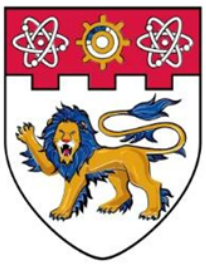


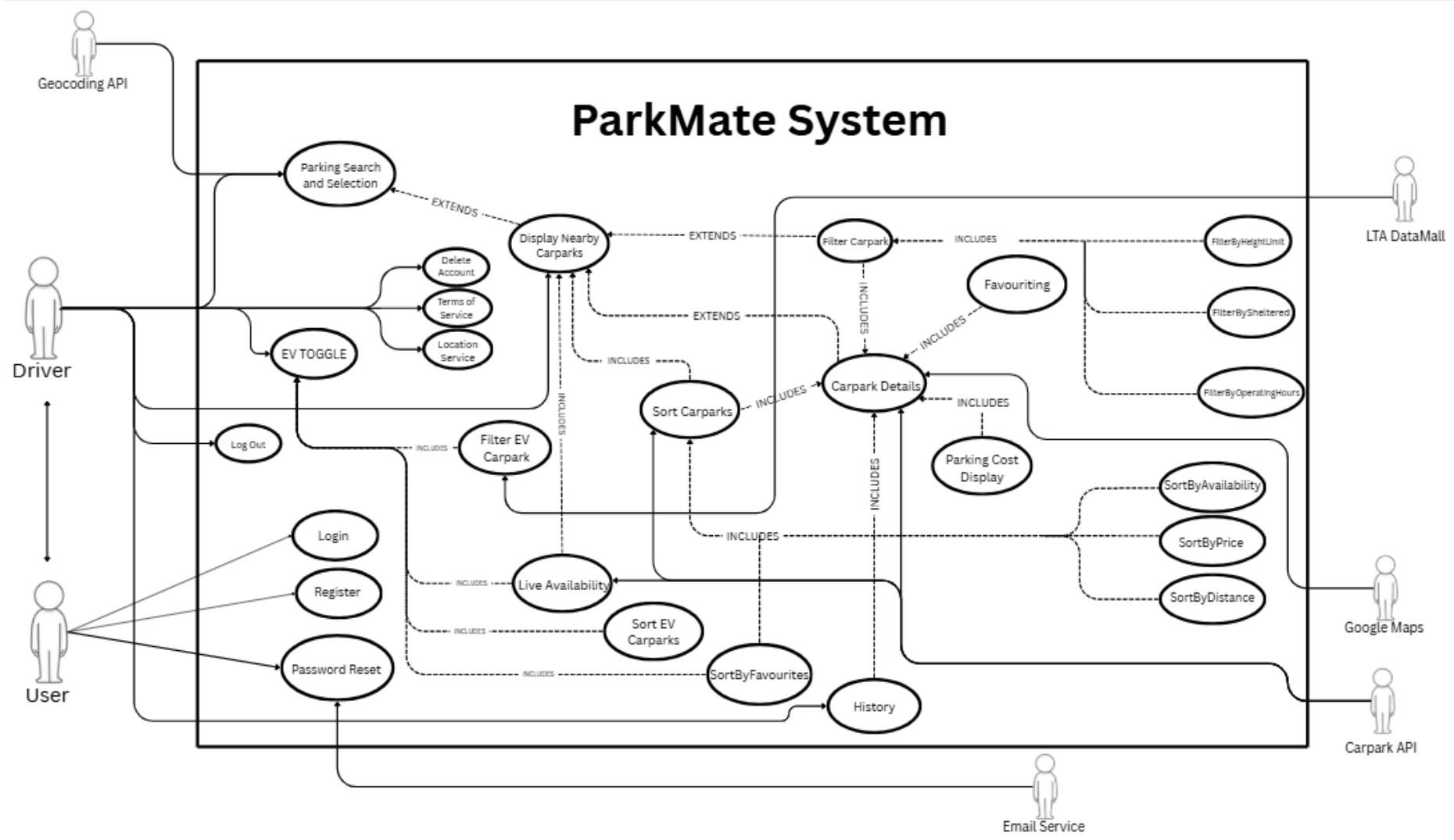
NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

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



2. 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:	Harshil
Date Created:	06 Sept 2025	Date Last Updated:	14 Nov 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 (#2-1) ● User logged in (#2-2) ● Parkmate has access to a geocoding service and a carpark database.
Postconditions:	Success: ● Destination is set as the search location and a 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 the Search Carpark. 2. Parkmate prompts users to input addresses with the “Current Location” option. 3. User input address. 4. Parkmate retrieves location data from the Geocoding API. 5. Parkmate displays a list of location suggestions based on the query. 6. User selects a location from the list. 7. Parkmate sets selection as search destination in google maps.
Alternative Flows:	No location suggestions 1. Display message: No matching locations found. 2. Return to Step 2.
Exceptions:	EX-01: Geocoding API unavailable 1. Parkmate is 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.

	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.
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:	Harshil
Date Created:	06 Sept 2025	Date Last Updated:	14 Nov 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) ● 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 carpark 3. User selects a marker. 4. The selected carpark will be saved to history. 5. Each carpark shows (map view) the following details (#1-8): - Name - Distance - Live availability of carpark slots - Price - Height limit - Last updated time
Alternative Flows:	List view 1. Displays each carpark as a list with the same details as Step 5. 2. Clicking on any list will be saved 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), #1-7 (display carpark details)

Special Requirements:	None
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:	Harshil
Date Created:	06 Sept 2025	Date Last Updated:	14 Nov 2025

Actor:	Driver(user), Parkmate System, Carpark API
Description:	ParkMate fetches and refreshes live carpark availability data at fixed intervals so users see up-to-date information; when data is outdated, the system flags it accordingly.
Preconditions:	● Nearby carparks displayed(#1-2)
Postconditions:	Success: ● Displays updated availability for all visible carparks Failure: ● Shows the 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:	Unable to fetch availability 1. Display last updated availability (cached), marked with its timestamp.
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 a timestamp. EX-02: API rate limit exceeded 1. Too many requests within a short period, API rejects 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:	Auto-refresh every 60s (subject to API constraints)
Assumptions:	AS-1: User logged in (#2-2) 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:	Harshil
Date Created:	06 Sept 2025	Date Last Updated:	14 Nov 2025

Actor:	Driver(user), Parkmate System
Description:	Users can toggle filters to show only the relevant carpark markers/results.
Preconditions:	● Nearby carparks are displayed (#1-2).
Postconditions:	Success: ● Filtered results are displayed instantly on both map view and list view Failure: ● Filters are not applied; the previous (unfiltered or last-known) results remain visible, and ParkMate shows an error/timeout message.
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User opens the filter interface. 2. Parkmate displays available filters. 3. User chooses one or more filter options. 4. ParkMate applies the selected filters and updates the current view (map/list) immediately.
Alternative Flows:	User reset filter 1. User click on reset button 2. Clear all filter selected 3. Parkmate displays non-filtered results. (#1-2) No matching filters for carpark 1. Users filter options unable to display a matching carpark 2. Prompts “No carparks matched” 3. Goes to Step 1
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 (#2-2) 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:	Harshil
Date Created:	08 Sept 2025	Date Last Updated:	14 Nov 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 Failure: ● Filters are not applied; the previous results remain visible, and ParkMate shows “Filtering failed/timed out, showing previous results.”
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User clicks on “height” filter button 2. Parkmate updates the result on the map view
Alternative Flows:	AF-S1: Reset 1. User taps Reset/Clear → all height filters cleared 2. ParkMate shows unfiltered results (#1-2). AF-S2: No matches 1. Zero results 2. ParkMate shows “No carparks match this height” with “Relax filter” action.
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:	Units are meters (m).
Assumptions:	AS-1: User logged in (#2-2) AS-2: User selected a destination (#1-1) AS-3: User on filter interface(#1-4 Step 2)
Notes and Issues:	None

2. Filter Sheltered Parking

Use Case ID:	#1-4-1-2		
Use Case Name:	Filter Sheltered Parking		
Created By:	Yibin	Last Updated By:	Harshil
Date Created:	08 Sept 2025	Date Last Updated:	14 Nov 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 map view. Failure: ● Filters are not applied; the previous results remain visible, and ParkMate shows: “Filtering failed/timed out, showing previous results.”
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User clicks on “Sheltered Parking” filter button 2. Parkmate updates the result on the map view.
Alternative Flows:	Reset Filters 1. User clears filters 2. ParkMate removes the sheltered constraint and shows unfiltered results (#1-2) 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 2 (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 (#2-2) 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:	#1-4-1-3		
Use Case Name:	Filter Operating Hours		
Created By:	Yibin	Last Updated By:	Harshil
Date Created:	08 Sept 2025	Date Last Updated:	14 Nov 2025

Actor:	Driver(user), Parkmate System
Description:	Users can modify Operating Hours filters
Preconditions:	● Nearby carparks are displayed (#1-2).
Postconditions:	Success: ● Filtered results are displayed instantly on the map view. Failure: ● Filters are not applied; the previous results remain visible, and ParkMate shows: “Filtering failed/timed out, showing previous results.”
Priority:	Medium
Frequency of Use:	High
Flow of Events:	1. User clicks on “Overnight / all / Non Overnight” filter button 2. Parkmate updates the result on the current view map
Alternative Flows:	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 2 (choosing filter)
Exceptions:	EX-01: Operating Hours information missing 1. Some carparks do not have operating hours information 2. Parkmate 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 (#2-2) 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:	Harshil
Date Created:	06 Sept 2025	Date Last Updated:	14 Nov 2025

Actor:	Driver(user), Parkmate System, Carpark API
Description:	User sorts the currently displayed carparks by a chosen criterion (Distance, Price, Availability) and order.
Preconditions:	● Nearby carparks displayed (#1-2)
Postconditions:	Success: ● Carparks sorted based on criteria Failure: ● Sorting is not applied; the previous order remains, and ParkMate shows: “Sorting failed/timed out—showing previous order.”
Priority:	Medium
Frequency of Use:	High
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)
Alternative Flows:	Resets to default 1. User taps Reset 2. ParkMate restores default order and removes the sort
Exceptions:	EX-01: Missing data for sorting criteria 1. Some carparks 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 (#2-2) AS-2: User selected a destination (#1-1)
Notes and Issues:	None

f. Parking Cost Details

Use Case ID:	#1-6		
Use Case Name:	Display Parking Cost		
Created By:	Yibin	Last Updated By:	Harshil
Date Created:	06 Sept 2025	Date Last Updated:	15 Nov 2025

Actor:	Driver(user), Parkmate System, Carpark API
Description:	This use case allows the user to view the official parking cost information for a selected carpark. When the user opens a carpark's details page, ParkMate retrieves the relevant tariff rules (base rate, hourly rate, peak/off-peak fees, and EV charging fees if applicable) and displays them clearly in the UI. This feature provides transparency and helps users make informed decisions before selecting a parking location.
Preconditions:	● User has selected a destination (#1-1) ● Nearby carparks displayed (#1-2) ● Tariff data must exist in the ParkMate carpark database (unless unavailable).
Postconditions:	Success: ● Parking cost information is displayed to the user. Failure: ● Parkmate displays "Parking details unavailable."
Priority:	Medium
Frequency of Use:	Medium
Flow of Events:	1. The user selects a carpark from the list or map. 2. The system loads the carpark details page. 3. The system retrieves tariff rules from the database. 4. The system displays the parking cost information, including relevant rates and fees.
Alternative Flows:	None
Exceptions:	EX-01: Database retrieval error 1. Unable to retrieve information from database or database network failure 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: "Pricing details unavailable." 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
Assumptions:	AS-1: User logged in (#2-2) AS-2: User selected a destination (#1-1)
Notes and Issues:	None

g. Carpark Details

Use Case ID:	#1-7		
Use Case Name:	Carpark Details		
Created By:	Yibin	Last Updated By:	Harshil
Date Created:	06 Sept 2025	Date Last Updated:	15 Nov 2025

Actor:	Driver(user), Parkmate System, Carpark API
Description:	Displays detailed information for a selected carpark, including pricing, lot availability, price, 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
Alternative Flows:	Retrieval Failure: 1. Shows warning message “Failed to retrieve data.” Partial Failure: 1. Some sections are missing from the source 2. Renders available sections; for each missing section, shows “Not provided by operator”
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)
Special Requirements:	None

Assumptions:	AS-1: User logged in (#2-2) AS-2: User selected a destination (#1-1)
Notes and Issues:	None

h. Add Favourites

Use Case ID:	#1-8		
Use Case Name:	Add Favourites		
Created By:	Yibin	Last Updated By:	Harshil
Date Created:	07 Sept 2025	Date Last Updated:	15 Nov 2025

Actor:	Driver(user), Parkmate System
Description:	Allows users to add a carpark to their favourites list by clicking on the heart button in the carpark information window (#1-8)
Preconditions:	● Carparks are already displayed(#1-2)
Postconditions:	Success: ● Selected carpark is added to favourites Failure: ● No change to favourites; icon reverts (if optimistically filled); error banner shown.
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User selects a carpark from the list or mapview. 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:	Already Favourited: 1. Server returns 409 Conflict (or client detects duplicate).. 2. System keeps state unchanged; shows non-blocking toast: “Already in Favourites.”
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 references are 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
Assumptions:	AS-1: User already logged in.(#2-2)

	AS-2: User already entered a destination.(#1-1)
Notes and Issues:	None

II. For Functional Requirement #2

a. Create Account

Use Case ID:	#2-1		
Use Case Name:	Create Account		
Created By:	Abdillah	Last Updated By:	Preetham
Date Created:	08 Sept 2025	Date Last Updated:	14 Nov 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:	Low
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:	#2-2		
Use Case Name:	Login		
Created By:	Abdillah	Last Updated By:	Yibin
Date Created:	08 Sept 2025	Date Last Updated:	14 Nov 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:	High
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:	nil
Special Requirements:	• Secure session management.
Assumptions:	User inputs correct credentials.
Notes and Issues:	nil

c. Password Reset

Use Case ID:	#2-3		
Use Case Name:	Password Reset		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	08 Sept 2025	Date Last Updated:	14 Nov 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:	Medium
Flow of Events:	1. User selects “Forgot Password” 2. User has to fill in their registered Email Address. 3. System sends reset link to user’s email. 4. User clicks reset link in the email and redirected back to ParkMate to set a new password. 5. Authenticated Users can also change password after verifying current password
Alternative Flows:	Unregistered Email → System shows “Unregisted Email. Please enter again.”
Exceptions:	Invalid reset link expired.
Includes:	nil
Special Requirements:	● Token expiration for reset link.
Assumptions:	User can access registered email.
Notes and Issues:	nil

d. Delete Account

Use Case ID:	#2-4		
Use Case Name:	Delete Account		
Created By:	Abdillah	Last Updated By:	Preetham
Date Created:	08 Sept 2025	Date Last Updated:	14 Nov 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:	Low
Flow of Events:	1. User goes into Settings 2. User selects “Delete Account.” 3. System prompts confirmation and user has to fill in their password to confirm 4. If confirmed -> Account deleted, favourites/preferences erased.
Alternative Flows:	User fills in the wrong password -> Account is not deleted
Exceptions:	nil
Includes:	nil
Special Requirements:	● Irreversible deletion.
Assumptions:	User intends permanent removal.
Notes and Issues:	nil

e. View Terms of Service & Privacy Policy

Use Case ID:	#2-5		
Use Case Name:	View Terms of Service and Policy		
Created By:	Abdillah	Last Updated By:	Abdillah
Date Created:	08 Sept 2025	Date Last Updated:	14 Nov 2025

Actor:	Driver (User), ParkMate System
Description:	User views TOS and from Profile page.
Preconditions:	User is logged in.
Postconditions:	Content displayed with date.
Priority:	Low
Frequency of Use:	Medium
Flow of Events:	1. User navigates to Settings. 2. User selects TOS/Privacy Policy. 3. System displays content with version/date
Alternative Flows:	None
Exceptions:	nil
Includes:	nil
Special Requirements:	● Must show latest date updated.
Assumptions:	TOS content is updated in system backend.
Notes and Issues:	Consider multilingual support.

f. Log Out

Use Case ID:	#2-6		
Use Case Name:	Log Out		
Created By:	Yibin	Last Updated By:	Yibin
Date Created:	09 Sept 2025	Date Last Updated:	14 Nov 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 taps the “Logout” button. 2. ParkMate invalidates the current user session. 3. ParkMate clears cached data related to the logged-in user. 4. ParkMate redirects the user to the Login page.
Alternative Flows:	None
Exceptions:	None
Includes:	nil
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

III. For Functional Requirement #3

a. Handle External API Failure

Use Case ID:	#3-1		
Use Case Name:	Handle External API Failure		
Created By:	Abdillah	Last Updated By:	Harshil
Date Created:	07 Sept 2025	Date Last Updated:	14 Nov 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). ● (If showing cached data) A previous successful live fetch exists and is stored.
Postconditions:	Success: 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:	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:	#3-2		
Use Case Name:	Location Permission disabled		
Created By:	Abdillah	Last Updated By:	Harshil
Date Created:	07 Sept 2025	Date Last Updated:	14 Nov 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:	If Geocoding API is unreachable or times out, ParkMate displays, “Location service temporarily unavailable, search results may be incomplete or inaccurate.”
Exceptions:	If Geocoding API returns an error or invalid data, System displays, “Could not process the destination address. Please check the input and try again.” If no network/offline, “You’re offline. Enter a destination when you’re back online.”
Includes:	#1-1 (Search Destination), #1-2 (Display Carparks).
Special Requirements:	None
Assumptions:	User manually inputs destination
Notes and Issues:	None

c. Display Refresh Loading State

Use Case ID:	#3-3		
Use Case Name:	Display Refresh Loading State		
Created By:	Abdillah	Last Updated By:	Harshil
Date Created:	07 Sept 2025	Date Last Updated:	14 Nov 2025

Actor:	Driver (User), ParkMate System
Description:	When data is being refreshed (manual pull-to-refresh or scheduled auto-refresh), ParkMate shows a non-blocking loading state and then replaces results atomically. If refresh fails, the app keeps previous results and shows an error.
Preconditions:	Carparks displayed (#1-2), A refresh is triggered
Postconditions:	Success: ● User sees updated results after refresh OR error message if refresh fails Failure: ● Previous results remain visible; a banner/toast shows “Couldn’t refresh” with a Retry action
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 is retrieved, the loading indicator disappears. 4. Updated results shown to user
Alternative Flows:	1. If the user switches away during refresh, the loading state is paused and resumes once the app returns to foreground.
Exceptions:	API failure 1. Keep previous data, show banner “Can’t refresh right now” with Retry Partial Failure 1. Update what succeeded; label affected sections “Live data unavailable”; keep prior availability for a short grace window with “Stale” tag
Includes:	#1-2 Display Carparks, #1-3 Live Availability
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

IV. For Functional Requirement #4

a. Open in Maps

Use Case ID:	#4-1		
Use Case Name:	Open in Maps		
Created By:	Abdillah	Last Updated By:	Preetham
Date Created:	07 Sept 2025	Date Last Updated:	15 Nov 2025

Actor:	Driver (User), ParkMate System, Google Maps Application, Browser
Description:	ParkMate allows users to open a carpark location in their device's Google Maps application.
Preconditions:	User has selected a carpark from the map.
Postconditions:	User is given the choice to use Google Maps through the Google Maps Application, or the Browser
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User opens the carpark to show the details window(#1-8). 2. User selects "Open in Google Maps" 3. ParkMate presents the user with a choice to open Google Maps through the Google Maps Application or through the Browser. 4. User selects the Google Maps Application 5. Routing is calculated through the Google Maps Application
Alternative Flows:	1. User opens the carpark to show the details window(#1-8). 2. User selects "Open in Google Maps" 3. ParkMate presents the user with a choice to open Google Maps through the Google Maps Application or through the Browser. 4. User selects the Browser 5. Routing is calculated through the Google Maps Web App.
Exceptions:	none
Includes:	none
Special Requirements:	None
Assumptions:	Device has internet connectivity
Notes and Issues:	none

V. For Functional Requirement #5

a. View & Manage Carpark Selection History

Use Case ID:	#5-1		
Use Case Name:	View & Manage Carpark Selection History		
Created By:	Yibin	Last Updated By:	Abdillah
Date Created:	09 Sept 2025	Date Last Updated:	14 Nov 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:	Empty History: 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.” Logged out: 1. 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:	#1-7
Special Requirements:	None
Assumptions:	User may be offline; local storage is available
Notes and Issues:	None

VI. For Functional Requirement #6

a. Toggle EV mode

Use Case ID:	#6-1		
Use Case Name:	Toggle EV mode		
Created By:	Abdi	Last Updated By:	Yibin
Date Created:	11 Nov 2025	Date Last Updated:	11 Nov 2025

Actor:	User (EV Driver), ParkMate System
Description:	This use case lets ParkMate change its whole UI to suit EV functionality. The UI colour scheme will be changed to a green colour and the filter functions and sorting navigation tabs will be changed.
Preconditions:	1. User has logged in.
Postconditions:	1. The UI is successfully switched to EV Mode. 2. EV-related filters and navigation options become active
Priority:	Medium-High (Enhances user personalization and usability for EV drivers)
Frequency of Use:	Medium-High
Flow of Events:	1. The user opens the profile menu. 2. The user selects the EV Mode toggle. 3. The system updates the UI and switches to EV Mode.
Alternative Flows:	None
Exceptions:	EX-01: Failure to load EV mode 1. If the system fails to load EV mode resources, the toggle is not applied and an error is displayed.
Includes:	None
Special Requirements:	None
Assumptions:	AS-01: User owns an EV 1. User has an EV and requires to use EV mode
Notes and Issues:	None

b. Filter by Connector Type & Charging Speed

Use Case ID:	#6-2		
Use Case Name:	Filter by Connector Type & Charging Speed		
Created By:	Harshil	Last Updated By:	Preetham
Date Created:	19 Oct 2025	Date Last Updated:	15 Nov 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 (Type 2, CCS 2 (Fast), CHAdeMO, CCS2, CC2). 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. Connector type and power information are included in the dataset
Postconditions:	Map updates to show only stations that match the selected connector type and speed.
Priority:	Medium (Enhances user personalization and usability for EV drivers)
Frequency of Use:	Medium (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 a preferred charging speed 4. The user taps Apply Filters. 5. ParkMate updates the map and list instantly, showing only chargers that meet the criteria. 6. If no stations match, the system displays a message suggesting to clear or adjust filters.
Alternative Flows:	1. To clear filters, The user taps "Reset All." 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 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

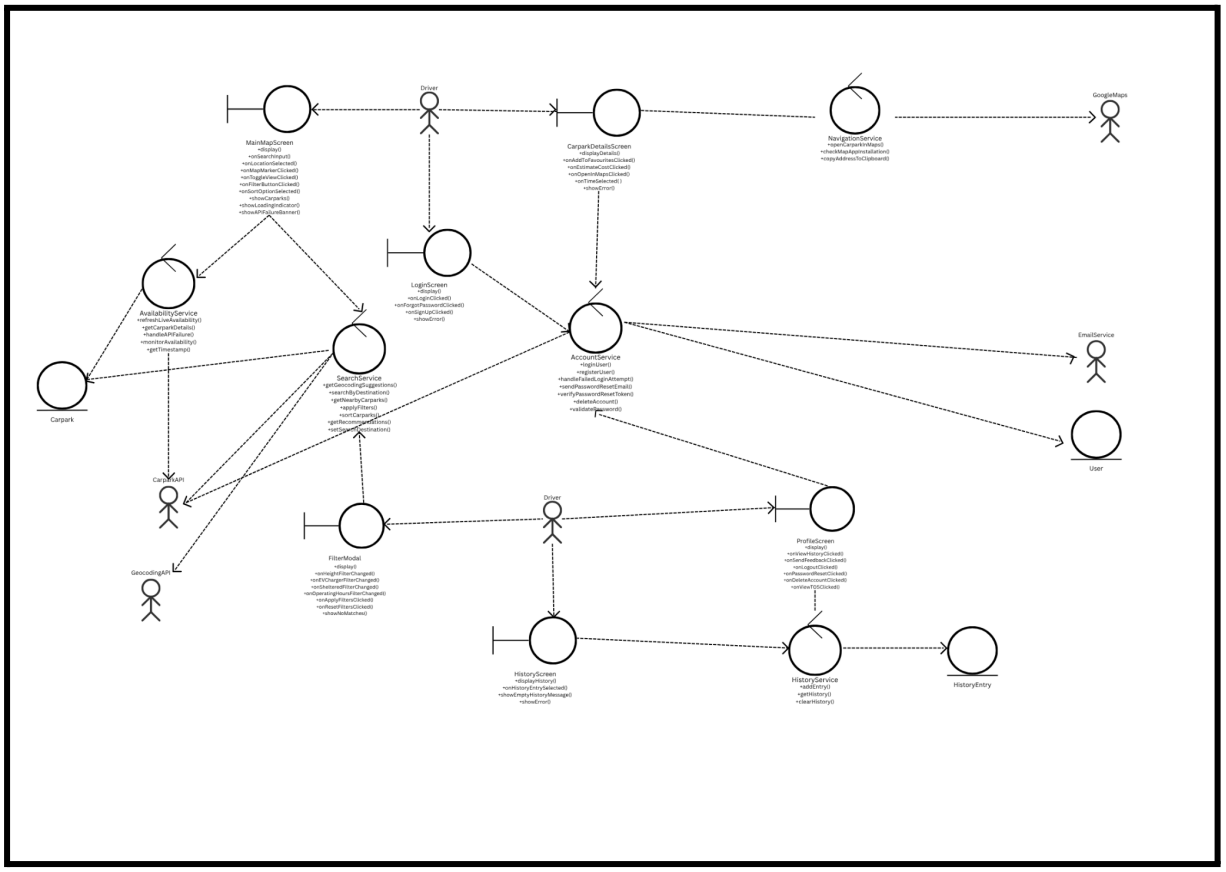
c. EV carpark sorting

Use Case ID:	#6-3		
Use Case Name:	Sort EV Carparks (Favourites & Nearby)		
Created By:	Abdi	Last Updated By:	Yibin
Date Created:	11 Nov 2025	Date Last Updated:	11 Nov 2025

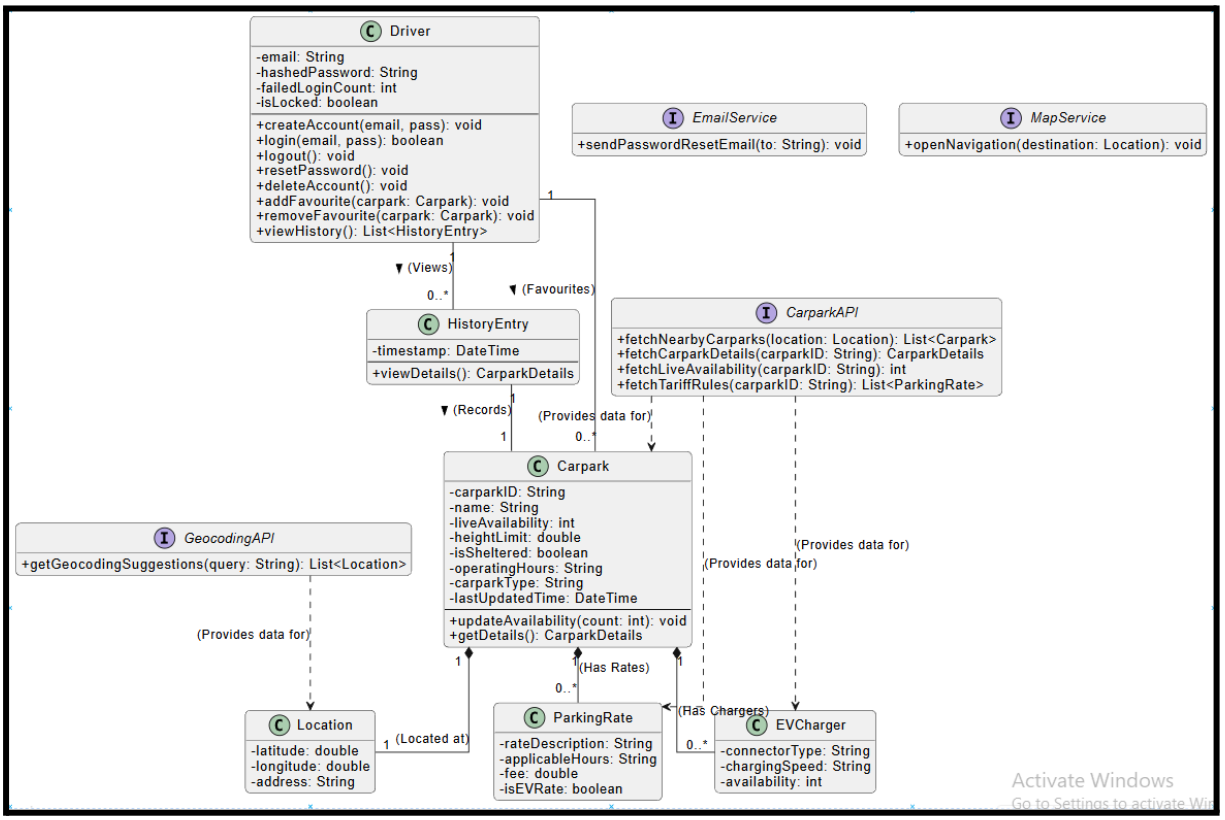
Actor:	User (EV Driver), ParkMate System
Description:	This use case enables the EV driver to sort EV-enabled carparks based on Favourites or Nearby options. When sorting is applied, ParkMate reorganizes the carpark list according to the selected criterion:
Preconditions:	1. User is logged in 2. Location Services are enabled. (required for nearby sorting)
Postconditions:	1. Carpark list is sorted based on the chosen criterion. 2. UI reflects updated sorted order.
Priority:	Medium-High (Enhances user personalization and usability for EV drivers)
Frequency of Use:	Medium-High
Flow of Events:	1. The user navigates to the EV Carpark List screen. 2. The user opens the sorting menu. 3. The user selects either Favourites or Nearby. 4. The system retrieves relevant data (favourites list or location). 5. The system sorts the EV carparks accordingly 6. The UI updates to show the new sorted order.
Alternative Flows:	AF-01: Location Permission Denied 1. If user selects Nearby but location access is disabled, the system prompts the user to enable location permission.
Exceptions:	EX-01: System fails to retrieve favourite list 1. System displays unsorted list and notify user.
Includes:	None
Special Requirements:	SR-01: Sorting under specific time requirement 1. Sorting must complete within 1 second under normal network conditions.
Assumptions:	AS-01: User has at least one favourite carpark
Notes and Issues:	None

3. CLASS DIAGRAM

1. Key Boundary + Control Classes



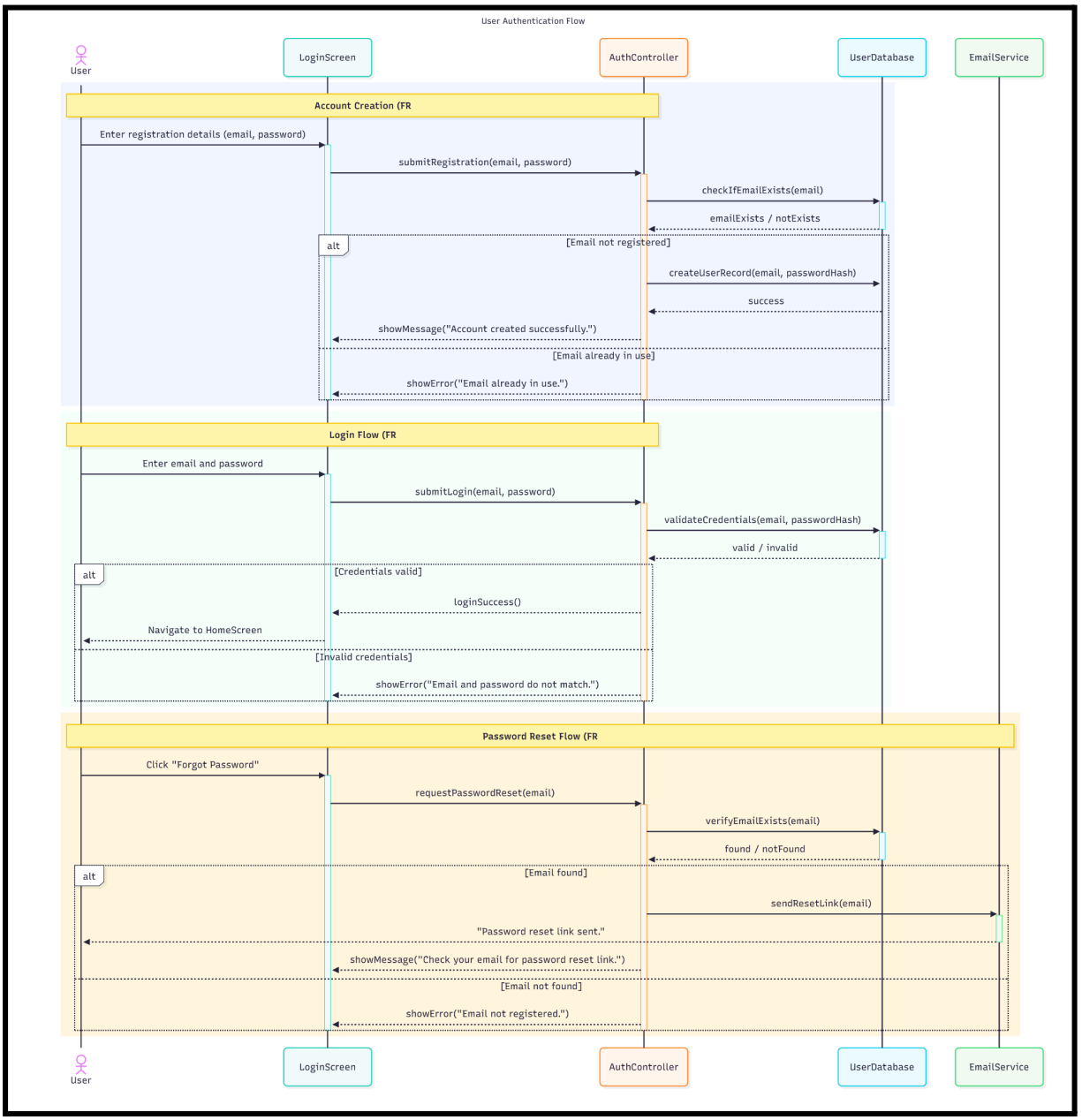
2. Entity Class Diagram



4. Sequence Diagrams

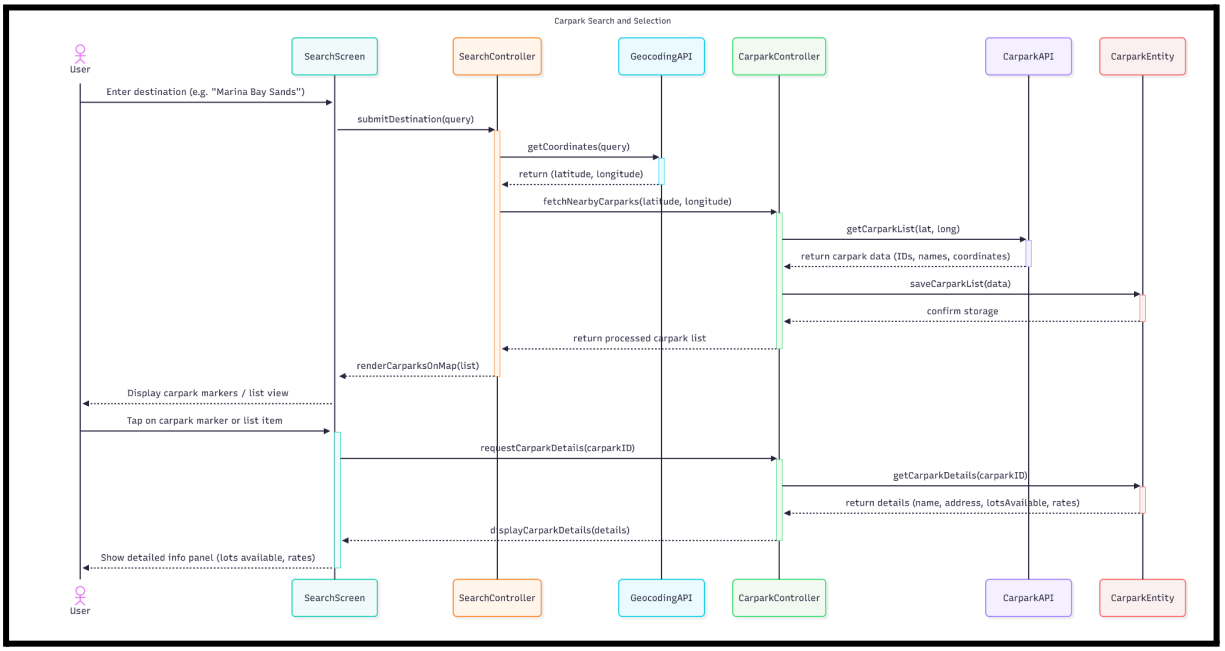
a. User Authentication Flow

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.



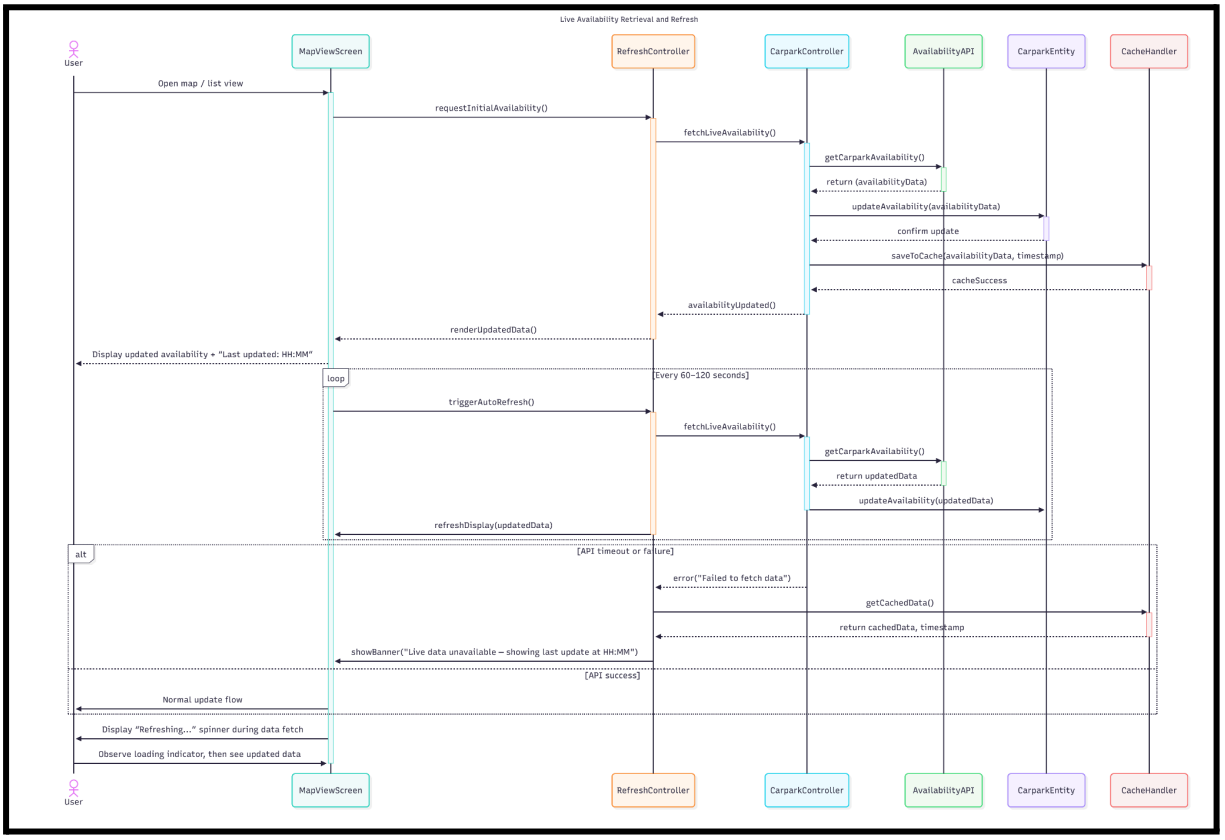
b. Carpark Search and Selection

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.



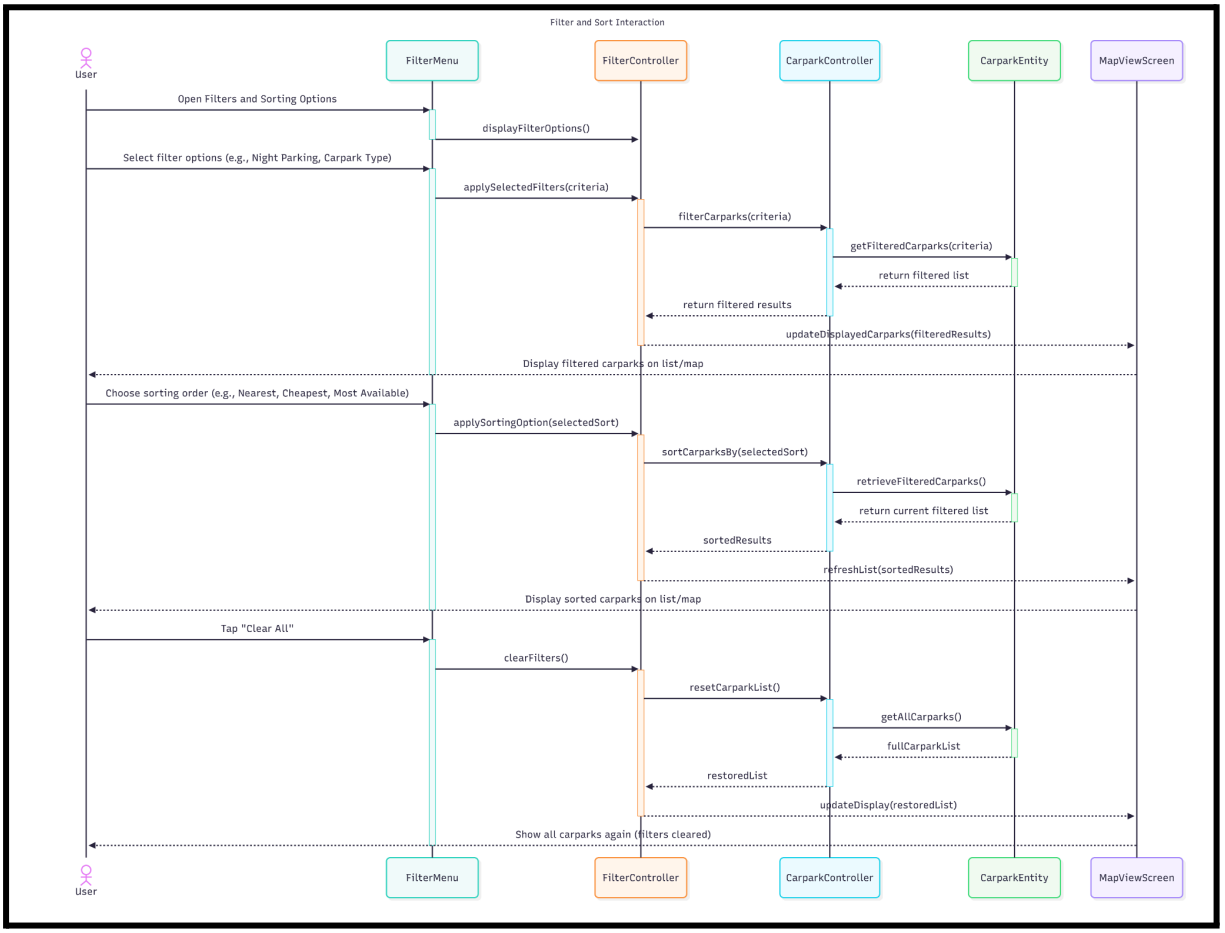
c. Live Availability Retrieval and Refresh

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.



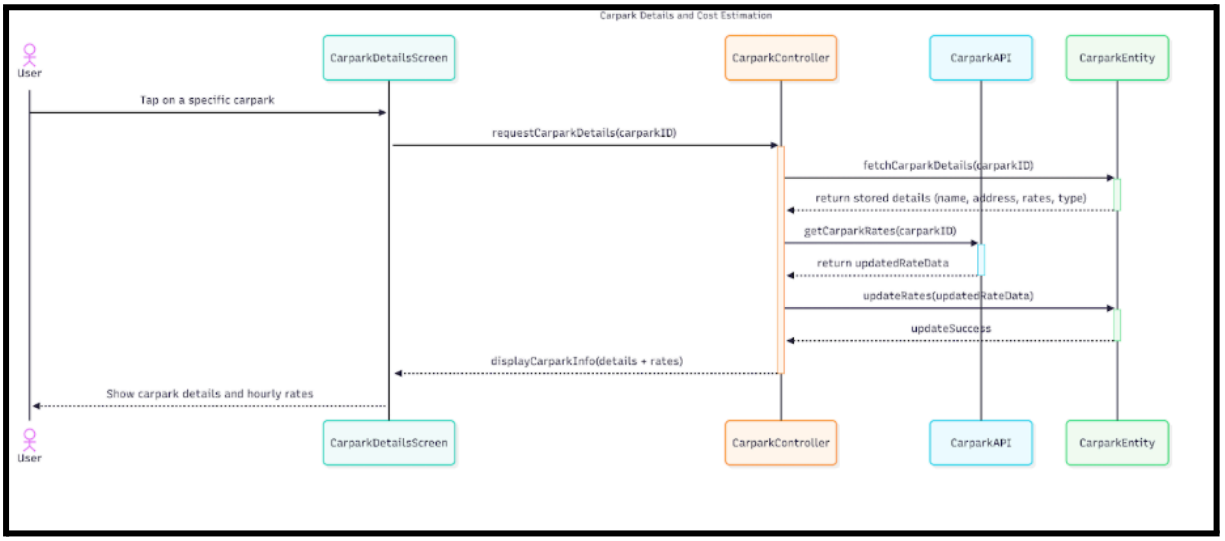
d. Filter and Sort Interaction

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, availability) or sorting order (e.g., nearest, cheapest, most available), which the system processes locally and displays updated results.



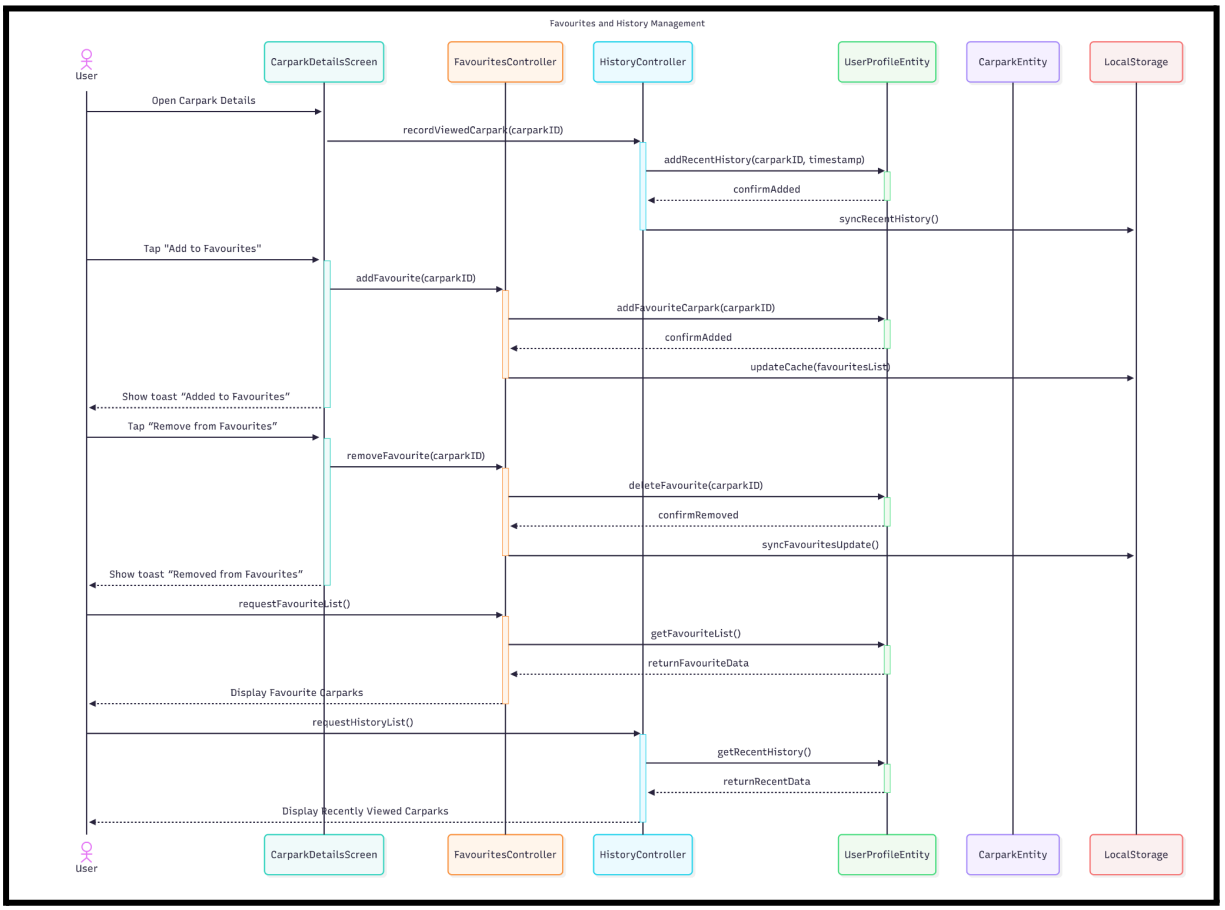
e. Carpark Details

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



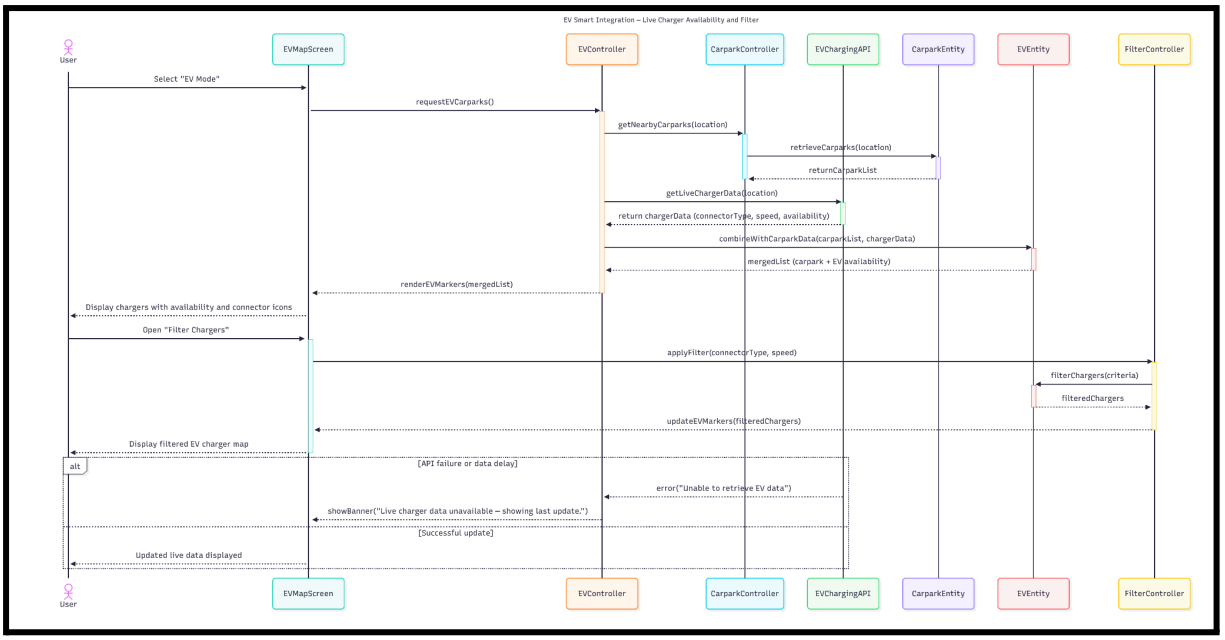
f. Favourites and History Management

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.



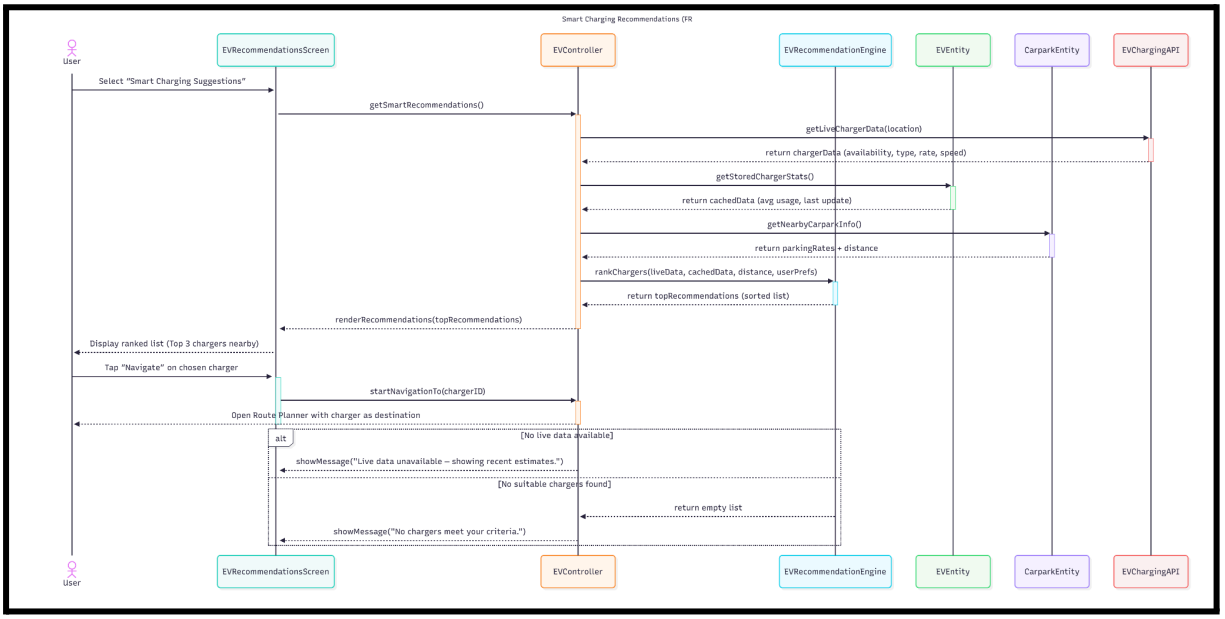
g. EV Smart Integration - Live Charger Availability and Filter

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).

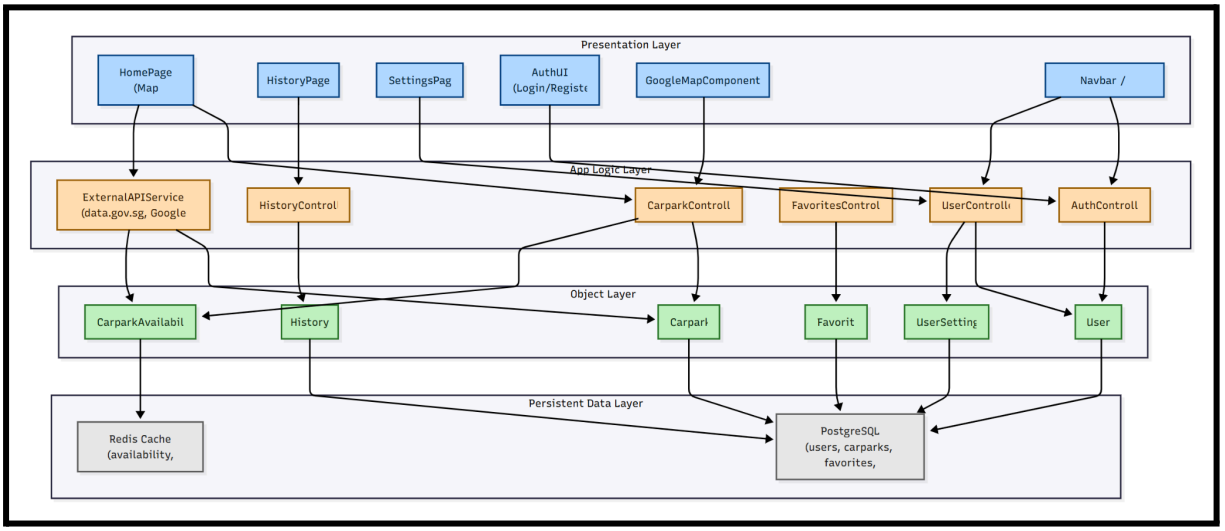


h. Smart Charging Recommendations

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.



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

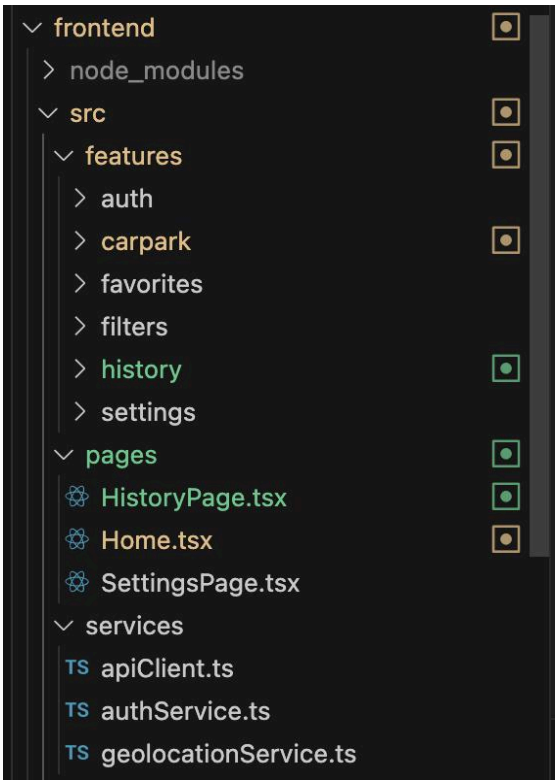
This layer maintains all long-term data storage. It acts as the system’s database backend.

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

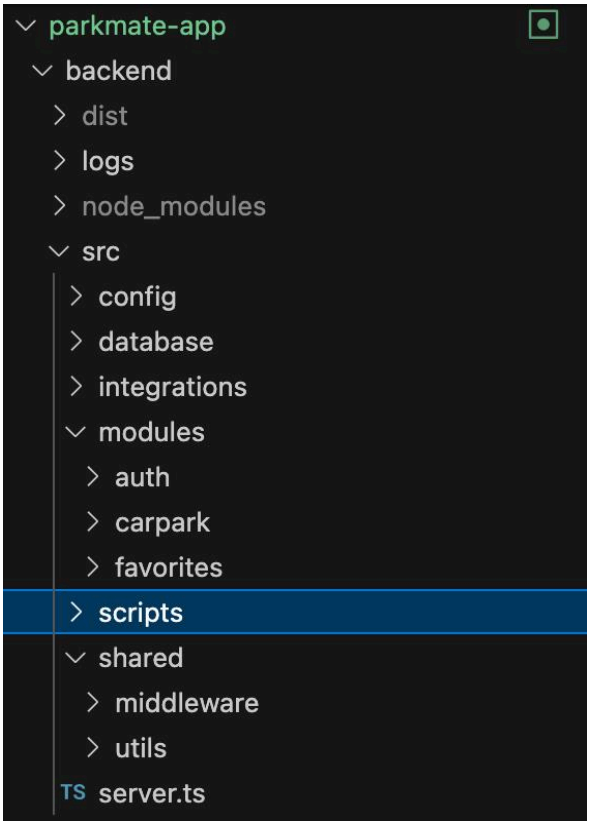


The frontend of ParkMate EV is built using React, TypeScript, and Vite, organized in a modular and scalable structure to ensure clarity, maintainability, and ease of collaboration among developers.Each major feature (e.g., carpark, favorites, filters) is isolated in its own folder to encapsulate related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