Use Cases

for

<CarParkGoWhere>

Version 1.1 approved

**Prepared by Ong Jing Jie, Seow Jing Woon, Cheng Yi Feng,
Jumana Haseen, Damien Lee Han Wei, Haoren**

<Group 46>

<01-Sep-2024>

Revision History

Name	Date	Reason For Changes	Version
Yi Feng	4/9/2024	Initial Version	1.0
Jing Jie	10/9/2024	Added Use Case Description	1.1

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 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:

2. User’s identity has been authenticated.
3. 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 the item in the database has been updated with a 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.

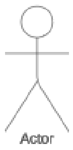
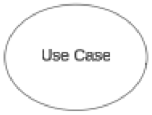
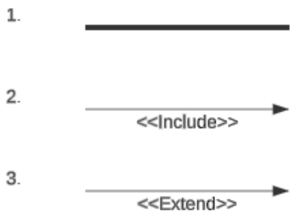
2.12. Assumptions

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.

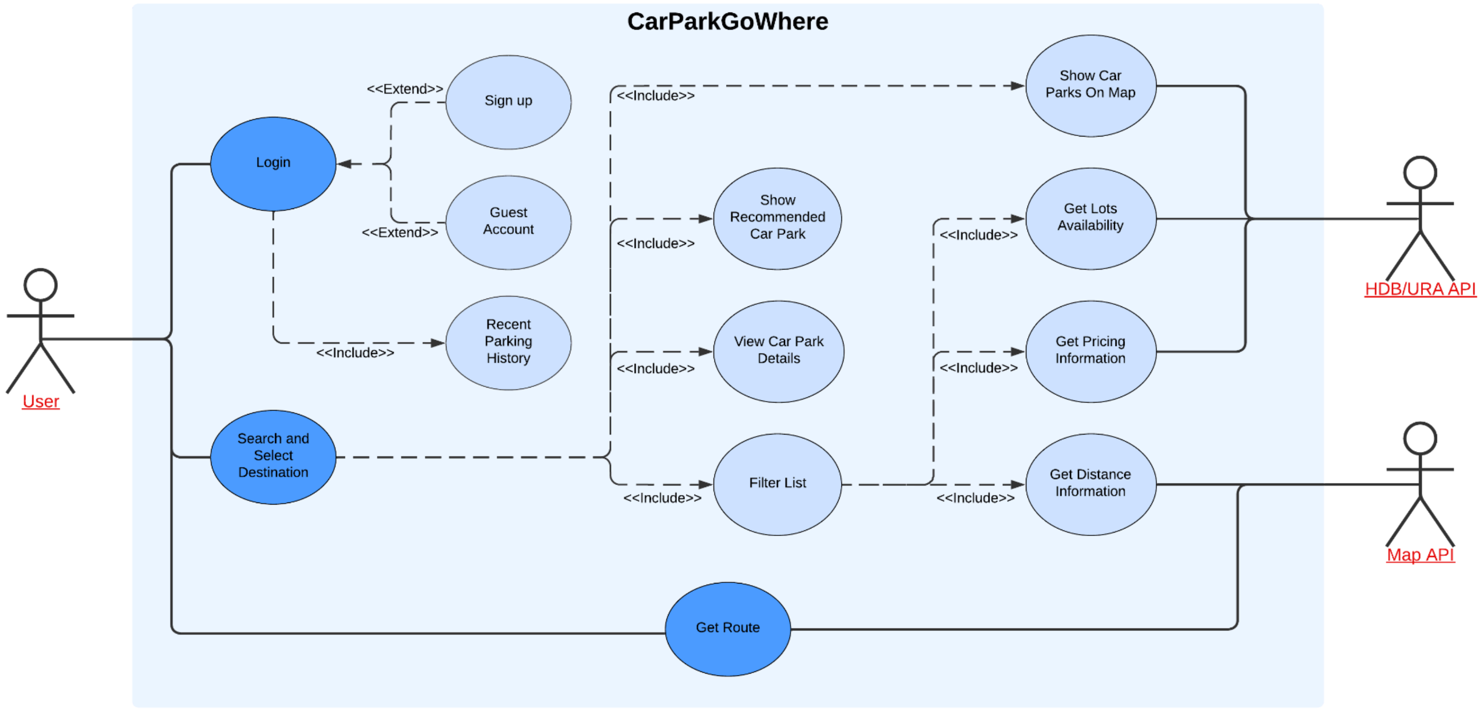
2.13. Notes and Issues

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 Diagram Elements

Component	Symbol	Descriptions
Actor		An external entity that interacts with system
Use Case		Unit of functionality performed by system, which yields result / value for Actor
System		It contains all the use cases that describe the system's functionalities and interactions.
Association		1. Connects Actors to Use Cases(s) in which they participate. 2. <<Include>> denotes a dependency on another Use Case. 3. <<Extend>> denotes an optional dependency on another Use Case

Use Case Diagram



Use Case Template

Use Case ID:	UID-001		
Use Case Name:	General Overview		
Created By:	Ong Jing Jie	Last Updated By:	Ong Jing Jie
Date Created:	05-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	User System HDB & URA API Map API
Description:	General overview of how the user can use the system to access the functions of the application to find the most suitable parking lots in car parks based on the user's final destination and other factors such as pricing and availability.
Preconditions:	1. The user has access to the website and is connected to the internet. 2. The system has access to real-time data from authoritative sources such as URA, HDB, and map data providers. 3. The user's location services (GPS) are enabled and accessible to the system.
Postconditions:	1. The user is provided with a list of parking spaces near the selected destination, sorted by criteria like proximity and price. 2. The user can view real-time availability and pricing of the selected parking spaces. 3. The user can choose a parking space and get the fastest / shortest route.
Priority:	High
Frequency of Use:	High
Flow of Events:	
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	The system must ensure data privacy and comply with local regulations regarding location-based services.
Assumptions:	The system assumes that the data provided by URA, HDB, and other sources is accurate and up-to-date. The system assumes that users have basic navigation and search skills on the website.
Notes and Issues:	

Flow Of Events

Actor Step	System step
1. User opens up the website	
	2. The system displays a login page, prompting the User to input his username and password to sign in, if not to sign up or continue as a guest account.
3. The user inputs his username and password correctly.	
	4. The system verifies the username and password and grants access.
	5. The system will transition to a map interface.
	6. The system will display a list of recent parking history, enabling users to select their recent destinations.
7. User inputs desired destination	
	8. The system queries real-time parking data from URA and HDB for parking spaces near the specified destination.
	9. The system retrieves real-time traffic and availability data.
	10. The system's algorithm uses the data it queried to choose the car park with the right balance of cost, car park distance from destination, lots availability, and distance to the car park.
	11. The system displays and highlights the best parking option based algorithm
12. The user selects a preferred parking space based on the provided option.	
	13. The system calculates and displays the fastest route from the user's location to the selected parking space.
14. The user can view the route on a map interface.	
	15. The system refreshes the page to update real-time availability and pricing information

	for the selected parking space.
16. The user navigates to the selected parking space using the route provided by the system.	

Flow of events: Alternative Flow

ID	Description
AF-S3	The user's username or password is incorrect. The system would not grant access and prompt the user to try again until the user chooses guest login, sign up for a new account or reset the password, and input the correct username and password. The system will return to step 2.
AF-S6	If the guest account is chosen, this step will be skipped and the system to proceed to step 7.
AF-S12	Users can customise the selection based on their needs instead of the result of the algorithm. The system will proceed to step 13 upon User selecting car park.
AF-14	User decides to change destination halfway through the journey and change the desired destination, then System will go to step 8.

Flow of Events: Exception

EX1-Destination is invalid or not accessible by land transport	System will display an error message "Destination unreachable" nad prompt user to input another destination

Use Case ID:	UID-002		
Use Case Name:	Login		
Created By:	Ong Jing Jie	Last Updated By:	Cheng Yi Feng
Date Created:	05-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	User System Database
Description:	User login using username and password to access system services.
Preconditions:	1. The User's device must have access to the internet. 2. The system does not currently have a User logged in. 3. The System must be connected to the database. 4. The database must be online.
Postconditions:	1. The user can access the map interface to access system services.
Priority:	High
Frequency of Use:	Low
Flow of Events:	1. The system will display the login interface. 2. The user inputs a username and password and clicks the "sign in" button. 3. The system verifies the login credentials. 4. Upon successful verification, the system will transit to the map interface. 5. Users then have access to system services.
Alternative Flows:	AF-S2:New User signup 1. The system will have a signup interface and allow User to create their account. The system will store the details in the database for further use. 2. Upon successful creation, the system will revert to the login interface, step 1. AF-S2:User chooses to use a guest account 1. The System will bypass step 3 and go to step 4. AF-S3:Incorrect credentials 1. The system will inform the user that the username or password is wrong. 2. The system will return to Step 1.
Exceptions:	EX1:System unavailable 1. The user is informed of system maintenance or downtime.
Includes:	UID-003 UID-004 UID-005
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-003		
Use Case Name:	Sign Up		
Created By:	Ong Jing Jie	Last Updated By:	Ong Jing Jie
Date Created:	05-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	User System Database
Description:	Creating a new account to access system services.
Preconditions:	1. The User's device must have access to the internet. 2. The system does not currently have a User logged in. 3. The System must be connected to the database. 4. The database must be online. 5. TheUser has no existing account
Postconditions:	1. The user can log in using his newly created account
Priority:	High
Frequency of Use:	Once for new users
Flow of Events:	1. The system requests the User to input a username and password 2. The system verifies that login credentials are not repeated with other accounts. 3. Upon successful verification, the system updates its database with the credentials 4. The system will output a "Sign up successful" message and return to the login interface.
Alternative Flows:	AF-S1:Text fields are not filled appropriately 1. The system will prompt the user to fill in the required field AF-S2:Username is already in use. 1. The system will prompt an error message "Username is taken" and go back to step 1.
Exceptions:	EX1:System unavailable 1. The user is informed of system maintenance or downtime.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-004		
Use Case Name:	Guest Account		
Created By:	Ong Jing Jie	Last Updated By:	Cheng Yi Feng
Date Created:	05-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	Guest User
Description:	An alternative way to access system services without having to have an account.
Preconditions:	1. The User's device must have access to the internet. 2. The system does not currently have a User logged in. 3. The System must be connected to the database. 4. The database must be online. 5. TheUser has no existing account
Postconditions:	1. The user can access the map interface to access system services.
Priority:	High
Frequency of Use:	High for Users that do not wish to create an account, no use for users with an account.
Flow of Events:	1. The system will display the login interface. 2. The user will select a guest account. 3. The system will transition to a map interface.
Alternative Flows:	
Exceptions:	EX1:System unavailable 1. The user is informed of system maintenance or downtime.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-005		
Use Case Name:	Recent Parking History		
Created By:	Ong Jing Jie	Last Updated By:	Ong Jing Jie
Date Created:	05-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	User System Database
Description:	To allow users to quickly access a history of car parks they have recently visited, facilitating easy reselection of preferred locations.
Preconditions:	1. The user has access to the website and is connected to the internet. 2. The system has access to real-time data from authoritative sources such as URA, HDB, and map data providers. 3. The user's location services (GPS) are enabled and accessible to the system. 4. System must have an account logged in. 5. The system has previously stored data on the user's parking activities.
Postconditions:	1. The user is provided with a list of parking spaces that the user have recently visited.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The user accesses the recently visited car parks feature from the application menu. 2. The system retrieves the user's parking history from the database. 3. The system presents a list of car parks the user has visited, sorted by date with the most recent visits at the top. 4. The user browses through the list and can select a car park to view more details or to reinitiate a route search to that location.
Alternative Flows:	AF-S2:If no history is available 1. The system displays a message indicating that there are no recently visited car parks.
Exceptions:	
Includes:	
Special Requirements:	The system must ensure data privacy and comply with local regulations regarding location-based services.
Assumptions:	The system assumes that the data provided by URA, HDB, and other sources is accurate and up-to-date. The system assumes that users have basic navigation and search skills on the website.
Notes and Issues:	

Use Case ID:	UID-006		
Use Case Name:	Search And Select Destination		
Created By:	Cheng Yi Feng	Last Updated By:	Ong Jing Jie
Date Created:	06-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	User HDB & URA API Map API
Description:	Allow users to input and select a destination for finding nearby parking lots.
Preconditions:	1. The User's device must have access to the internet. 2. The System must be connected to the database. 3. The database must be online. 4. The user must have logged in or using a guest account.
Postconditions:	1. A destination is set for the parking lot search.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The User inputs the address line, coordinates or postal code of his destination. 2. The system verifies the destination and prepares to display nearby parking options. 3. The system will query for data such as lot availability and price from HDB & URA API. 4. The system will query the map API for the distance between the destination and the car park. 5. By default, the system's algorithm will come up with a suggestion for the best choice of car park, taking into account the cheapest price, highest lot availability, and shortest car park distance from the destination. 6. The system will display the choices of car parks together with respective prices, lot availability, car park distance from the destination and car park distance from the current location. 7. The user will then choose the recommended option.
Alternative Flows:	AF-S1: Users can view car park details by clicking/tapping on the car park icon AF-S7: Users can opt out of choosing the recommended option, instead filter the choices of car parks based on: 1. Shortest distance from destination 2. Cheapest car park 3. Highest parking lots available
Exceptions:	EX1:System unavailable 1. The user is informed of system maintenance or downtime.
Includes:	UID-010 UID-009 UID-008
Special Requirements:	
Assumptions:	

Notes and Issues:	

Use Case ID:	UID-007		
Use Case Name:	Get Route		
Created By:	Cheng Yi Feng	Last Updated By:	Ong Jing Jie
Date Created:	06-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	User Map API
Description:	To provide the user with the fastest route to the selected car park.
Preconditions:	1. The User’s device must have access to the internet. 2. The System must be connected to the database. 3. The database and API must be online. 4. Users must have logged in or using a guest account. 5. System and map API must have access to the user's current location.
Postconditions:	1. The user views the driving route to the selected car park and the walking route from the car park to the destination.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. Users select the desired car park from the list of choices. 2. The system captures the user’s current location and the location of the selected car park. 3. The system sends a request to the Map API to generate a route from the user’s current location to the selected car park. 4. The Map API provides the route data, which the system then displays on the map, highlighting the driving route in green. 5. The user will take the recommended route and reach the car park. 6. When the user's location matches the car park location, the system requests the Map API to generate a walking route from the car park to the user’s final destination.
Alternative Flows:	AF-S5:User did not follow the recommended route. 1. The system presents one or more alternative routes to the user, highlighting the quickest or most convenient path based on current conditions.
Exceptions:	EX1:System unavailable 1. The user is informed of system maintenance or downtime.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-008		
Use Case Name:	Get Lot Availability		
Created By:	Ong Jing Jie	Last Updated By:	Ong Jing Jie
Date Created:	06-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	User System HDB & URA API Map API
Description:	To provide the system with real-time information on the availability of parking spaces near their destination.
Preconditions:	1. The User's device must have access to the internet. 2. The System must be connected to the database. 3. The database and API must be online. 4. Users must have logged in or using a guest account. 5. System and map API must have access to the user's current location. 6. The user has input a destination.
Postconditions:	1. The system is provided with up-to-date information about parking availability near their destination.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system sends a request to the HDB & URA API to retrieve the real-time availability of nearby parking spaces. 2. The system receives and stores parking availability data, indicating which car parks have available spots.
Alternative Flows:	AF-S1: No Lot Availability Data 1. If the API fails to return availability data, the system informs the user and suggests trying again later.
Exceptions:	EX1:System unavailable 1. The user is informed of system maintenance or downtime. EX2:API Failure 1. If the Car Park API is down, the system alerts the user and provides cached or default data where possible.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-009		
Use Case Name:	Get Pricing Information		
Created By:	Ong Jing Jie	Last Updated By:	Ong Jing Jie
Date Created:	06-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	User System HDB & URA API Map API
Description:	To provide the system with real-time pricing information for parking lots near their destination.
Preconditions:	1. The User's device must have access to the internet. 2. The System must be connected to the database. 3. The database and API must be online. 4. Users must have logged in or using a guest account. 5. System and map API must have access to the user's current location. 6. The user has input a destination.
Postconditions:	1. The system is provided with up-to-date information about real-time pricing information near their destination.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system sends a request to the HDB & URA API to retrieve the real-time pricing data of nearby parking spaces. 2. The system receives and stores real-time pricing data for further use.
Alternative Flows:	AF-S1: No Pricing Information Available 1. If pricing data is unavailable or incomplete, the system informs the user and suggests trying again later or provides estimated pricing based on historical data.
Exceptions:	EX1:System unavailable 1. The user is informed of system maintenance or downtime. EX2:API Failure 1. If the Car Park API is down, the system alerts the user and provides cached or default data where possible.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-010		
Use Case Name:	Get Distance Information		
Created By:	Ong Jing Jie	Last Updated By:	Ong Jing Jie
Date Created:	06-Sep-2024	Date Last Updated:	06-Sep-2024

Actor:	User System HDB & URA API Map API
Description:	To provide the system with the distance from their current location or destination to nearby parking lots.
Preconditions:	1. The User's device must have access to the internet. 2. The System must be connected to the database. 3. The database and API must be online. 4. Users must have logged in or using a guest account. 5. System and map API must have access to the user's current location. 6. The user has input a destination.
Postconditions:	1. The system receives accurate distance information for nearby parking lots.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system sends a request to the Map API to calculate the distance from the user's current location to parking lots near the destination. 2. The system receives and stores the distance to each parking lot, measured in kilometres.
Alternative Flows:	AF-S1: No GPS Data Available 1. If the system cannot access the user's current location, it prompts the user to manually enter a starting point for distance calculation.
Exceptions:	EX1:System unavailable 1. The user is informed of system maintenance or downtime. EX2:API Failure 1. If the Car Park API is down, the system alerts the user and provides cached or default data where possible.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-011		
Use Case Name:	Show Car Parks on Map		
Created By:	Ong Jing Jie	Last Updated By:	Ong Jing Jie
Date Created:	10-Sep-2024	Date Last Updated:	10-Sep-2024

Actor:	User System HDB & URA API Map API
Description:	To provide the system with data of car parks on a map near the user's selected destination, allowing the system to
Preconditions:	1. The User's device must have access to the internet. 2. The System must be connected to the database. 3. The database and API must be online. 4. Users must have logged in or using a guest account. 5. System and map API must have access to the user's current location. 6. The user has input a destination.
Postconditions:	1. The system gets the address of car parks near the user's input destination.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system sends a query to the HDB/URA API to retrieve real-time data on car parks near the destination. 2. The system receives and stores the data for further processing.
Alternative Flows:	AF-S1: No GPS Data Available 1. If the system cannot access the user's current location, it prompts the user to manually enter a starting point for distance calculation. AF-S1:No Car Parks Available 1. If no car parks are available near the destination, the system displays a notification and suggests increasing the search radius or selecting a different area.
Exceptions:	EX1:System unavailable 1. The user is informed of system maintenance or downtime. EX2:API Failure 1. If the Car Park API is down, the system alerts the user and provides cached or default data where possible.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-012		
Use Case Name:	Show Recommended Car park		
Created By:	Ong Jing Jie	Last Updated By:	Ong Jing Jie
Date Created:	10-Sep-2024	Date Last Updated:	10-Sep-2024

Actor:	User System HDB & URA API Map API
Description:	To provide users with recommendations for car parks based on their preferences, proximity, pricing, availability, and other relevant factors, allowing them to easily find the best parking option near their destination.
Preconditions:	1. The User's device must have access to the internet. 2. The System must be connected to the database. 3. The database and API must be online. 4. Users must have logged in or using a guest account. 5. System and map API must have access to the user's current location. 6. The user has input a destination.
Postconditions:	1. The system displays the recommended car park.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system evaluates the available car parks based on user preferences, such as price range, proximity to the destination, and real-time availability. 2. System uses an algorithm to decide the best car park for users. 3. System will highlight the recommended choice and rank the other choices of car park.
Alternative Flows:	AF-S2: User has chosen a preference. 1. System will display the choice of car park and rank it base on information and user's preference.
Exceptions:	EX1: System unavailable 2. The user is informed of system maintenance or downtime. EX2: API Failure 2. If the Car Park API is down, the system alerts the user and provides cached or default data where possible.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-013		
Use Case Name:	View Car Park Details		
Created By:	Cheng Yi Feng	Last Updated By:	Ong Jing Jie
Date Created:	10-Sep-2024	Date Last Updated:	10-Sep-2024

Actor:	User System HDB & URA API Map API
Description:	Allow users to view detailed information about the car park such as pricing rates of car park, distance from car park to destination, lots availability, estimated walking duration from car park to destination.
Preconditions:	1. The User's device must have access to the internet. 2. The System must be connected to the database. 3. The database and API must be online. 4. Users must have logged in or using a guest account. 5. System and map API must have access to the user's current location. 6. The user has input a destination.
Postconditions:	1. System displays detailed car park information.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system shows a list of car parks available within a specified radius of the user's destination. 2. The user can scroll through the list of available car parks and select a specific car park to view more details. 3. The system retrieves detailed information from Car Park API for the selected car park from the database.
Alternative Flows:	AF-S2: User does not select a specific car park from list of available car parks 1. System does not show detailed car park information
Exceptions:	EX1: System unavailable 1. The user is informed of system maintenance or downtime. EX2: API Failure 1. If the Car Park API is down, the system alerts the user and provides cached or default data where possible.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	UID-014		
Use Case Name:	Filter List		
Created By:	Cheng Yi Feng	Last Updated By:	Ong Jing Jie
Date Created:	10-Sep-2024	Date Last Updated:	10-Sep-2024

Actor:	User System HDB & URA API Map API
Description:	Allows users to customize their car park search results by toggling filters on or off. Users can apply filters like the cheapest car park rates, shortest distance from the car park to their destination, and highest parking lot availability to refine their search based on their preferences.
Preconditions:	1. The User’s device must have access to the internet. 2. The System must be connected to the database. 3. The database and API must be online. 4. Users must have logged in or using a guest account. 5. System and map API must have access to the user's current location. 6. The user has input a destination.
Postconditions:	1. Users successfully apply filters to view car park options that match their preferences.
Priority:	High
Frequency of Use:	High for users that want to filter their search results, no use for users that do not require to filter their search results.
Flow of Events:	1. The user navigates to the filter list section. 2. The system displays available filter options: ○ Cheapest car park rates ○ Shortest distance to the destination ○ Highest parking lot availability 3. The user toggles the desired filters on or off. 4. The system dynamically updates the search results based on the selected filters. 5. The user views the refined list of car parks that meet their selected criteria.
Alternative Flows:	AF-S3: User did not turn on any filters 1. By default, system shows the recommended car park.
Exceptions:	EX1: System unavailable 1. The user is informed of system maintenance or downtime.

	EX2: API Failure 1. If the Car Park API is down, the system alerts the user and provides cached or default data where possible.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	