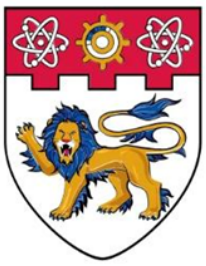


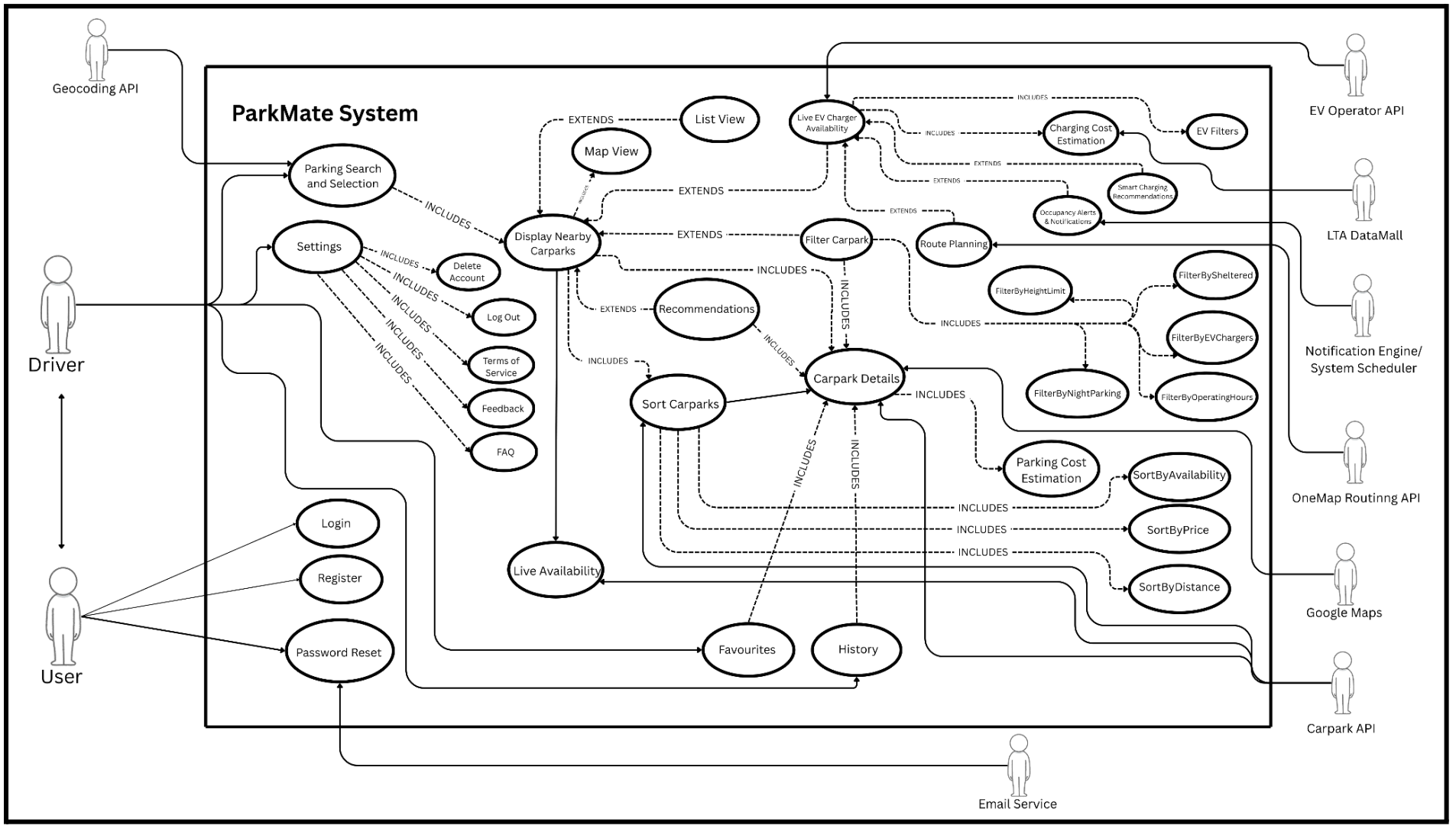
NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC2006 - Software Engineering

Lab 3 Deliverables

Lab Group	SCED
Team	Glitch
Members	Harshil Gupta (U2421166J)
	Goh Jin Long Abdillah (U2321634L)
	Guan Yibin (U2423353E)
	Kumar Preetham (U2422986F)
	Goh Jun Xian Bryant (U2423462J)

1. Use Case Diagram



3. Use Case Descriptions

I. For Functional Requirement #1

a. Parking Search & Selection

Use Case ID:	#1-1		
Use Case Name:	Parking Search & Selection		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	06 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System, Geocoding API
Description:	Searches for nearby carparks by entering the destination. ParkMate provides geocoding suggestions based on the query, and when a suggestion is selected, the system sets the location as the search destination in google maps.
Preconditions:	● User account exists (#4-1) ● User logged in (#4-2) ● Parkmate has access to a geocoding service and a carpark database.
Postconditions:	Success: ● Destination is set as the search location ● List of nearby carparks is retrieved and displayed Failure: ● No location set
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User navigates to Search Carpark 2. Parkmate prompts users to input addresses with the “Current Location” option. 3. User input address 4. Parkmate retrieve location data from geocoding API 5. Parkmate displays displays list of locations suggestions based on query 6. User selects a location from list 7. Parkmate sets selection as search destination in google maps
Alternative Flows:	AF-S3: No location suggestions 1. Display message: No matching locations found. 2. Goes to Step 2
Exceptions:	EX-01: Geocoding API unavailable 1. Parkmate unable to connect to API 2. Display message: Unable to retrieve location data. Please try again later. EX-02: Invalid address format 1. User enters gibberish, special characters, or unsupported language.

Actor:	Driver(user), Parkmate System, Geocoding API
Description:	Searches for nearby carparks by entering the destination. ParkMate provides geocoding suggestions based on the query, and when a suggestion is selected, the system sets the location as the search destination in google maps.
Preconditions:	● User account exists (#4-1) ● User logged in (#4-2) ● Parkmate has access to a geocoding service and a carpark database.
Postconditions:	Success: ● Destination is set as the search location ● List of nearby carparks is retrieved and displayed Failure: ● No location set
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User navigates to Search Carpark 2. Parkmate prompts users to input addresses with the “Current Location” option. 3. User input address 4. Parkmate retrieve location data from geocoding API 5. Parkmate displays displays list of locations suggestions based on query 6. User selects a location from list 7. Parkmate sets selection as search destination in google maps
Alternative Flows:	AF-S3: No location suggestions 1. Display message: No matching locations found. 2. Goes to Step 2
	2. Display message: Invalid input. Please enter a valid address. EX-03: Carpark database not reachable 1. Parkmate cannot connect to the backend database storing the carpark information. EX-04:
Includes:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

b. Display Nearby Carparks

Use Case ID:	#1-2		
Use Case Name:	Display Nearby Carparks		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	06 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	ParkMate displays nearby carparks around the selected destination. The user can toggle between map view and list view.
Preconditions:	● Destination selected (#1-1: Parking search and selection) ● Parkmate has access to the carpark data
Postconditions:	Success: ● Parkmate displays nearby carparks Failure: ● No carpark display
Priority:	High
Frequency of Use:	High
Flow of Events:	1. Parkmate displays a toggle for user to choose list or map based view while showing map view on default 2. Parkmate displays markers on the map each representing a car park 3. User selects a marker. 4. Selected carpark will be saved to history. 5. Displays each carpark with following details(map view)(#1-8): - Name - Distance - Live availability of carpark slots - Price - Height limit - Last updated time
Alternative Flows:	AF-S1: List view 1. Displays each carpark as a list with the same details in Step 2 2. Clicking on any list will be save to history.
Exceptions:	EX-01: Carpark data service unavailable 1. Parkmate cannot fetch carpark details due to database outage. 2. Display: Carpark information unavailable. Please try again later EX-02: Map Rendering Failure 1. Map view cannot load 2. Default list view with available carparks
Includes:	#1-1 (destination selected), #1-8 (display carpark details)
Special	None

Actor:	Driver(user), Parkmate System
Description:	ParkMate displays nearby carparks around the selected destination. The user can toggle between map view and list view.
Preconditions:	● Destination selected (#1-1: Parking search and selection) ● Parkmate has access to the carpark data
Postconditions:	Success: ● Parkmate displays nearby carparks Failure: ● No carpark display
Priority:	High
Frequency of Use:	High
Flow of Events:	1. Parkmate displays a toggle for user to choose list or map based view while showing map view on default 2. Parkmate displays markers on the map each representing a car park 3. User selects a marker. 4. Selected carpark will be saved to history. 5. Displays each carpark with following details(map view)(#1-8): - Name - Distance - Live availability of carpark slots - Price - Height limit - Last updated time
Alternative Flows:	AF-S1: List view 1. Displays each carpark as a list with the same details in Step 2 2. Clicking on any list will be save to history.
Requirements:	
Assumptions:	AS-01: User has selected a destination
Notes and Issues:	None

c. Live Availability

Use Case ID:	#1-3		
Use Case Name:	Live Availability		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	06 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System, Carpark API
Description:	ParkMate fetches and refreshes live carpark availability data at fixed intervals. The system ensures users always see up-to-date information and flags outdated data when necessary.
Preconditions:	● Nearby carparks displayed(#1-2)
Postconditions:	Success: ● Displays updated availability for all visible carparks Failure: ● Timestamp of the last updated information
Priority:	High
Frequency of Use:	High
Flow of Events:	1. Parkmate fetch availability from Carpark API for all visible carparks 2. Parkmate displays the latest updated availability
Alternative Flows:	AF-S1: unable to fetch availability 1. Display last updated availability
Exceptions:	EX-01: Carpark API unavailable 1. Parkmate cannot connect to Carpark API 2. Display: “Live availability temporarily unavailable.” and shows last known data with timestamp. EX-02: API rate limit exceeded 1. Too many requests within a short period, API reject calls. 2. Parkmate pauses auto-refresh and displays: Service busy. Retrying shortly. EX-03: Timeout on data fetch 1. API does not response within timeframe 2. Parkmate shows last availability, display: Data not refreshed.
Includes:	#1-2 (carparks displayed)
Special Requirements:	SR-1: Auto-refresh every 60s (tentative to API constraints)
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1)
Notes and Issues:	None

d. Filter Carparks

Use Case ID:	#1-4		
Use Case Name:	Filter Carparks		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	06 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Users can apply filters to refine the list and map of nearby carparks in ParkMate. Users can enable or disable filters such as height limit, EV chargers, sheltered parking, night parking and operating hours.
Preconditions:	● Nearby carparks are displayed (#1-2).
Postconditions:	Success: ● Filtered results are displayed instantly on both map view and list view
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User clicks on filter button 2. Parkmate displays available filters (e.g. height limit, EV chargers, sheltered parking, night parking, carpark type.) 3. User chooses filter options 4. Parkmate updates the result on current view(map / list)
Alternative Flows:	AF-S3: User reset filter 1. User click on reset button 2. Clear all filter selected 3. Parkmate displays non-filtered results. (#1-2) AF-S4: No matching filters for carpark 1. Users filter options unable to display a matching carpark 2. Prompts (“No carparks matched”) 3. Goes to Step 2
Exceptions:	EX-01: Filter data missing 1. Carpark database has missing or inconsistent data for a chosen filter. 2. Parkmate shows partial data, display message: Some carparks may have missing information.
Includes:	#1-1 (destination selected), #1-2 (carparks displayed)
Special Requirements:	None
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1)
Notes and Issues:	None

1. Filter Height

Use Case ID:	#1-4-1-1		
Use Case Name:	Filter Height		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	08 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Users can enable or disable height filters
Preconditions:	● Nearby carparks are displayed (#1-2).
Postconditions:	Success: ● Filtered results are displayed instantly on both map view and list view
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User clicks on “height” filter button 2. Parkmate updates the result on the current view(map/list)
Alternative Flows:	AF-S3: No matching filters for carpark 1. Users filter options unable to display a matching carpark 2. Prompts (“No carparks matched”) 3. Goes to #1-4 Step 3 (choosing filter)
Exceptions:	EX-01: Height data missing 1. Some carparks do not have height limit informations 2. Parkmate display with message: Height data unavailable EX-02: Timeout on filter execution 1. Filtering takes too long >5 minutes 2. Displays: FItlering timed out, displaying previous results.
Includes:	#1-1 (destination selected), #1-2 (carparks displayed)
Special Requirements:	None
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1) AS-3: User on filter interface(#1-4 Step 2)
Notes and Issues:	None

Use Case ID:	#1-4-1-2		
Use Case Name:	Filter EV Charger		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	08 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Users can enable or disable “EV Charger” filters
Preconditions:	● Nearby carparks are displayed (#1-2).
Postconditions:	Success: ● Filtered results are displayed instantly on both map view and list view
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User clicks on “EV Charger” filter button 2. User chooses filter options 3. Parkmate updates the result on the current view(map/list)
Alternative Flows:	AF-S3: No matching filters for carpark 1. Users filter options unable to display a matching carpark 2. Prompts (“No carparks matched”) 3. Goes to #1-4 Step 3 (choosing filter)
Exceptions:	EX-01: EV Charger information missing 1. Some carparks do not have EV information 2. Parkamates display: EV information unavailable. EX-02: Timeout on filter execution 1. Filtering takes too long >5 minutes 2. Displays: FItlering timed out, displaying previous results.
Includes:	#1-1 (destination selected), #1-2 (carparks displayed)
Special Requirements:	None
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1) AS-3: User on filter interface(#1-4 Step 2)
Notes and Issues:	None

3. Filter Sheltered Parking

Use Case ID:	#1-4-1-3		
Use Case Name:	Filter Sheltered Parking		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	08 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Users can enable or disable “Sheltered Parking” filters
Preconditions:	● Nearby carparks are displayed (#1-2).
Postconditions:	Success: ● Filtered results are displayed instantly on both map view and list view
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User clicks on “Sheltered Parking” filter button 2. Parkmate updates the result on the current view(map/list)
Alternative Flows:	AF-S3: No matching filters for carpark 1. Users filter options unable to display a matching carpark 2. Prompts (“No carparks matched”) 3. Goes to #1-4 Step 3 (choosing filter)
Exceptions:	EX-01: Shelter information missing 1. Some carparks do not have shelter information 2. Parkamates display: shelter information unavailable. EX-02: Timeout on filter execution 1. Filtering takes too long >5 minutes 2. Displays: FItlering timed out, displaying previous results.
Includes:	#1-1 (destination selected), #1-2 (carparks displayed)
Special Requirements:	None
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1) AS-3: User on filter interface(#1-4 Step 2)
Notes and Issues:	None

4. Filter Night Parking

Use Case ID:	#1-4-1-4		
Use Case Name:	Filter Night Parking		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	08 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Users can enable or disable “Night Parking” filters
Preconditions:	● Nearby carparks are displayed (#1-2).
Postconditions:	Success: ● Filtered results are displayed instantly on both map view and list view
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User clicks on “Night Parking” filter button 2. Parkmate updates the result on the current view(map/list)
Alternative Flows:	AF-S3: No matching filters for carpark 1. Users filter options unable to display a matching carpark 2. Prompts (“No carparks matched”) 3. Goes to #1-4 Step 3 (choosing filter)
Exceptions:	EX-01: Night Parking information missing 1. Some carparks do not have night parking information 2. Parkamates display: Night Parking information unavailable. EX-02: Timeout on filter execution 1. Filtering takes too long >5 minutes 2. Displays: FItlering timed out, displaying previous results
Includes:	#1-1 (destination selected), #1-2 (carparks displayed)
Special Requirements:	None
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1) AS-3: User on filter interface(#1-4 Step 2)
Notes and Issues:	None

5. Filter Operating Hours

Use Case ID:	#1-4-1-5		
Use Case Name:	Filter Operating Hours		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	08 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Users can enable or disable “Operating Hours” filters
Preconditions:	● Nearby carparks are displayed (#1-2).
Postconditions:	Success: ● Filtered results are displayed instantly on both map view and list view
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User clicks on “Operating Hours” filter button 2. Parkmate updates the result on the current view(map/list)
Alternative Flows:	AF-S3: No matching filters for carpark 1. Users filter options unable to display a matching carpark 2. Prompts (“No carparks matched”) 3. Goes to #1-4 Step 3 (choosing filter)
Exceptions:	EX-01: Operating Hours information missing 1. Some carparks do not have operating hours information 2. Parkamates display: Operating Hours information unavailable. EX-02: Timeout on filter execution 1. Filtering takes too long >5 minutes 2. Displays: FItlering timed out, displaying previous results
Includes:	#1-1 (destination selected), #1-2 (carparks displayed)
Special Requirements:	None
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1) AS-3: User on filter interface(#1-4 Step 2)
Notes and Issues:	None

e. Sort Carparks

Use Case ID:	#1-5		
Use Case Name:	Sort Carparks		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	06 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System, Carpark API
Description:	Sort the displayed carparks based on different criteria such as availability, price, and distance. ParkMate updates the carpark list and map view accordingly in real time.
Preconditions:	● Nearby carparks displayed (#1-2)
Postconditions:	Success: ● Carparks sorted based on criteria
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. Parkmate displays sorting options (Availability, Price, Distance) 2. User selects a sorting criteria 3. Parkmate reorders the result on current view (map & list) 4. If two or more carparks are tied, Parkmate resolves by shortest walking distance
Alternative Flows:	AF-S2: User selects or deselects criteria 1. Goes to Step 3
Exceptions:	EX-01: Missing data for sorting criteria 1. Some carpark do not have values for chosen criteria (e.g. missing prices, no availability data) 2. Parkmate places them at bottom, displays: Data unavailable EX-02: Sorting Logic Failure 1. Internal error when reordering results (e.g. null values or calculation errors) 2. Parkmate displays previous unsorted results with message: Sorting failed, showing default values.
Includes:	#1-2 (carparks displayed)
Special Requirements:	None
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1)
Notes and Issues:	None

f. Parking Cost Estimation

Use Case ID:	#1-6		
Use Case Name:	Parking Cost Estimation		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	06 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System, Carpark API
Description:	Estimate their parking cost by entering a start and end time. ParkMate calculates the cost based on the carpark's tariff rules and informs the user if cost estimation is not available due to missing data
Preconditions:	● User has selected a destination (#1-1) ● Nearby carparks displayed (#1-2)
Postconditions:	Success: ● Parkmate displays an estimated cost based on the time window Failure: ● Parkmate displays "Cost estimate unavailable."
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User opens a specific carpark's detail 2. User selects "Estimate Parking Cost" 3. Parkmate prompts user for start and end time 4. User enters desired start and end time 5. Parkmate retrieves carpark's fees from Carpark API 6. Parkmate calculates estimated cost based on time window 7. Parkmate displays estimated cost
Alternative Flows:	AF-S5: No available date on carpark's fee 1. Display "Cost estimate unavailable"
Exceptions:	EX-01: Invalid time window 1. Start time is later than end time, or end time is in the past 2. Display: Invalid time range. Please enter a valid range. EX-02: Carpark Tariff data missing 1. Carpark API does not return tariff rules, or rules are incomplete (e.g. weekend rates missing) 2. Display: Cost estimation failed due to missing data. EX-03: API timeout 1. Parkmate cannot fetch information due to slow response or API downtime. 2. Display: Unable to retrieve information. Please try again later.
Includes:	#1-2 (carparks displayed)
Special Requirements:	None

Actor:	Driver(user), Parkmate System, Carpark API
Description:	Estimate their parking cost by entering a start and end time. ParkMate calculates the cost based on the carpark's tariff rules and informs the user if cost estimation is not available due to missing data
Preconditions:	● User has selected a destination (#1-1) ● Nearby carparks displayed (#1-2)
Postconditions:	Success: ● Parkmate displays an estimated cost based on the time window Failure: ● Parkmate displays "Cost estimate unavailable."
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User opens a specific carpark's detail 2. User selects "Estimate Parking Cost" 3. Parkmate prompts user for start and end time 4. User enters desired start and end time 5. Parkmate retrieves carpark's fees from Carpark API 6. Parkmate calculates estimated cost based on time window 7. Parkmate displays estimated cost
Alternative Flows:	AF-S5: No available date on carpark's fee 1. Display "Cost estimate unavailable"
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1) AS-3: User provides a valid time window
Notes and Issues:	None

g. Recommendations

Use Case ID:	#1-7		
Use Case Name:	Recommendations		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	06 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Recommends up to three carparks based on key factors such as price, walking distance, and availability. Recommended carparks are displayed with badges for quick identification
Preconditions:	● Nearby carparks are displayed (#1-2).
Postconditions:	Success: ● ParkMate displays up to three recommended carparks with labels. Failure: ● If ranking fails, ParkMate displays the carparks without recommendations.
Priority:	Low
Frequency of Use:	High
Flow of Events:	1. Parkmate ranks carparks based on metrics (cheapest, closest walk, most available) 2. Parkmate displays recommendations (up to three) with metrics label 3. User selects a carpark 4. Parkmate opens selected carpark's details (#1-8)
Alternative Flows:	AF-S2: Ranking Algorithm Failure 1. Display carparks without recommendations.
Exceptions:	EX-01: Missing/Incomplete carpark data 1. Exclude affected entries from ranking. 2. Message: "Some carpark details are unavailable; recommendations shown from available data." EX-02: Offline/Network Loss 1. Connection lost during ranking or label fetch 2. Use last available cached recommendations. 3. Message: "Offline – showing cached data (if available)."
Includes:	#1-2 (carparks displayed), #1-8 (display carpark details)
Special Requirements:	None
Assumptions:	AS-1: User logged in (#4-2) AS-2: User selected a destination (#1-1)
Notes and Issues:	None

h. Carpark Details

Use Case ID:	#1-8		
Use Case Name:	Carpark Details		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	06 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System, Carpark API
Description:	Displays detailed information for a selected carpark, including pricing, lot availability, height limits, and operating hours.
Preconditions:	● User has selected a carpark
Postconditions:	Success: ● Parkmate displays complete carpark information Failure: ● Data retrieval fails, shows warning message
Priority:	High
Frequency of Use:	High
Flow of Events:	1. Parkmate retrieves carpark details from Carpark API 2. Parkmate displays carpark details: - Name - Distance - Live availability of carpark slots - Price - Carpark type - Height limit - Last updated time
Alternative Flows:	AF-S1: Retrieval Failure 1. Shows warning message “Fail to retrieve data.”
Exceptions:	EX-01: API timeout 1. Parkmate cannot fetch information due to slow response or API downtime. 2. Display: Unable to retrieve information. Please try again later. EX-02: Network Timeout / Connectivity Loss 1. Request to Carpark API takes too long or fails due to poor internet. 2. Show cached details (if available) 3. Message: “Offline – showing cached data (if available).”
Includes:	#1-2 (carparks displayed), #1-8 (display carpark details)
Special Requirements:	None
Assumptions:	AS-1: User logged in (#4-2)

Actor:	Driver(user), Parkmate System, Carpark API
Description:	Displays detailed information for a selected carpark, including pricing, lot availability, height limits, and operating hours.
Preconditions:	● User has selected a carpark
Postconditions:	Success: ● Parkmate displays complete carpark information Failure: ● Data retrieval fails, shows warning message
Priority:	High
Frequency of Use:	High
Flow of Events:	1. Parkmate retrieves carpark details from Carpark API 2. Parkmate displays carpark details: - Name - Distance - Live availability of carpark slots - Price - Carpark type - Height limit - Last updated time
Alternative Flows:	AF-S1: Retrieval Failure 1. Shows warning message “Fail to retrieve data.”
	AS-2: User selected a destination (#1-1)
Notes and Issues:	None

II. For Functional Requirement #2

a. Add Favourites

Use Case ID:	#2-1		
Use Case Name:	Add Favourites		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	07 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Allows users to add a carpark to their favourites list from the list view, map pop-up, or carpark details page.
Preconditions:	● Carparks are already displayed(#1-2)
Postconditions:	Success: ● Selected carpark is added to favourites Failure: ● Upon failure to add to favourites, shows error message.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User selects a carpark from the list, map, or details view. 2. User taps the “Favourite” icon. 3. ParkMate marks the carpark as favourited. 4. Carpark is stored in the user’s Favourites list. 5. ParkMate updates the UI with a filled favourite icon.
Alternative Flows:	AF-S2: Already Favourite 1. User already added to favourites. 2. Remove selected carpark from favourites. 3. Parkmate updates the icon to unfilled
Exceptions:	EX-01: Database/Storage Failure 1. ParkMate cannot write to the database or local storage (e.g., storage full, server error) 2. Display message: “Unable to add to favourites. Please try again later.” EX-02: Invalid Carpark ID 1. Selected carpark reference is missing, expired, or corrupted in the database. 2. Prevent add action; show: “Carpark not found. Cannot add to favourites.” EX-03: Network/Connectivity Loss 1. Action requires server confirmation but network is down 2. Message: No internet connection. Please try again later.
Includes:	#1-2 (carpark displayed)
Special Requirements:	None

Actor:	Driver(user), Parkmate System
Description:	Allows users to add a carpark to their favourites list from the list view, map pop-up, or carpark details page.
Preconditions:	● Carparks are already displayed(#1-2)
Postconditions:	Success: ● Selected carpark is added to favourites Failure: ● Upon failure to add to favourites, shows error message.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User selects a carpark from the list, map, or details view. 2. User taps the “Favourite” icon. 3. ParkMate marks the carpark as favourited. 4. Carpark is stored in the user’s Favourites list. 5. ParkMate updates the UI with a filled favourite icon.
Alternative Flows:	AF-S2: Already Favourite 1. User already added to favourites. 2. Remove selected carpark from favourites. 3. Parkmate updates the icon to unfilled
Assumptions:	AS-1: User already logged in.(#4-2) AS-2: User already entered a destination.(#1-1)
Notes and Issues:	None

III. For Functional Requirement #3

a. Feedback

Use Case ID:	#3-1		
Use Case Name:	Feedback		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	07 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Users can report incorrect carpark data or app-related issues using a feedback form accessible from the Profile Page and Carpark Details page.
Preconditions:	● User logged in. (#4-2)
Postconditions:	Success: ● Feedback is submitted successfully and the user sees a confirmation message.
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. User opens the Feedback Form via the Profile or Carpark Details page 2. User selects a feedback type (e.g., availability incorrect, entrance closed, height mismatch, other) 3. User optionally attaches photos and enters a free-text description 4. User taps Submit. 5. ParkMate validates and sends feedback to the server. 6. ParkMate displays a success message.
Alternative Flows:	AF-S2: Type not selected 1. User submits feedback without selecting a type. 2. Parkmate prompts user to complete required fields.
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	AS-1: User already logged in.(#4-2)
Notes and Issues:	None

b. Network Error or Database Failure for Feedback

Use Case ID:	#3-2		
Use Case Name:	Network Error or Database Failure for Feedback		
Created By:	Preetham	Last Updated By:	Preetham
Date Created:	23 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	User is unable to send Feedback successfully due to network or database issues.
Preconditions:	● User logged in. (#4-2)
Postconditions:	Success: ● Feedback is not submitted successfully and the user sees an error message.
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. User fills in required fields. 2. User taps Submit. 3. ParkMate fails to validate and sends feedback to the server. 4. ParkMate displays an error message.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	AS-1: User already logged in.(#4-2)
Notes and Issues:	None

IV. For Functional Requirement #4

a. Create Account

Use Case ID:	#4-1		
Use Case Name:	Create Account		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	08 Sept 2025	Date Last Updated:	08 Sept 2025

Actor:	Driver (User), ParkMate System
Description:	User creates a ParkMate account using email and password, with validation and duplicate account checks.
Preconditions:	● User does not already have an account with that email. ● Internet connection is available.
Postconditions:	● Success: New account created, user directed to login page. ● Failure: No account created, error message shown.
Priority:	High
Frequency of Use:	Occasional (first-time use).
Flow of Events:	1. User selects “Sign Up” 2. User enters email & password. a. System validates email format b. System checks for password requirement (ideally more than 12 characters, and use a combination of uppercase letters, lowercase letters, numbers, and symbols) 3. If valid -> Account created 4. If invalid -> System shows error message 5. If email already registered -> System shows “Email already in use” 6. If passwords do not match -> System shows “Password do not match”
Alternative Flows:	Weak password -> “Password must meet the requirements”
Exceptions:	Network/server error -> “Service unavailable”
Includes:	Validation Service
Special Requirements:	Secure password storage (hashed)
Assumptions:	User owns email provided
Notes and Issues:	Login via social may be added later.

b. Login

Use Case ID:	#4-2		
Use Case Name:	Login		
Created By:	Abdillah	Last Updated By:	Yibin
Date Created:	08 Sept 2025	Date Last Updated:	09 Sept 2025

Actor:	Driver (User), ParkMate System
Description:	User logs into the system with registered credentials.
Preconditions:	• User has a registered account.
Postconditions:	• On login: User navigates to Home screen.
Priority:	High
Frequency of Use:	Frequent
Flow of Events:	1. User enters email and password 2. System verifies credentials. 3. If valid -> Navigate to Home screen 4. If invalid -> “Email and password do not match” 5. Logout -> User session cleared, redirected to login screen
Alternative Flows:	Account not found → “Email not registered.”
Exceptions:	Server down → “Unable to connect.”
Includes:	Authentication Service
Special Requirements:	• Secure session management.
Assumptions:	User inputs correct credentials.
Notes and Issues:	

c. Password Reset

Use Case ID:	#4-3		
Use Case Name:	Password Reset		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	08 Sept 2025	Date Last Updated:	08 Sept 2025

Actor:	Driver (User), ParkMate System, Email Service
Description:	User resets forgotten password or changes password after login
Preconditions:	● Registered email. ● Internet connection.
Postconditions:	User has new valid password.
Priority:	Medium
Frequency of Use:	Seldom
Flow of Events:	1. User selects “Forgot Password” 2. System sends reset email. 3. User clicks reset link and sets new password 4. Authenticated Users can also change password after verifying current password
Alternative Flows:	Reset email not delivered → System shows “Unable to send reset email. Please try again.”
Exceptions:	Invalid reset link expired.
Includes:	Email Service
Special Requirements:	● Token expiration for reset link.
Assumptions:	User can access registered email.
Notes and Issues:	

d. Delete Account

Use Case ID:	#4-4		
Use Case Name:	Delete Account		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	08 Sept 2025	Date Last Updated:	08 Sept 2025

Actor:	Driver (User), ParkMate System
Description:	User deletes account permanently.
Preconditions:	User is logged in.
Postconditions:	Account and preferences removed.
Priority:	Medium
Frequency of Use:	Rare
Flow of Events:	1. User selects “Delete Account.” 2. System prompts confirmation 3. If confirmed -> Account deleted, favourites/preferences erased.
Alternative Flows:	User cancels deletion → Return to Profile page.
Exceptions:	Server error prevents deletion → “Unable to delete account at this time.”
Includes:	Confirmation Service
Special Requirements:	● Irreversible deletion.
Assumptions:	User intends permanent removal.
Notes and Issues:	

e. View Terms of Service and FAQ

Use Case ID:	#4-5		
Use Case Name:	View Terms of Service and FAQ		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	08 Sept 2025	Date Last Updated:	08 Sept 2025

Actor:	Driver (User), ParkMate System
Description:	User views TOS and FAQ from Profile page.
Preconditions:	User is logged in.
Postconditions:	Content displayed with version/date.
Priority:	Low
Frequency of Use:	Occasional
Flow of Events:	1. User navigates to Profile Page. 2. User selects TOS or FAQ 3. System displays content with version/date
Alternative Flows:	None
Exceptions:	Content unavailable → “Unable to load content.”
Includes:	Content Display Service
Special Requirements:	● Must show content version/date.
Assumptions:	TOS/FAQ content is updated in system backend.
Notes and Issues:	Consider multilingual support.

f. Log Out

Use Case ID:	#4-6		
Use Case Name:	Log Out		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	09 Sept 2025	Date Last Updated:	09 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Logged-in user logs out of the ParkMate application via the Settings page.
Preconditions:	● User logged in
Postconditions:	Success: ● Upon successful logout, the user is redirected to the Login page.
Priority:	High
Frequency of Use:	Medium
Flow of Events:	1. User navigates to the Settings page. 2. User taps the “Logout” button. 3. ParkMate invalidates the current user session. 4. ParkMate clears cached data related to the logged-in user. 5. ParkMate redirects the user to the Login page.
Alternative Flows:	None
Exceptions:	None
Includes:	#4-2 (logged in)
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

g. Failed Login Attempts & Account Lockout

Use Case ID:	#4-7		
Use Case Name:	Failed Login Attempts & Account Lockout		
Created By:	Harshil	Last Updated By:	Harshil
Date Created:	22 Sept 2025	Date Last Updated:	22 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Ensures that repeated failed login attempts are tracked, and the account is locked after reaching the retry limit to protect against brute-force attacks.
Preconditions:	● User has an existing registered account ● User attempts to log in with invalid credentials
Postconditions:	Success: ● User account remains active if below the retry limit. Failure: ● User account locked after exceeding retry limit.
Priority:	High
Frequency of Use:	Occasional
Flow of Events:	1. User enters incorrect email or password. 2. System verifies credentials. 3. System increments failed login counter. 4. If failed attempts < 5, show error “Invalid email or password.” 5. If failed attempts > 5, lock account and show “Too many failed attempts. Your account has been locked.” 6. System logs the lockout event
Alternative Flows:	● Admin manually unlocks account upon request. ● User resets password to unlock account (#4-3)
Exceptions:	● Network/server error shows "Unable to verify credentials at this time.”
Includes:	● Authentication Service
Special Requirements:	● Retry limit shall be configurable (default: 5 attempts) ● Locked accounts shall remain blocked until reset by user or admin
Assumptions:	● Users remember their credentials but may occasionally mistype
Notes and Issues:	● Consider adding captcha for suspicious activity.

h. Session Timeout & Auto-Logout

Use Case ID:	#4-8		
Use Case Name:	Session Timeout & Auto-Logout		
Created By:	Harshil	Last Updated By:	Harshil
Date Created:	22 Sept 2025	Date Last Updated:	22 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	Ensures inactive user sessions are automatically logged out after a defined period for security.
Preconditions:	● User is logged in successfully
Postconditions:	Success: ● User stays logged in with activity Failure: ● User session automatically ends after timeout
Priority:	Medium
Frequency of Use:	Regular (background security feature)
Flow of Events:	1. User logs in successfully (#4-2) 2. Systems starts inactivity timer (e.g., 15 minutes) 3. If user performs an action, time resets 4. If no activity within timeout, session expires 5. System logs the user out automatically and redirects to login screen 6. System displays “Your session has expired due to inactivity. Please log in again.”
Alternative Flows:	● User manually logs out before timeout (#4-6)
Exceptions:	● Network failure during session refresh, System logs out immediately
Includes:	● Authentication Service. Secure Session Management
Special Requirements:	● Timeout value shall be configurable (default: 15minutes) ● All cached sensitive data must be cleared upon logout
Assumptions:	● Users may forget to log out manually
Notes and Issues:	● Could allow user option to “Stay logged in” on trusted devices

V. For Functional Requirement #5

a. Handle External API Failure

Use Case ID:	#5-1 (possible edge case to look out for)		
Use Case Name:	Handle External API Failure		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	07 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver (User), ParkMate System, External API
Description:	When live availability data cannot be retrieved due to API issues, ParkMate displays a visible banner and continues showing last-known data
Preconditions:	User has searched for carparks (#1-1, #1-2)
Postconditions:	Banner displayed with timestamp, last-known data shown with “Stale” tag
Priority:	High
Frequency of Use:	High
Flow of Events:	1. ParkMate requests live carpark data from API 2. API call fails (unreachable or error) 3. ParkMate displays banner: “Live data temporarily unavailable - showing last update at HH:MM” 4. ParkMate shows cached availability with “Stale” tag 5. System retries in background 6. If the system successfully calls API, ParkMate refreshes data and removes banner.
Alternative Flows:	1. If the API recovers, ParkMate updates results 2. If the API is unavailable for more than 15 minutes, ParkMate displays, “Live data unavailable for extended period, please try again later or check offline options.” . Suggests user to retry after some time or contact support if persistent. 3. If local cached data is missing or corrupted, ParkMate displays, “Carpark availability information cannot be retrieved. We are working to restore service.”
Exceptions:	1. If there is a Database/network error when accessing cached data, ParkMate displays, “Carpark availability information cannot be retrieved. We are working to restore service.”
Includes:	#1-3 (Live Availability)
Special Requirements:	None
Assumptions:	Internet connection available
Notes and Issues:	none

b. Location Permission disabled

Use Case ID:	#5-2		
Use Case Name:	Location Permission disabled		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	07 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver (User), ParkMate System, Geocoding api
Description:	If location access is not enabled, ParkMate still allows destination-based search and adjusts distance labels accordingly.
Preconditions:	User had denied location access
Postconditions:	Success: Carparks are displayed relative to chosen destination Failure: None (search still works)
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User launches ParkMate and denies location permission when prompted. 2. User initiates a search by entering a destination address or postal code. 3. ParkMate queries the external API using the provided destination. 4. ParkMate calculates distances from the chosen destination to nearby carparks. 5. ParkMate displays the carpark list, with distances clearly labeled as **“from destination”** instead of **“from current location.”**
Alternative Flows:	1. If Geocoding API is unreachable or times out, ParkMate displays, “Location service temporarily unavailable, search results may be incomplete or inaccurate.”
Exceptions:	1. If Geocoding API returns an error or invalid data, System displays, “Could not process the destination address. Please check the input and try again.”
Includes:	#1-1 (Search Destination), #1-2 (Display Carparks).
Special Requirements:	None
Assumptions:	User manually inputs destination
Notes and Issues:	none

c. Throttling Manual refresh

Use Case ID:	#5-3		
Use Case Name:	Throttling Manual refresh		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	07 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver (User), ParkMate System
Description:	ParkMate limits excessive manual refresh requests to avoid API overload.
Preconditions:	Carparks displayed (#1-2).
Postconditions:	Success: Requests are queued/throttled Failure: User notified refresh not possible
Priority:	Low
Frequency of Use:	Low
Flow of Events:	1. User attempts manual refresh. 2. ParkMate checks rate limit (<=1 request/second) 3. If allowed, refresh executes. 4. If exceeded, requests are queued.
Alternative Flows:	1. If refresh rate exceeds limit, System will lockout, ParkMate displays, “Too many refresh attempts. Please wait before trying again.”
Exceptions:	1. If there is a Failure in queueing refresh requests due to internal error, User is notified: “Unable to refresh carpark list at this time. Try again later.”
Includes:	none
Special Requirements:	None
Assumptions:	None
Notes and Issues:	none

d. Display Refresh Loading State

Use Case ID:	#5-4		
Use Case Name:	Display Refresh Loading State		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	07 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver (User), ParkMate System
Description:	Show user a visible loading indicator when data refresh is in progress
Preconditions:	Carparks displayed (#1-2)
Postconditions:	User sees updated results after refresh OR error message if refresh fails
Priority:	Medium
Frequency of Use:	High (whenever user refreshes or data auto-updates)
Flow of Events:	1. User initiates refresh (manual or auto) 2. ParkMate displays loading state 3. Once data retrieved, loading indicator disappears. 4. Updated results shown to user
Alternative Flows:	1. If User switches away during refresh, loading state is paused and resumes once app returns to foreground.
Exceptions:	API failure (fallback to #5-1)
Includes:	#1-2 (Display Carparks)
Special Requirements:	None
Assumptions:	None
Notes and Issues:	none

VI. For Functional Requirement #6

a. Open in Maps

Use Case ID:	#6-1		
Use Case Name:	Open in Maps		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	07 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver (User), ParkMate System, google maps
Description:	ParkMate allows users to open a carpark location in their device's google maps application.
Preconditions:	User has selected a carpark from the list or map view.
Postconditions:	Either: maps app opens at correct location , or error displayed + address shown
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User selects "Open in google Maps" 2. ParkMate checks if google maps app is installed 3. If maps app available -> ParkMate opens location. 4. If no maps app -> ParkMate displays error + copyable address
Alternative Flows:	1. If unable to open in Google Maps due to OS restrictions, ParkMate displays, "Unable to launch maps application, Please install Google Maps. Address is copied to clipboard for manual pasting."
Exceptions:	1. Maps app installed but lacks required permissions, ParkMate displays, "Please allow our app to have permissions to access Google Maps". 2. If Google Maps returns an error for invalid or incomplete address, ParkMate displays, "Address not found in maps. Please verify or enter manually."
Includes:	none
Special Requirements:	None
Assumptions:	Device has internet connectivity
Notes and Issues:	none

b. Attribution and Transparency

Use Case ID:	#6-2		
Use Case Name:	Attribution and Transparency		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	07 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver (User), ParkMate System
Description:	ParkMate displays source attribution for all external data sources and explains assumptions behind derived values via tooltips.
Preconditions:	Carparks successfully fetched
Postconditions:	Data source attribution always visible on list and detail views, tooltips shown if available
Priority:	Low
Frequency of Use:	Constant
Flow of Events:	1. Park Mate fetches and displays data. 2. Data source attribution shown (e.g. HDB/URA). 3. For estimated values, tooltip displayed with calculation assumptions.
Alternative Flows:	1. If attribution data not supplied by external API, ParkMate displays, “Source data attribution unavailable.”
Exceptions:	1. If Tooltip explanation is broken or missing, Fallback to generic disclaimer: “Values estimated; see documentation for details.”
Includes:	none
Special Requirements:	None
Assumptions:	External APIs provide attribution requirements.
Notes and Issues:	none

VII. For Functional Requirement #7

a. View & Manage Carpark Selection History

Use Case ID:	#7-1		
Use Case Name:	View & Manage Carpark Selection History		
Created By:	Yibin	Last Updated By:	Abdillah
Date Created:	09 Sept 2025	Date Last Updated:	23 Sept 2025

Actor:	Driver(user), Parkmate System
Description:	View and manage their Carpark Selection History in ParkMate. The system automatically records selected carparks and allows users to revisit them.
Preconditions:	● User must be logged in. ● User has previously selected at least one carpark to generate history entries.
Postconditions:	Success: ● User's history of selected carpark is displayed. Failure: ● ParkMate displays a non-blocking message and loads locally cached entries if available.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User navigates to the History Tab from the Profile Page or main navigation. 2. ParkMate retrieves and displays the list of previously selected carparks. 3. Each history row displays the carpark name and timestamp. 4. User taps a history entry to reopen the Carpark Details and set it as the current selection. 5. ParkMate updates the history list immediately and syncs the changes.
Alternative Flows:	1. If there is no carpark selection history exists for the user (first-time use or cleared data).e, ParkMate displays a message: “No history found. Start selecting carparks to build your history.” 2. If User is not logged in, ParkMate displays only local (on-device) selection history and a prompt: “Log in to sync your full carpark history across devices.”
Exceptions:	1. If History sync to the server fails due to network or API outage, ParkMate displays a message: “Unable to sync carpark history. Changes will be synced when you are online.” 2. If Corrupted or unreadable history data (local or server-side), ParkMate displays an error: “History data could not be loaded.
Includes:	#1-8(to display selected carpark details)

Actor:	Driver(user), Parkmate System
Description:	View and manage their Carpark Selection History in ParkMate. The system automatically records selected carparks and allows users to revisit them.
Preconditions:	● User must be logged in. ● User has previously selected at least one carpark to generate history entries.
Postconditions:	Success: ● User's history of selected carpark is displayed. Failure: ● ParkMate displays a non-blocking message and loads locally cached entries if available.
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. User navigates to the History Tab from the Profile Page or main navigation. 2. ParkMate retrieves and displays the list of previously selected carparks. 3. Each history row displays the carpark name and timestamp. 4. User taps a history entry to reopen the Carpark Details and set it as the current selection. 5. ParkMate updates the history list immediately and syncs the changes.
Alternative Flows:	1. If there is no carpark selection history exists for the user (first-time use or cleared data).e, ParkMate displays a message: “No history found. Start selecting carparks to build your history.” 2. If User is not logged in, ParkMate displays only local (on-device) selection history and a prompt: “Log in to sync your full carpark history across devices.”
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

VIII. For Functional Requirement #8

a. Live EV Charger Availability

Use Case ID:	#8-1		
Use Case Name:	Live EV Charger Availability		
Created By:	Harshil	Last Updated By:	Harshil
Date Created:	19 Oct 2025	Date Last Updated:	19 Oct 2025

Actor:	Driver(user), Parkmate System, EV Operator/Aggregator APIs, Geocoding API
Description:	Displays nearby EV charging station availability (free / occupied / out-of-service) in real time, alongside carparks. Results refresh automatically and on demand, showing a “Last updated” timestamp and a stale data banner when needed.
Preconditions:	● User is on Map/List search views with a valid destination or current location (FR #1-1/#1-2). ● Network connectivity is available. ● At least one EV data source is configured (operator/aggregator feed)
Postconditions:	Success: ● Current EV charger availability is retrieved and rendered on map/list; entries show status, connector types, and update time. Failure: ● If live data cannot be fetched, ParkMate displays cached availability (if present) with a “Live data temporarily unavailable—showing last update at HH:MM” banner and disables the refresh action until backoff elapses.
Priority:	High (differentiating USP for EV users in Singapore)
Frequency of Use:	Medium
Flow of Events:	1. User opens the map or list view in ParkMate. 2. ParkMate automatically detects the user’s location or destination. 3. The system fetches nearby EV charging station data from available government or operator sources. 4. The system displayed chargers on the map, showing how many are available, in use, or offline. 5. Each charger shows extra details such as connector type, charging speed, and last updated time. 6. The user can refresh the results manually or wait for ParkMate to update automatically every few minutes. 7. If no chargers are found nearby, ParkMate shows a message suggesting the user expand the search radius or move the map. 8. If live data cannot be fetched, ParkMate displays the last known availability with a clear notice: “Showing last update at HH:MM.”

Alternative Flows:	1. If device location permission is denied, System uses the destination coordinates. If neither is available, System prompts: “Enter a destination to see nearby EV chargers.” 2. If no chargers are found, System displays: “No EV chargers found near this area.” and System offers quick actions: Expand radius / Pan map / Clear filters.
Exceptions:	1. No internet connection ○ ParkMate notifies the user that live data cannot be updated and continues showing the last known information until the connection is restored. 2. No EV Chargers found nearby ○ The app shows a friendly message like *“No EV chargers found in this area.”* It also suggests expanding the search radius. 3. Data Source Unavailable ○ If the external data source (e.g., LTA DataMall or an operator) is temporarily offline, ParkMate will use cached results and display *“Live data temporarily unavailable.”*
Includes:	● Geocoding (FR #1-1) for destination resolution ● Map/List rendering (FR #1-2) ● Filter & Sort (FR #1-4/#1-5) extended with EV filters
Special Requirements:	1. EV charger availability updates automatically every 1–2 minutes, or whenever the user taps “Refresh.” 2. Each station displays the last updated time so users know how fresh the information is. 3. EV chargers are color-coded and labeled clearly (e.g., green = available, red = in use, grey = offline). 4. Information is readable both through colors and text, so colorblind users can still understand charger status. 5. If live data is unavailable, cached information from the last successful update is shown instead.
Assumptions:	● At least one EV operator/aggregator provides programmatic feeds (push or poll) ● Static charger metadata (location, connectors, power) is available from public datasets or partner catalogs
Notes and Issues:	NIL

b. Filter by Connector Type & Charging Speed

Use Case ID:	#8-2		
Use Case Name:	Filter by Connector Type & Charging Speed		
Created By:	Harshil	Last Updated By:	Harshil
Date Created:	19 Oct 2025	Date Last Updated:	19 Oct 2025

Actor:	User (EV Driver), ParkMate System
Description:	This use case lets EV users filter and sort carparks or charging stations based on their vehicle's plug type and preferred charging speed (AC, DC, or Fast). It helps users quickly find chargers that are compatible with their car and reduce waiting time by showing only stations that match their needs.
Preconditions:	1. User has opened the Search Results or Map View after entering a destination. 2. EV charger data is available (either live or cached) 3. Connector type and power information are included in the dataset
Postconditions:	Success: - The list or map view updates to show only stations that match the selected connector type and speed. Failure: - If no stations match the filter, ParkMate displays *"No chargers found for selected options."*
Priority:	Medium-High (Enhances user personalization and usability for EV drivers)
Frequency of Use:	Medium-High (Users are likely to apply filters during every search session)
Flow of Events:	1. The user opens the Filter menu on the search or map screen. 2. ParkMate shows a section labeled "EV Charger Filters." 3. The user selects one or more connector types (e.g. Type 2, CCS2, CHAdeMO). 4. The user selects a preferred charging speed (e.g. AC Normal, DC Fast). 5. The user taps Apply Filters. 6. ParkMate updates the map and list instantly, showing only chargers that meet the criteria. 7. If no stations match, the system displays a message suggesting to clear or adjust filters.
Alternative Flows:	1. To clear filters, The user taps "Clear All Filters." and ParkMate resets the results to show all carparks and EV stations again. 2. When live data refreshes, ParkMate keeps the active filters and updates only the visible chargers that match them.

Exceptions:	● If some chargers don't provide connector or speed information, ParkMate still shows them but marks them as *"Unknown Type."* ● If no matching results are found, the app displays *"No EV chargers found matching your filters."* ● If a refresh fails while filters are active, cached data remains visible until the next update.
Includes:	NIL
Special Requirements:	● Results update immediately when filters are applied, no need to reload the whole page. ● ParkMate remembers the user's last chosen connector and speed preferences for the next session. ● Users can select multiple connector types and charging speeds simultaneously. ● Each filter option includes both icons and text for accessibility. ● Filters rely on standardized connector names from LTA or operator datasets.
Assumptions:	● The app has access to structured metadata for each EV station (connector type, speed). ● At least one filterable EV attribute (type or speed) is available for every listed station.
Notes and Issues:	NIL

c. Charging Cost Estimation

Use Case ID:	#8-3		
Use Case Name:	Charging Cost Estimation		
Created By:	Harshil	Last Updated By:	Harshil
Date Created:	19 Oct 2025	Date Last Updated:	19 Oct 2025

Actor:	User (EV Driver), ParkMate System, EV Operator APIs, LTA/DataMall datasets
Description:	This use case allows EV users to estimate the total cost of charging their vehicle at a chosen station. The app calculates the expected price based on the charger's tariff (cost per kWh) and the user's estimated battery percentage to be charged. This helps drivers compare stations and choose the most cost-effective option before heading there.
Preconditions:	1. The user has selected an EV charging station from the map or list view. 2. The station has available tariff or price data (fetched from operator API or dataset). 3. The user has entered (or defaulted) an estimated battery charge needed.
Postconditions:	Success: - The app displays the estimated total cost (e.g., "Approx. $6.20 for 28 kWh at $0.22/kWh") and an estimated charging time. Failure: - If price information is missing, ParkMate displays *"Pricing information unavailable for this station."*
Priority:	Medium (Enhances decision-making and transparency for EV drivers)
Frequency of Use:	Medium (Mainly when comparing multiple charging stations)
Flow of Events:	1. The user selects an EV charging station on the map to list. 2. ParkMate shows a "View Charging Details" screen with station info. 3. The user enters or adjusts the desired charge percentage (e.g., from 30% to 90%). 4. ParkMate retrieves the tariff rate for that station from the operator's data (e.g., $0.25 per kWh). 5. The app calculates: a. energy required = battery capacity × (target % − current %) b. total cost = energy required × price per kWh 6. The system displays both cost estimate and estimated charging time (based on charger power rating).

	7. If multiple tariffs apply (peak/off-peak), ParkMate shows the relevant rate or asks the user to select one.
Alternative Flows:	1. If the operator does not provide pricing info, ParkMate displays a message, *"Pricing unavailable, please refer to operator's app for latest rates."* 2. If the user's vehicle battery capacity is unknown, ParkMate uses an average EV size (e.g., 50 kWh) and labels the result *"Estimated based on typical EV capacity."* *3.* When the user adjusts charge percentage, ParkMate instantly recalculates and updates the cost.
Exceptions:	● If the data is not updated, ParkMate notifies the user that price and time data cannot be refreshed. Cached information (if any) is shown. ● *If tariff data cannot be retrieved, ParkMate displays "Live pricing unavailable—showing last known rate."* ● If the user enters unrealistic values (e.g., 110%), the app prompts *"Please enter a valid percentage."*
Includes:	NIL
Special Requirements:	● Results estimation updates immediately when the user changes input values. ● Display both rate ($/kWh) and total estimated cost, not just a single number. ● Allow manual input of vehicle battery capacity for more accurate results. ● ParkMate clearly labels that displayed costs are estimates and may vary by operator. ● All prices shown in Singapore Dollars (SGD).
Assumptions:	● Operator APIs or datasets provide at least one of: price per kWh, tariff table, or typical session cost. ● Basic EV model data (battery size, charging efficiency) are available in-app or from user input.
Notes and Issues:	NIL

d. Route Planning with Charging Stops

Use Case ID:	#8-4		
Use Case Name:	Route Planning with Charging Stops		
Created By:	Harshil	Last Updated By:	Harshil
Date Created:	19 Oct 2025	Date Last Updated:	19 Oct 2025

Actor:	User (EV Driver), ParkMate System, OneMap Routing API, EV Operator APIs / Charger Dataset
Description:	This use case enables EV users to plan routes that include charging stops automatically, based on their current battery level and available chargers along the way. The feature helps prevent “range anxiety” by identifying convenient charging points en route, ensuring users can reach their destinations efficiently.
Preconditions:	1. The user has entered a destination and selected “Route Planning.” 2. Location access is enabled or a starting point is manually entered. 3. The app has access to live or cached EV charger data.
Postconditions:	Success: - ParkMate displays a suggested route including optimal charging stops and estimated travel + charging time. Failure: - If no suitable chargers are found, ParkMate displays *“No available charging stops found along this route.”*
Priority:	High (Core differentiator for EV-focused users)
Frequency of Use:	Medium (Mainly for long-distance trips)
Flow of Events:	1. User enters a destination and taps “Plan Route.” 2. ParkMate retrieves the user’s current battery level (manual input). 3. The Route Planner Engine calculates distance and compares it against estimated remaining range. 4. If range is insufficient, ParkMate automatically identifies EV charging stations along or near the route using map and API data. 5. The app will suggest one route with the fastest overall time (driving + charging). 6. The user selects a preferred route. 7. The route is displayed on the Map View, with icons marking charging stops. 8. The app shows estimated travel time, charging durations, and total cost summary.
Alternative Flows:	1. If the user manually adds a charging stop, ParkMate recalculates the route and updates timing estimates.

	2. If several routes are viable, ParkMate lists them with estimated time and cost for user selection.
Exceptions:	● If network failure, ParkMate displays cached charger data and a message: *“Live route updates unavailable, using offline data.”* ● If no chargers are found, App displays: *“No chargers available along your selected route.”* and suggests expanding search radius.
Includes:	NIL
Special Requirements:	● Uses OneMap Routing API for navigation paths and LTA DataMall or operator datasets for charger data. ● If a selected charging stop becomes full, ParkMate refreshes and suggests the next nearest one. ● Route line includes charging icons, with colors showing availability (green = free, red = full). ● The interface must remain responsive within 3 seconds even when calculating multi-stop routes. ● All route and stop info should include text labels for users with color-vision deficiencies.
Assumptions:	● Real-time data (location, charger status, map routes) is accessible.
Notes and Issues:	NIL

e. Smart Charging Recommendations

Use Case ID:	#8-5		
Use Case Name:	Smart Charging Recommendations		
Created By:	Harshil	Last Updated By:	Harshil
Date Created:	19 Oct 2025	Date Last Updated:	19 Oct 2025

Actor:	User (EV Driver), ParkMate System, EV Charger APIs, LTA DataMall
Description:	This use case provides personalized recommendations for the best charging stations based on the user's location, current battery level, cost preferences, and live charger availability. It helps drivers quickly decide where to charge next, choosing between the nearest, cheapest, or fastest option.
Preconditions:	1. The user's current location or intended route is known. 2. Live or cached data on charger availability, pricing, and charging speed is accessible. 3. The user has granted permission for ParkMate to use their location or EV profile data.
Postconditions:	Success: - ParkMate displays a ranked list of recommended chargers labeled "Nearest," "Cheapest," and "Fastest." Failure: - If no live data is available, ParkMate displays cached results with a note: *"Live updates unavailable, showing last known data."*
Priority:	High (Core differentiator for EV-focused users)
Frequency of Use:	Medium-High (Mainly for long-distance trips)
Flow of Events:	1. **User** opens ParkMate's **"Find Charger"** feature. 2. The system automatically detects the user's **location and battery level** (manual entry if unavailable). 3. ParkMate retrieves live charger data, location, price per kWh, and charger type, from external APIs. 4. The recommendation engineer evaluates nearby stations based on: Distance, Charger Speed, Current queue or availability, and Price per kwh. 5. Upon display, the user selects a recommended option to view more details or start route planning.
Alternative Flows:	1. If user adjust preferences, ParkMate regenerates recommendations instantly. 2. If a charger's availability changes, ParkMate automatically refreshes rankings and highlights the new best option.
Exceptions:	● If data source unavailable, ParkMate displays: *"Unable to fetch live charger data. Please try again later."* ● If no chargers are found, The app suggests: *"No chargers within 3km, expand your search radius?"*

Includes:	NIL
Special Requirements:	● Combine distance, cost, and speed using a weighted algorithm for fairness. ● Automatically refresh recommendations every 2 minutes or on user request. ● Save driver's EV model, preferred connectors, and price range for smarter suggestions. ● Each recommendation shows why it was chosen (e.g., "Nearest – 0.8km away, $0.25/kWh"). ● Recommendations must load within 3 seconds under normal network conditions.
Assumptions:	● EV data APIs provide price, speed, and availability information in real time. ● User profile stores EV battery size and preferred connector types.
Notes and Issues:	NIL

f. Real-time Charger Occupancy Alerts & Notifications

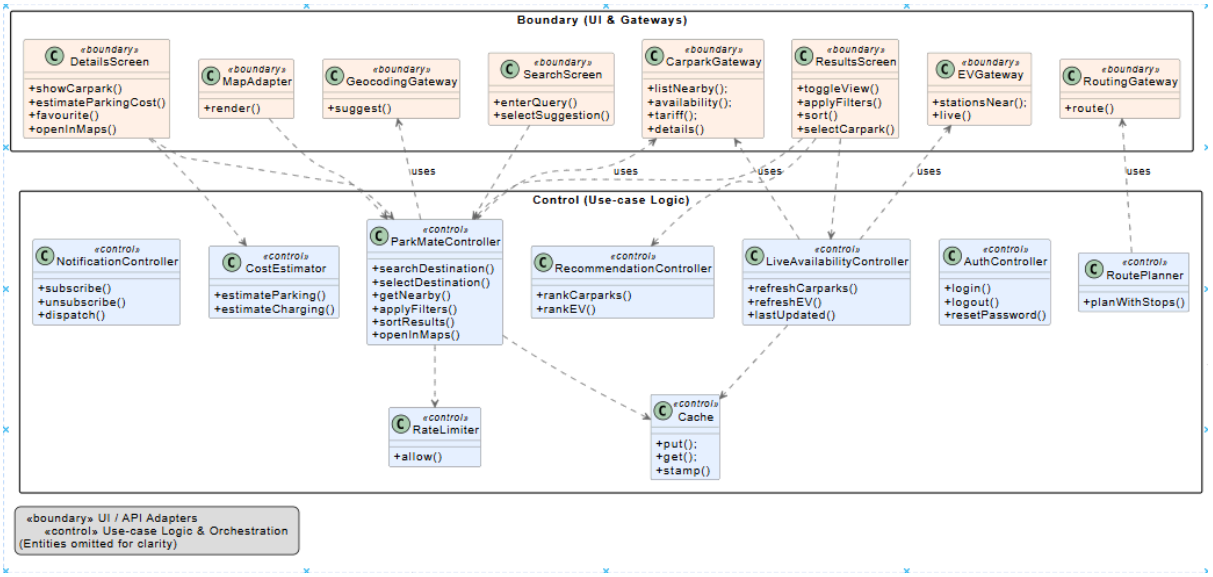
Use Case ID:	#8-6		
Use Case Name:	Real-time Charger Occupancy Alerts & Notifications		
Created By:	Harshil	Last Updated By:	Bryant
Date Created:	19 Oct 2025	Date Last Updated:	25 Oct 2025

Actor:	User (EV Driver), ParkMate Notification Engine, System Scheduler, EV Charger APIs, LTA DataMall
Description:	This use case enables ParkMate to notify users automatically when an EV charger near them, or one they've saved as a favourite, becomes available, occupied, or out of service. It reduces the need for manual refreshing and helps drivers time their arrival efficiently, especially in high-demand areas.
Preconditions:	1. The user is logged in and has location services enabled. 2. The user has viewed chargers via search or map at least once (to subscribe for alerts). 3. Notification permissions are granted to ParkMate.
Postconditions:	Success: - ParkMate sends an in-app or push notification when a charger's status changes, matching the user's preferences (e.g., "Charger A is now available"). Failure: - If alerts cannot be delivered (e.g., network down), they are queued and sent once connectivity resumes.
Priority:	High (Enhances convenience and makes the app feel "alive")
Frequency of Use:	High (Continuous background process)
Flow of Events:	1. User taps the bell icon or "Notify Me" on a charger card. 2. ParkMate subscribes the user to that charger's availability feed (through backend listener). 3. The Notification Engine monitors live API updates every 1–2 minutes. 4. When the charger's state changes (e.g., occupied → available), the system triggers an event. 5. ParkMate sends a notification: "A Charging slot at (Favourited Location) has freed up!" 6. The user can tap the notification to open the charger's details or start route planning. 7. Notifications auto-expire if the charger is re-occupied before user interaction.
Alternative Flows:	3. If a user unsubscribes from alerts, ParkMate stops monitoring that charger. 4. If system notifications are muted, alerts appear only within the app (banner style)
Exceptions:	● During API downtime, ParkMate Displays "Live updates temporarily unavailable." Cached state is retained.

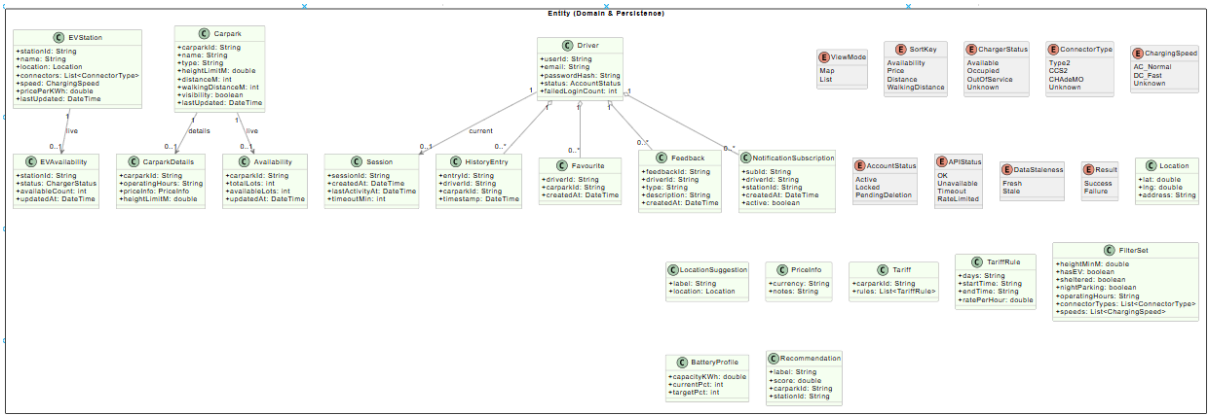
	● If notification permission is denied, ParkMate shows prompt “Enable notifications to receive live charger updates.”
Includes:	NIL
Special Requirements:	NIL
Assumptions:	● EV charger APIs support frequent updates (≤ 2 minutes). ● Push notification service (e.g., Firebase Messaging) is configured and reliable.
Notes and Issues:	NIL

3. CLASS DIAGRAM

1. Boundary + Control Classes



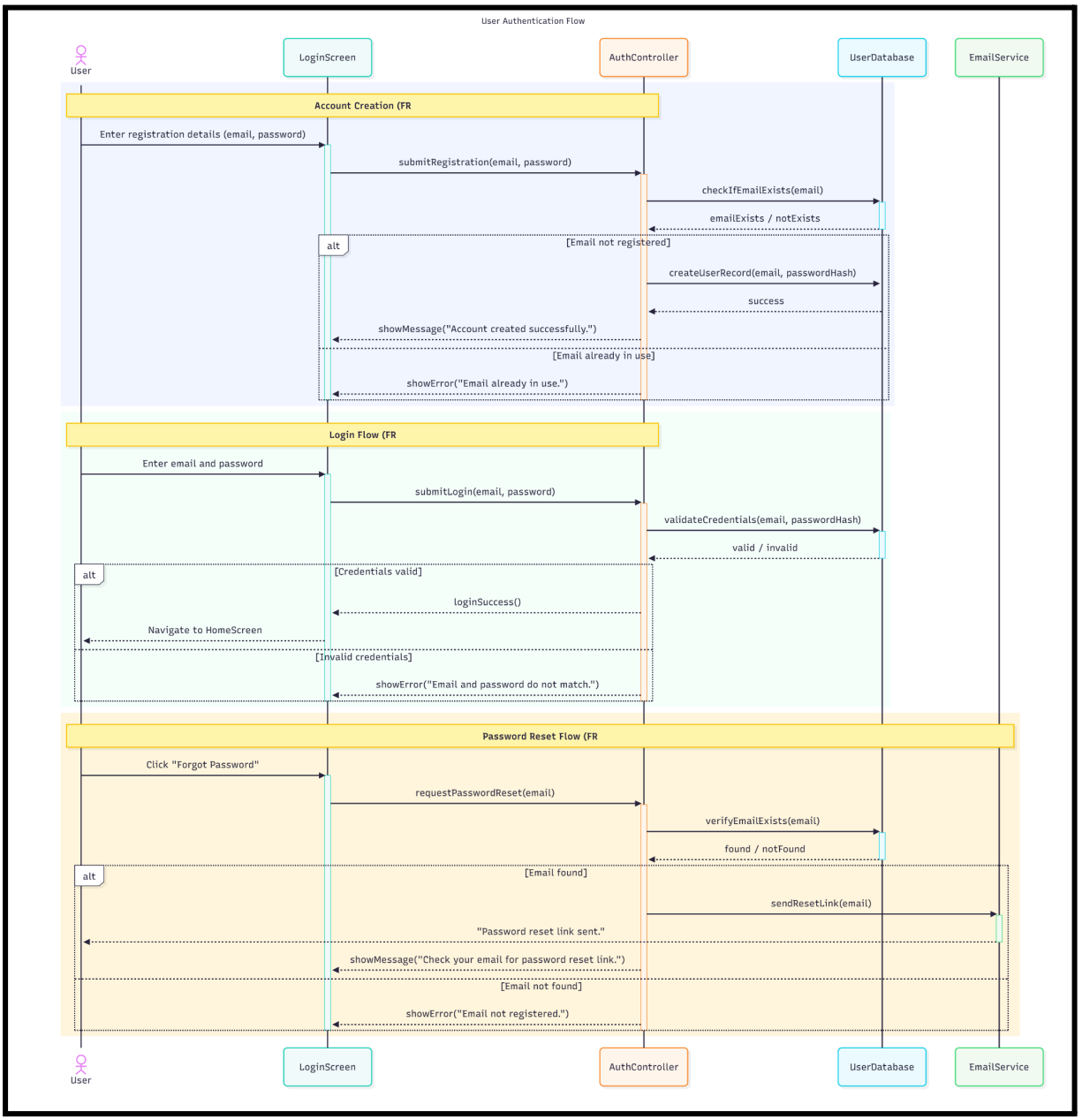
2. Entity Classes



4. Sequence Diagrams

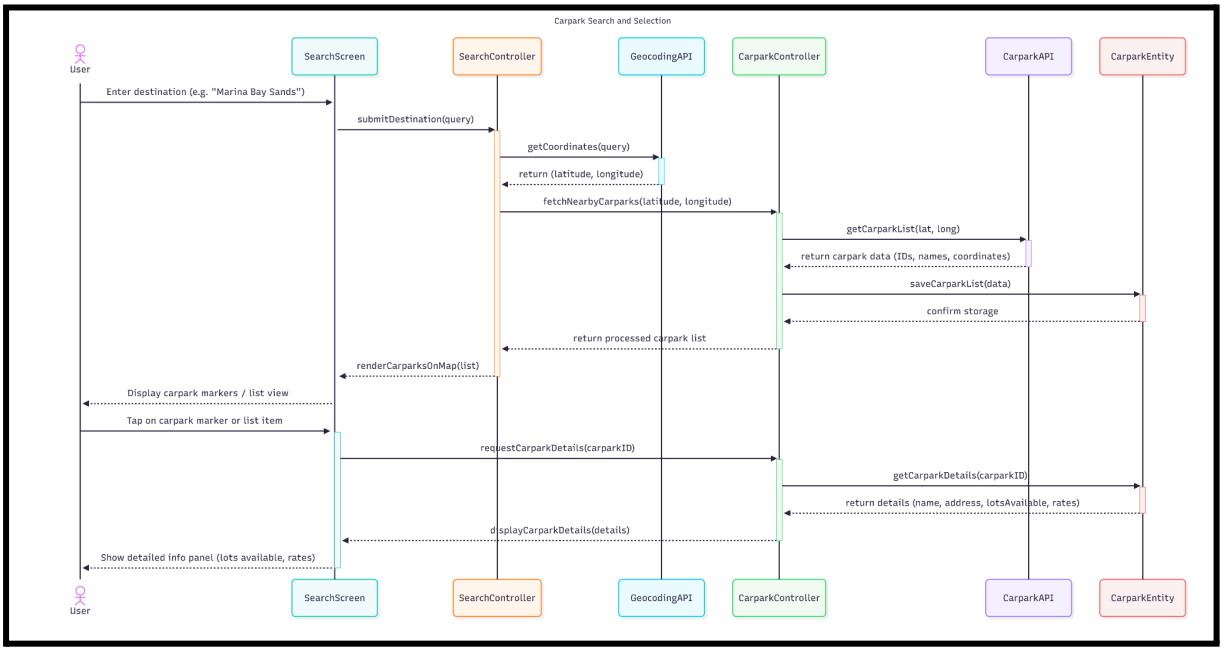
a. User Authentication Flow

This diagram illustrates the full user authentication process, including account registration, login, and password reset. It captures how the user interacts with the LoginScreen, AuthController, UserDatabase, and EmailService to perform secure access and credential recovery.



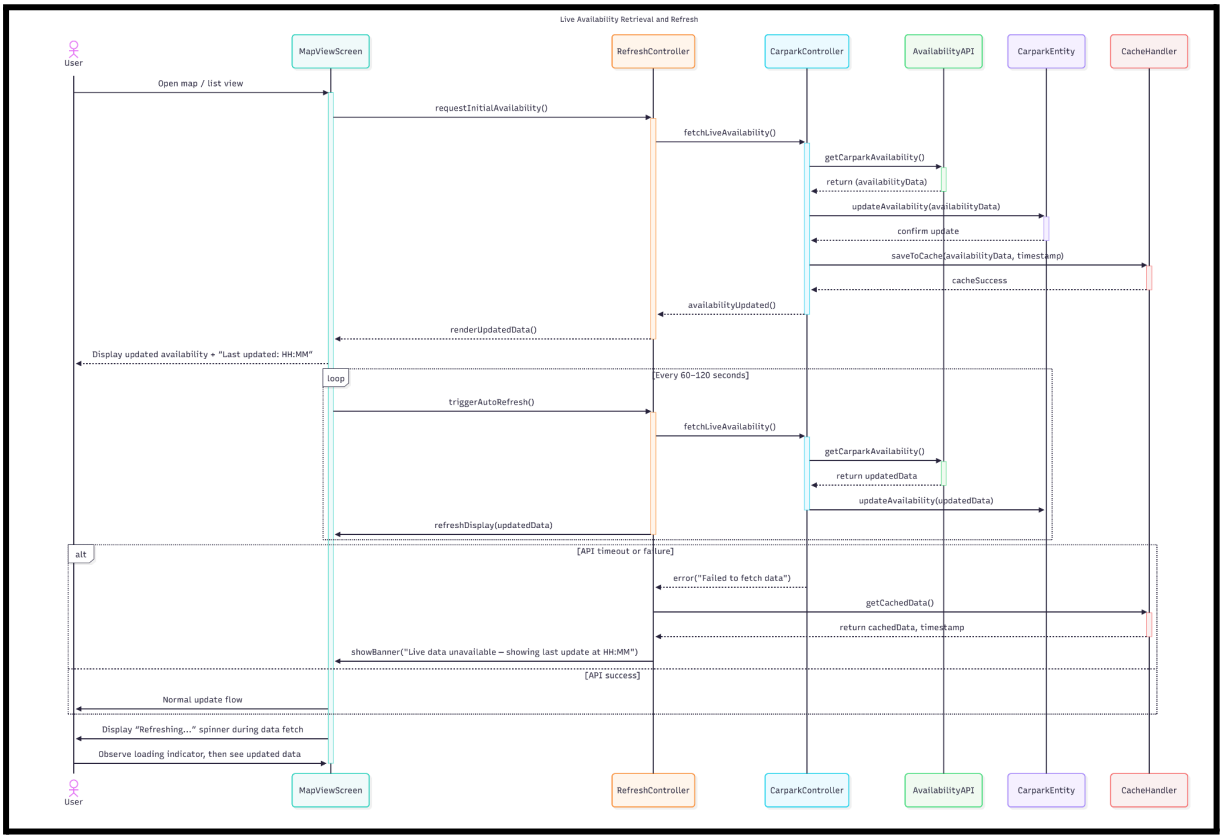
b. Carpark Search and Selection

This diagram represents the user flow for searching and displaying carparks near a destination. It follows the use cases under FR #1-1 (Destination Search) and FR #1-2 (Display Carparks), showing how user input is processed through the search interface, Geocoding API, and CarparkController to render the map/list results.



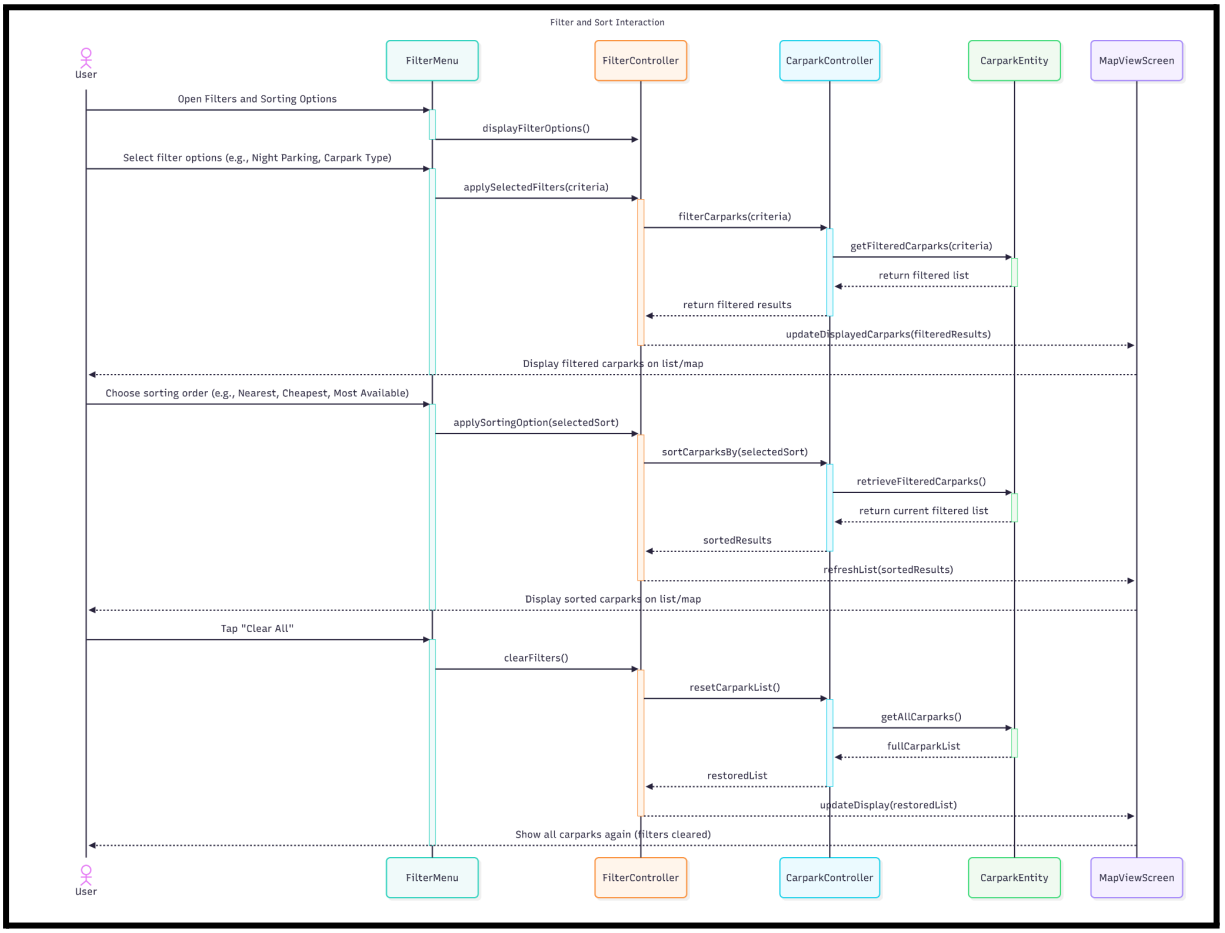
c. Live Availability Retrieval and Refresh

This diagram illustrates how ParkMate periodically fetches live carpark availability data from the external API (e.g., LTA DataMall), updates the internal entities, and displays refreshed information on the map/list view. It also shows what happens during API failures and loading state management, as described under your reliability functional requirements.



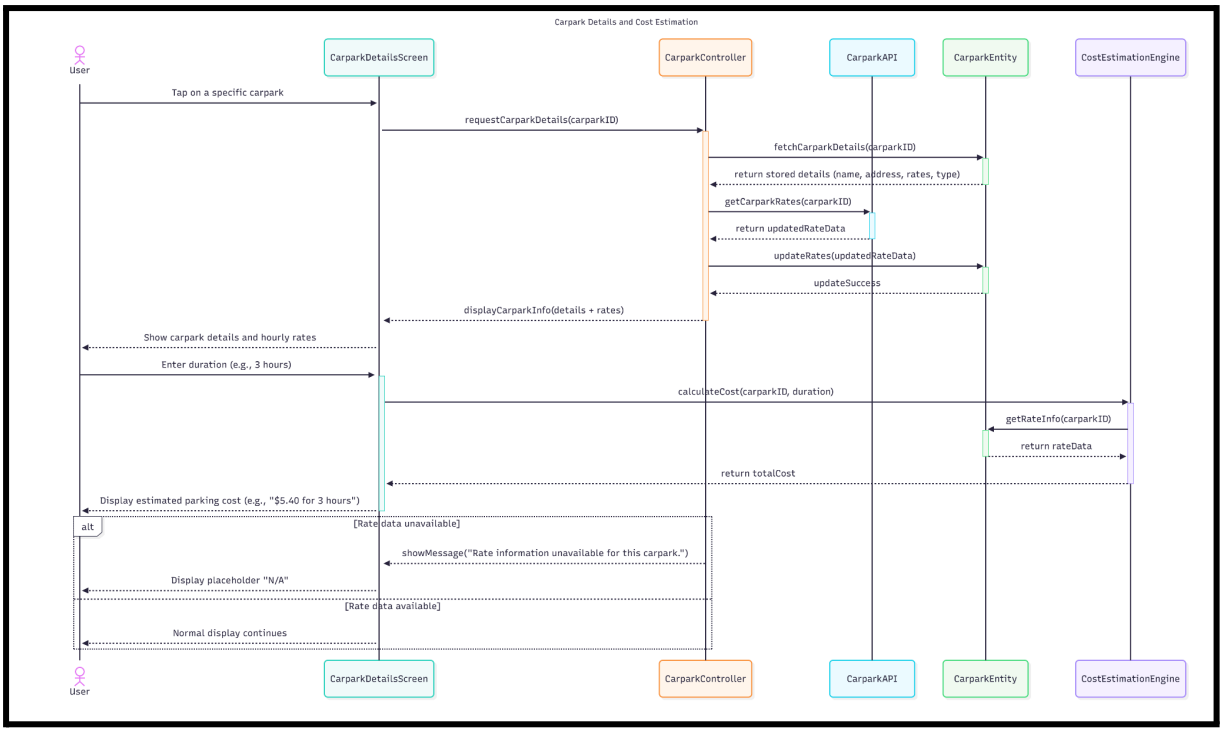
d. Filter and Sort Interaction

This diagram shows the interaction flow when a user applies filters and sorting preferences to refine carpark search results. It follows exactly the documented behavior — the user selects filters (e.g., carpark type, night parking, availability) or sorting order (e.g., nearest, cheapest, most available), which the system processes locally and displays updated results.



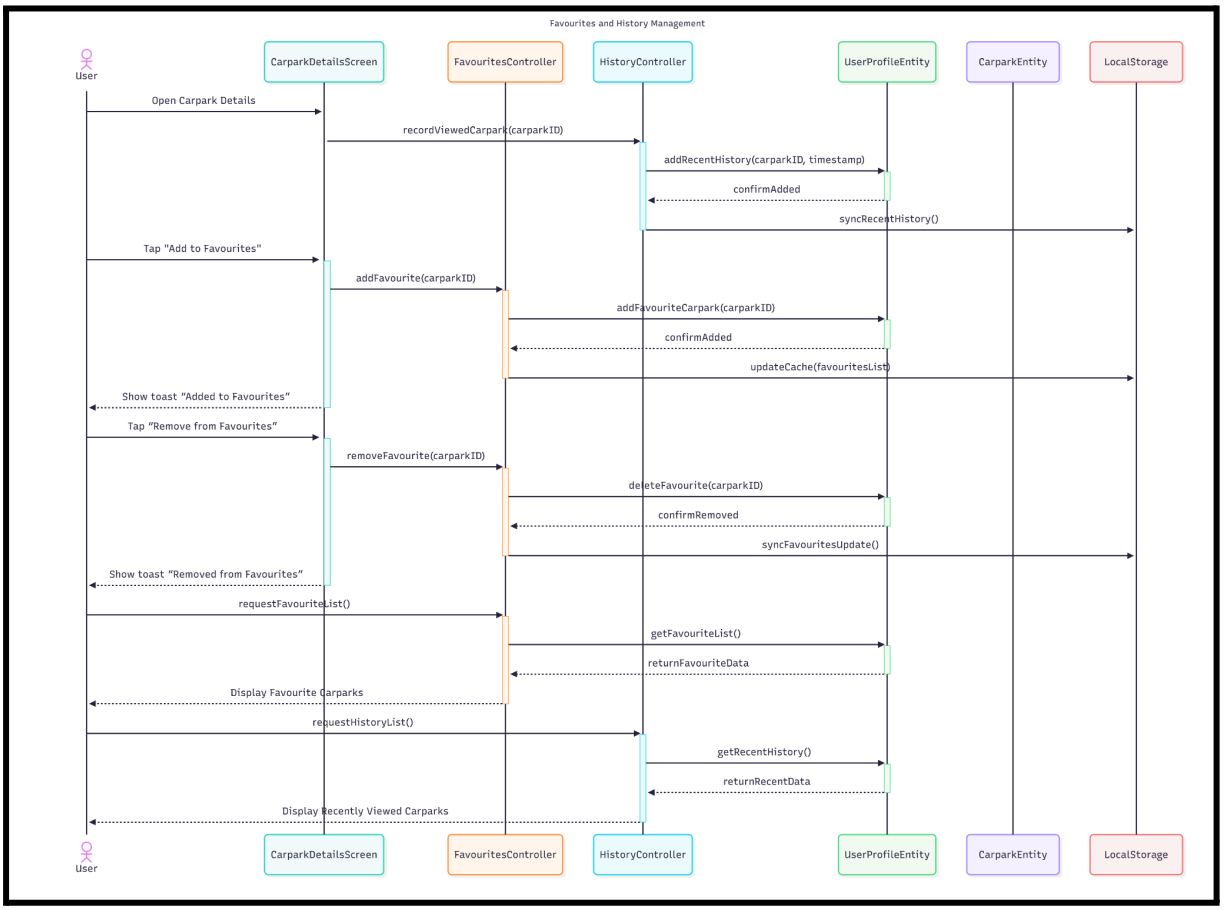
e. Carpark Details and Cost Estimation

This diagram illustrates how ParkMate handles the process when a user selects a carpark to view its details and estimates parking cost based on the rates provided by the data source. It follows the flow from FR #1-6 (View Carpark Details) and FR #1-8 (Parking Cost Estimation), showing the interaction between the user, display screen, controllers, and external data sources.



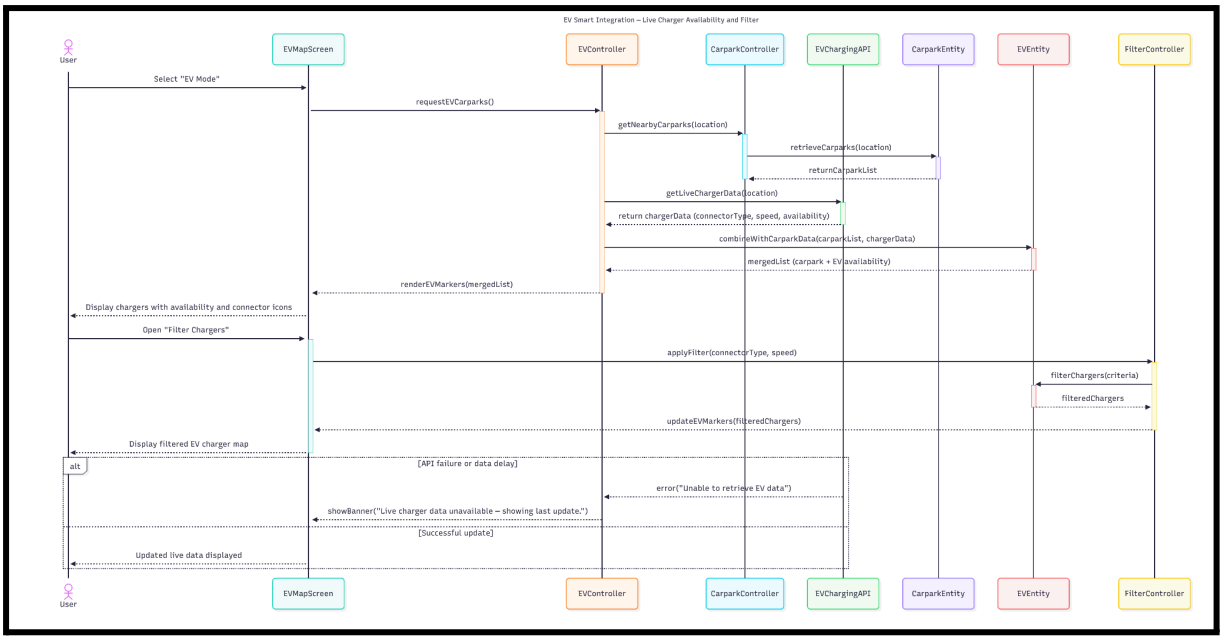
f. Favourites and History Management

This diagram shows how the user marks or removes a carpark as a favourite and how the system maintains and displays the user's favourite list and viewing history. It follows FR #2-1 (Manage Favourites) and FR #7-1 (Recent History Tracking) from your specification.



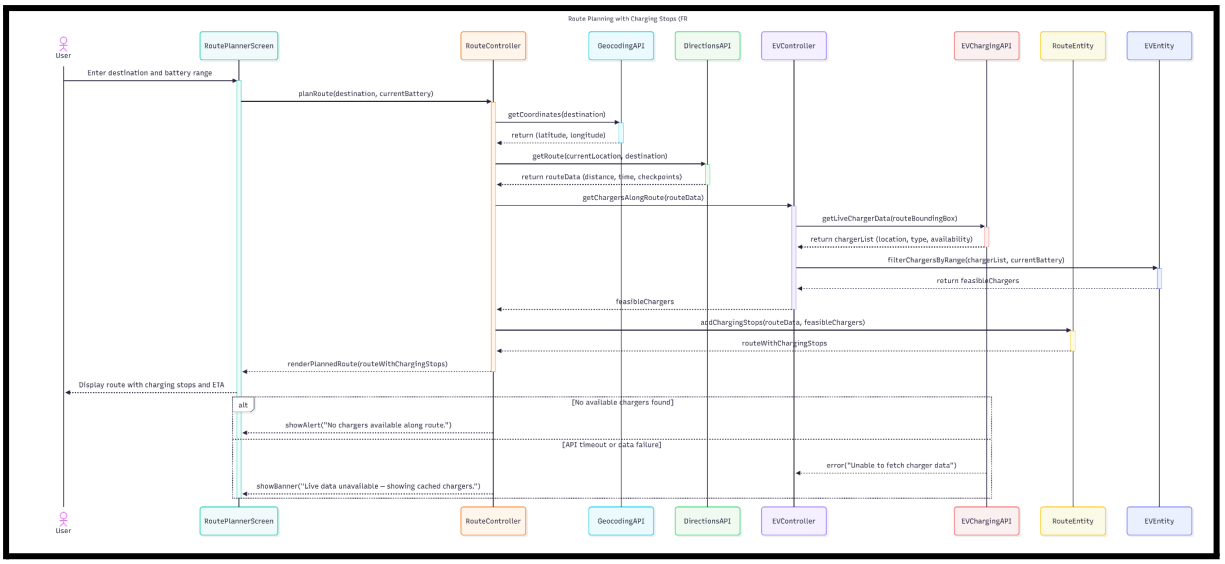
g. EV Smart Integration - Live Charger Availability and Filter

This sequence diagram shows how the system retrieves real-time EV charger availability from government data sources (e.g., LTA DataMall, EMA API) and merges it with the standard carpark dataset. It also shows how users can filter the results by charger type (Type 2, CCS2, CHAdeMO) or charging speed (slow/fast). This corresponds to FR #8-1 (Live EV Charger Availability) and FR #8-2 (Filter by Connector Type & Speed).



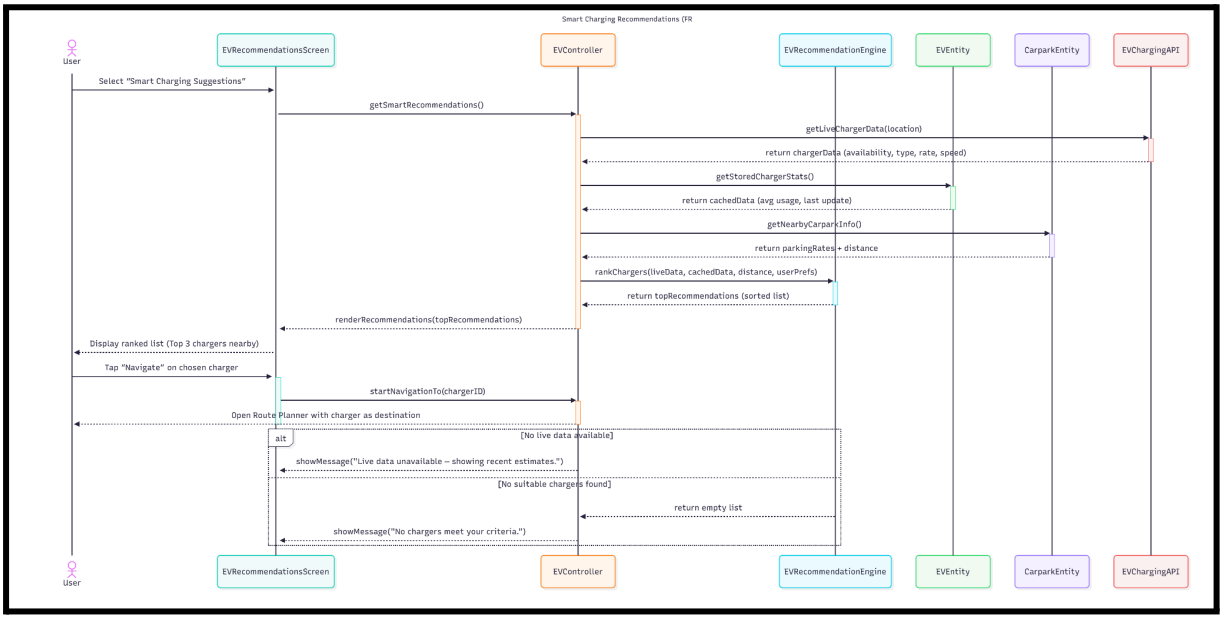
h. Route Planning with Charging Stops

This sequence diagram illustrates how ParkMate assists EV drivers in planning routes that include optimal charging stops along the way. It reflects FR #8-4, where the system uses the destination input, retrieves route data from a Mapping API (e.g., OneMap or Google Directions API), overlays live EV charging station data, and suggests stops based on vehicle range and charger availability.



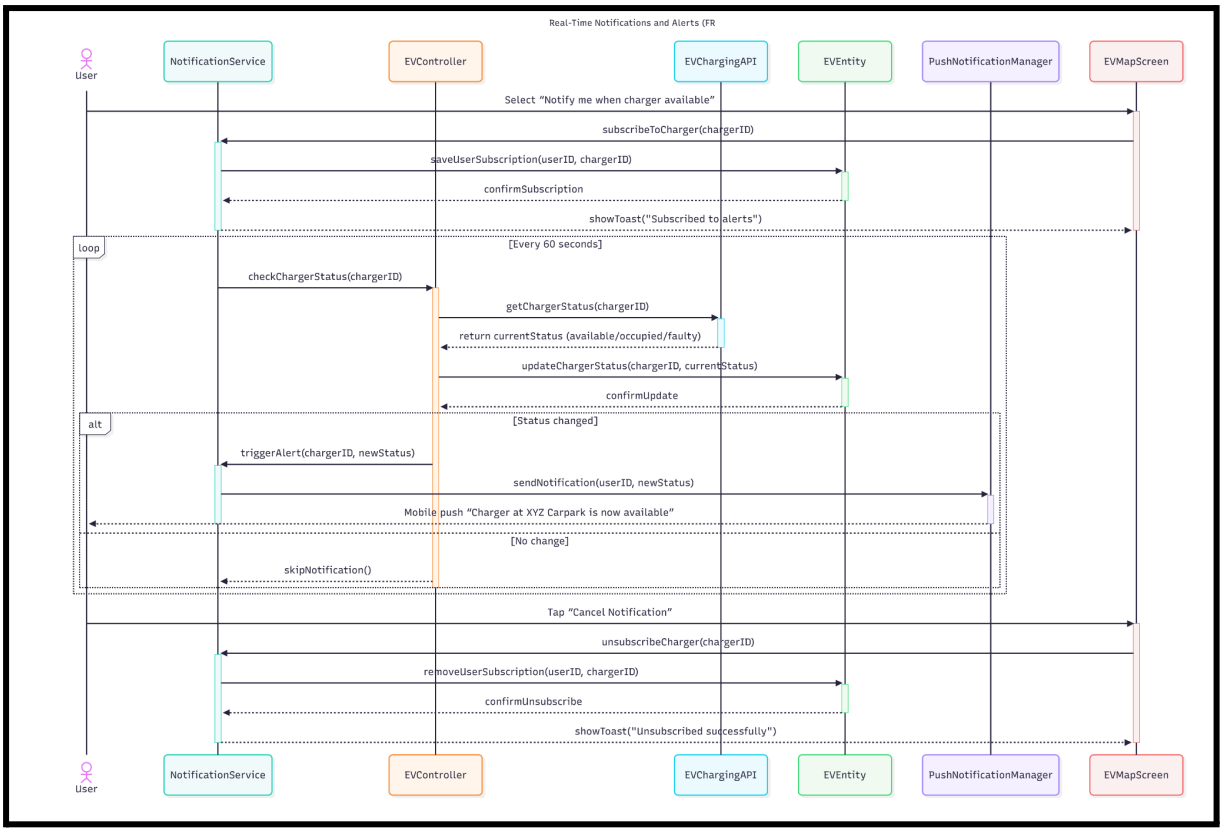
i. Smart Charging Recommendations

This diagram demonstrates how the system gathers EV charger data, applies internal ranking logic based on availability, distance, cost, and user preferences, and displays the top recommended charging options. It showcases how the recommendation engine communicates with data entities and the UI layer.

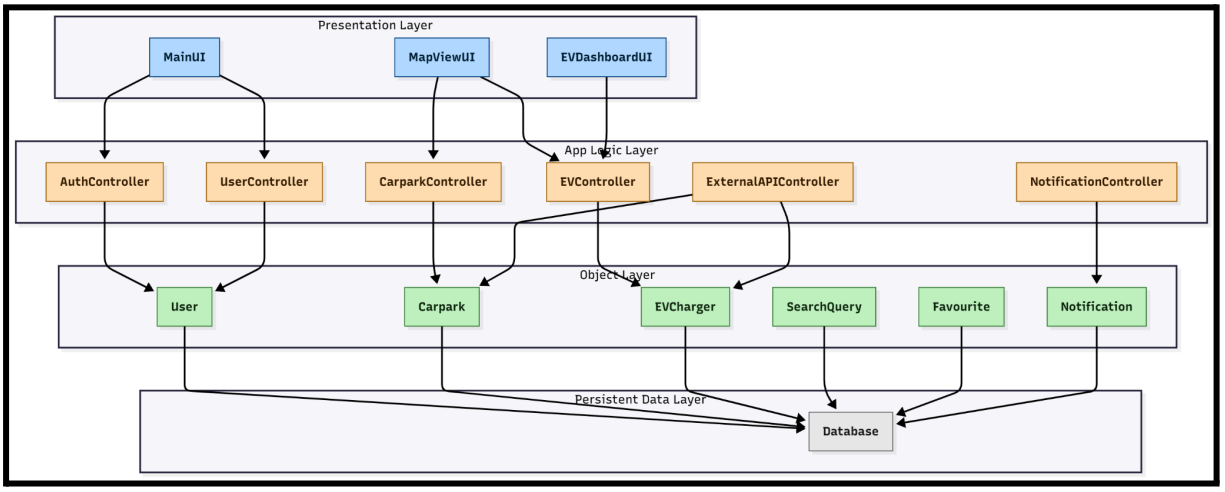


j. Real-Time Notifications and Alerts

This sequence diagram illustrates how ParkMate sends real-time notifications to users about EV charger status changes, for instance, when a selected charger becomes occupied, available, or faulty. It directly maps to FR #8-6 (Real-Time Occupancy Alerts / Notifications) from your documentation, showing background polling, alert triggers, and message delivery to the user interface.



5. System Architecture



a. Presentation Layer

This layer manages all user interactions and screen interfaces. Each interface calls the relevant controller in the App Logic Layer to process requests and return responses. This layer consists of:

1. MainUI
    - Central hub for navigation. Users can access carpark search, favourites, and settings.
2. MapViewUI
    - Displays nearby carparks and EV charging stations on a real-time map. It interacts with CarparkController and EVController to fetch and visualize live data.
3. EVDashboardUI
    - Dedicated dashboard for electric vehicle users. It displays charger availability, session progress, and suggested routes for recharging stops.

b. App Logic Layer

This layer contains the system's controllers, which act as intermediaries between the user interfaces and the data models in the Object Layer. Each controller handles a distinct responsibility (Single Responsibility Principle):

1. AuthController
    - Manages account creation, login, and session handling for users.
2. CarparkController
    - Retrieves carpark details (rates, operating hours, live availability) and manages display logic for MapViewUI.
3. UserController
    - Handles user-related operations such as profile updates and retrieving personal preferences.
4. EVController
    - Fetches EV charging station data (availability, connector type, power rating) and computes optimal EV-friendly routes.
5. NotificationController

    - Triggers user alerts such as “Charger available” or “Parking session ending soon.”
6. ExternalAPIController
    - Integrates with third-party data sources, e.g., LTA DataMall Carpark API, EMA EVSE API, and OneMap Geocoding API, to update live carpark and charger data.

c. Object Layer

The Object Layer defines all the entities (data models) that represent the core objects within ParkMate. These entities are manipulated by the controllers and stored in the database:

1. User
    - Represents user accounts, including email, password hash, preferences, and EV profile settings.
2. Carpark
    - Stores data such as name, address, capacity, rates, and live availability.
3. EVCharger
    - Represents charging stations, including plug type, charging speed (kW), cost, and real-time status.
4. SearchQuery
    - Keeps track of recent user searches, destinations, and filters applied.
5. Favourite
    - Links users to their saved carparks or charging stations.
6. Notification
    - Stores alerts and their delivery status (e.g., pending, delivered).

d. Persistent Data Layer

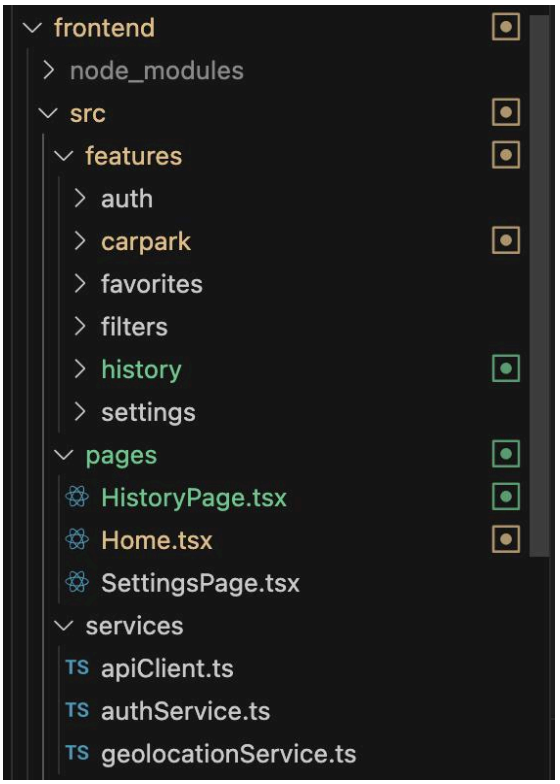
This layer maintains all long-term data storage. It acts as the system’s database backend.

Database:
- Stores all entity objects (User, Carpark, EVCharger, etc)
- Performs CRUD operations for persistent data
- Sync with external APIs periodically to ensure data freshness (carpark lots, charger availability)
- Maintains indexes for faster searches and ensures referential integrity between entities

6. Application Skeleton

Please refer to the source code uploaded in the github repository for the application skeleton.

A. Frontend

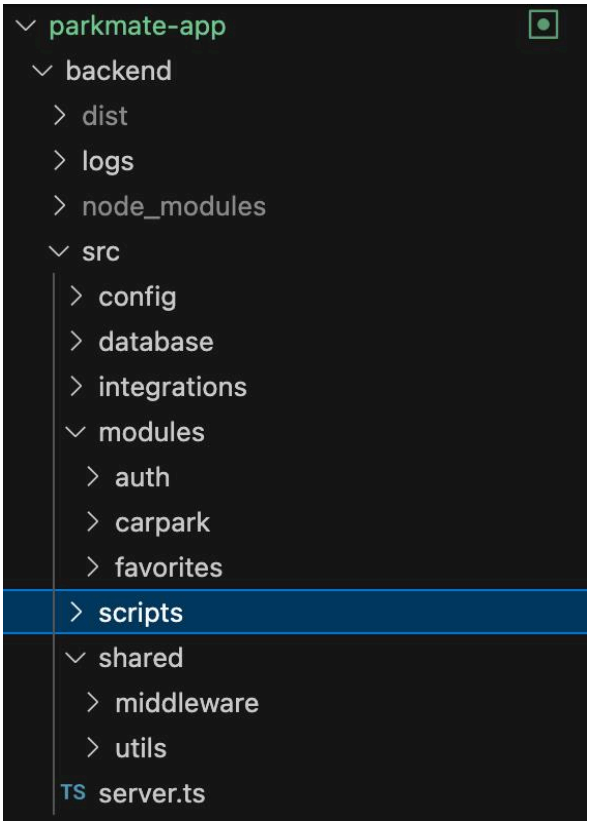


The frontend of ParkMate EV is built using React, TypeScript, and Vite, organized in a modular and scalable structure to ensure clarity, maintainability, and ease of collaboration among developers.Each major feature (e.g., carpark, favorites, filters) is isolated in its own folder to encapsulate related components, logic, and state management, following the feature-driven development (FDD) principle.

Highlights:

- Feature-Based Organization: Each domain (auth, carpark, etc.) is self-contained, allowing parallel development.
- Reusable Components: Shared UI and logic (buttons, cards, API calls) are abstracted under ‘services’ and ‘components’.
- Routing & Pages Separation: High-level route pages are kept distinct from internal feature logic.
- Scalability: New features (e.g., payments, reservations) can easily be added under ‘features’ with minimal disruption.
- Performance: Vite provides lightning-fast builds and hot module reloading for improved developer productivity.

B. Backend



The backend is developed using Node.js, Express, and TypeScript, structured to maintain separation of concerns between configurations, modules, database access, and shared utilities. This architecture supports scalability, modularity, and integration with real-time APIs for carpark and EV data.

Highlights:

- Modular Structure: Each core module (auth, carpark, favorites) encapsulates routes, controllers, and service logic.
- Integrations Folder: Handles external data sources like LTA Datamall, EMA EVSE, and OneMap API for live updates.
- Database & Caching: Combines PostgreSQL (for persistent storage) and Redis (for real-time EV and carpark data).
- Shared Middleware: Common request validation, authentication, and logging ensure consistency across routes.
- Scalable Design: Additional modules (e.g., Reservation, Payment) can be seamlessly integrated for future implementations.

7. Dialogue Map