owner)    3. Available |
| Priority: | High |
| Frequency of Use: | High |
| Flow of Events: | 1. User login into the system using his login credentials 2. User select a car park to view the cars available for rent 3. User selects a car he wants to view the details of. 4. User clicks the ‘Rent’ button. 5. The status of the post will be changed to ‘Pending’. 6. The car owner will receive a notification for the booking request. 7. The car owner accepts the request by clicking the ‘Accept’ button. 8. The listing post will be hidden from the public. 9. A ‘accepted’ notification will be sent to both car owner and renter. 10. The renter can complete the booking by clicking the ‘Complete’ button. 11. A notification will be sent to the car owner that the booking has been completed. |
| Alternative Flows: | AF-S7: The car owner wants to reject booking requests.   1. The car owner rejects booking requests by clicking the ‘Reject’ button. 2. A notification will be sent to the renter that the booking request has been rejected.   AF-S10: The renter wants to cancel the renting request.   1. The renter can cancel the booking by clicking the ‘Cancel’ button. 2. A notification will be sent to the car owner that the booking request has been canceled. |
| Exceptions: | EX1: If an error happens when canceling or completing a booking, an error message must be shown to notify users. |
| Includes: |  |
| Special Requirements: | Both renter and car owner must own valid driving license |
| Assumptions: | 1. All necessary information (age, driving license, etc.) has been verified when user creating an account |
| Notes and Issues: |  |

| Use Case ID: | 5.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | ListCar for Rent | | |
| Created By: | Kavithran | Last Updated By: | Kavithran |
| Date Created: | 19 March 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | User, Database |
| --- | --- |
| Description: | This use case will cover the listing of a car for rental in the application. |
| Preconditions: | 1. User must have an account and be able to login into the system successfully |
| Postconditions: | The listing will be created. |
| Priority: | High |
| Frequency of Use: | Medium |
| Flow of Events: | 1. The user logs into the application. 2. System displays a map with the carparks available. 3. User select a car park and clicks on ‘Add your car’. 4. System displays a page for the user to upload information about his/her car. 5. User clicks on ‘Add car’ and the car information is posted onto the carpark’s car listing. |
| Alternative Flows: | AF-S4: If the user has not provided all the required information   1. The application displays the message “Please enter all the required information!” 2. The user returns to step 4 in normal flow. |
| Exceptions: | - |
| Includes: | SelectCarpark |
| Special Requirements: | - |
| Assumptions: | 1. The user has all the necessary information and photos to complete the listing. |
| Notes and Issues: | - |

| Use Case ID: | 6.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | ManageListing | | |
| Created By: | Woon Keng | Last Updated By: | Wai Man |
| Date Created: | 29 Jan 2023 | Date Last Updated: | 9 Feb 2023 |

| Actor: | User, Database |
| --- | --- |
| Description: | This use case will cover all the aspects of listings in the application, including the addition, deletion and editing of listings by users. |
| Preconditions: | 1. The user has registered an account and logged in to the application. 2. The user must have at least one listing if he wants to edit or delete |
| Postconditions: | The listing will be created/updated/deleted |
| Priority: | High |
| Frequency of Use: | Medium |
| Flow of Events: | 1. The user navigates to his Profile page in the app. 2. All of the user’s active listings will be shown under the “Listings” tab 3. Click the “+" button at the bottom center 4. The user needs to fill up the car information form 5. Click the “Post” button to submit listing 6. The listing will then be shown in the car renting list |
| Alternative Flows: | AF-S4: If the user has not provided all the required information   1. The application displays the message “Please enter all the required information!” 2. The user returns to step 2) in normal flow   AF-S3: If the user wants to update the existing listing   1. Select the listing that he wants to edit 2. Click the “Edit” button 3. The user makes the edits he wants in the detail fields. 4. The user clicks on the “Save” button when he is finished. 5. The application registers the changes, closes the edited listing and returns to the listing management page.   AF-S2: If the user wants to delete an existing listing   1. The user selects the listing that he wants to delete 2. The application displays the full listing with its details along with a “Delete listing” button at the bottom of the page 3. The user clicks on the “Delete listing” button. 4. The application asks the user to confirm his action with a warning that the action is not reversible. 5. The user clicks “Confirm”. |
| Exceptions: |  |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: | 1. The user has all the necessary information and photos to complete the listing |
| Notes and Issues: |  |

| Use Case ID: | 7.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | SelectCarpark | | |
| Created By: | Kavithran | Last Updated By: | Kavithran |
| Date Created: | 29 Jan 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | User, Database |
| --- | --- |
| Description: | Retrieve the details of picked carpark location. |
| Preconditions: | 1. Mobile location services must be on. 2. User has logged in successfully. |
| Postconditions: | 1. System retrieves the details of carpark selected. |
| Priority: | High |
| Frequency of Use: | High |
| Flow of Events: | 1. System displays a map with carpark locations and user location marked. 2. User picks a carpark location. 3. System prompts user to ‘view car list’ or ‘add car’. 4. System queries database and displays appropriate page upon option selection by user. |
| Alternative Flows: | - |
| Exceptions: | EX1: Mobile phone’s location not turned on.   1. System will prompt the user to turn on location services. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: | 1. Geolocation API will return the most accurate user’s location. |
| Notes and Issues: |  |

| Use Case ID: | 8.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | RetrieveUserLocation | | |
| Created By: | Kavithran | Last Updated By: | Kavithran |
| Date Created: | 29 Jan 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | Geolocation API |
| --- | --- |
| Description: | Retrieves the user's location through the device. |
| Preconditions: | 1. Mobile’s location services must be on. 2. Mobile connected to internet services. |
| Postconditions: | 1. Geolocation API returns location of user |
| Priority: | - |
| Frequency of Use: | - |
| Flow of Events: | 1. User logs in to the system. 2. The system gets user locations through a Geolocation API. 3. System displays user location with a marker on the map. |
| Alternative Flows: | - |
| Exceptions: | EX1: Mobile phone’s location not turned on.   1. System will prompt the user to turn on location services. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: | 1. Geolocation API will return the most accurate user locations of the user. |
| Notes and Issues: |  |

| Use Case ID: | 9.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | DisplayMap | | |
| Created By: | Aki | Last Updated By: | Wai Man |
| Date Created: | 29 Jan 2023 | Date Last Updated: | 9 Feb 2023 |

| Actor: | Google Maps’ API |
| --- | --- |
| Description: | Display the selected location and nearby car park on map |
| Preconditions: | 1. An internet connection must be established 2. Carpark information has been retrieved successfully. 3. User location has been retrieved successfully. |
| Postconditions: | 1. User can view carparks and his/her location on the map. |
| Priority: | High. |
| Frequency of Use: | High. |
| Flow of Events: | 1. User has logged in to the system. 2. Google Maps’ API is called. 3. The user location will be located on a map with a marker. 4. System displays the carparks on the map with markers 5. The user is able to pan around and zoom in or out of the map to view the location of the car parks. 6. The user will be able to view information about the selected car park after tapping on the marker. |
| Alternative Flows: |  |
| Exceptions: | EX1: Mobile phone’s location not turned on.   1. System will prompt the user to turn on location services. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

| Use Case ID: | 10.0 | | |
| --- | --- | --- | --- |
| Use Case Name: | RetrieveCarparkLocation | | |
| Created By: | Kavithran | Last Updated By: | Kavithran |
| Date Created: | 29 Jan 2023 | Date Last Updated: | 19 March 2023 |

| Actor: | User, Carpark-Location API |
| --- | --- |
| Description: | Retrieves a list of car parks near selected location. |
| Preconditions: | 1. Mobile’s location services must be on. 2. Mobile connected to internet services. |
| Postconditions: | 1. Carpark-Location API returns a list of car parks to the system. |
| Priority: | - |
| Frequency of Use: | - |
| Flow of Events: | 1. User logs in to the system. 2. The system gets car park locations through a Carpark-Location API. 3. System displays car parks on the map. |
| Alternative Flows: | - |
| Exceptions: | EX1: Mobile phone’s location not turned on.   1. System will prompt the user to turn on location services. |
| Includes: |  |
| Special Requirements: |  |
| Assumptions: | 1. Carpark-Location API will return the most accurate car park locations. |
| Notes and Issues: |  |