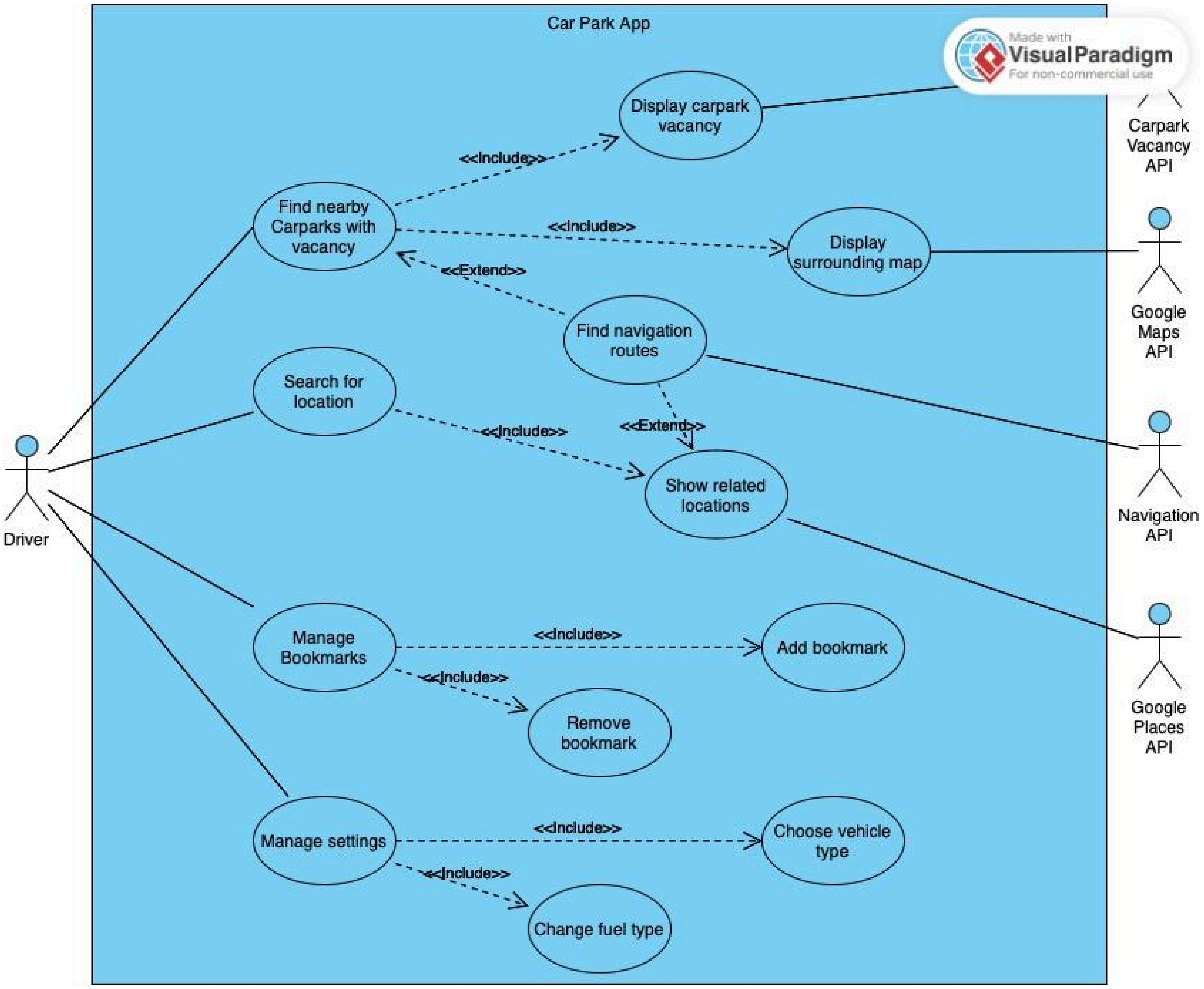
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SC2006-24S1 Software Engineering #Lab 2 Deliverables

Group Member	Matric Number
Quek Jun Siong	U2322145G
Sun Sitong	U2322401J
Jiang Zong Zhe	U2322460F
Tan Yu Xiu	U2322532B
Tan Chong Yao	U2321552F
Solis Aaron Mari Santos	U2322252G

Complete Use Case diagram



Use Case descriptions

Use Case 1: Locating Nearby Carparks

Actor	User (Initiating), Carpark vacancy API, Navigation API
Description	Users can **find nearby carparks with vacancy**
Entry Conditions	1. User has GPS and internet access turned on 2. User is located in Singapore
Exit Conditions	1. Successful exit condition: The app user receives information about nearest carparks and presses the ‘leave’ tab 2. Unsuccessful exit condition: The system informs the user that no relevant locations were found, as well as the possible source of error (such as a lack of internet connection or location permissions)
Priority	High
Frequency of use	High System automatically performs above use case when User initialises the application
Flow of events	1. User opens application and triggers the search for nearby carparks by allowing access to their gps location(need confirmation for this step 1) 2. System will display surrounding carparks around the user’s location 3. System will fetch vacancy, routing, and pricing data from the Carpark vacancy API and navigation API for the carparks displayed. 4. User selects a carpark and chooses to navigate there 5. System provides navigation details to the selected carpark
Alternative flows	1. Carpark nearby gets its last vacancy filled 2. System shows User that the carpark has no more vacancies but still show vacancies for other nearby carparks 3. System returns to normal flow from step #3
Exceptions	1. User does not turn on GPS or navigation services unavailable 2. User does not have Network connectivity 3. Carpark API or Navigation API is down
Includes	1. Display Carpark Vacancy 2. Display Surrounding Map

Use Case 2: Searching Locations

Actor	User (Initiating), Google Maps API
Description	Drivers can **search** for a **location of interest**
Entry Conditions	1. The app user has launched the app and is logged in 2. The app user has access to the internet
Exit Conditions	3. Successful exit condition: The app user receives information about location of interest and presses the ‘leave’ tab 4. Unsuccessful exit condition: The system informs the user that no relevant locations were found with the search query or there was an invalid location input
Priority	High
Frequency of use	Medium User can choose to perform this action soon after opening the app
Flow of events	1. User selects SearchLocation on the UI. 2. System displays consolidated details of most relevant location results in a dropdown - LocationName, Address, DistanceAway, EstimatedArrivalTime, using the *included use case ‘Show list of related locations’.* 3. User selects the chosen location. 4. System shows details of selected location on the map and shows valid navigation routes using the extended use case ‘Find Navigation Routes’.
Alternative flows	1. User enters location that is not relevant to singapore 2. The system fails to find any relevant locations based on the search query 3. System displays a message that informs the app user that no results were found and suggests refining the search 4. User refines the query and the system resumes from the normal flow at step #1
Exceptions	1. User does not have Network connectivity 2. User does not turn on GPS or navigation services unavailable
Includes	1. Show Related Locations

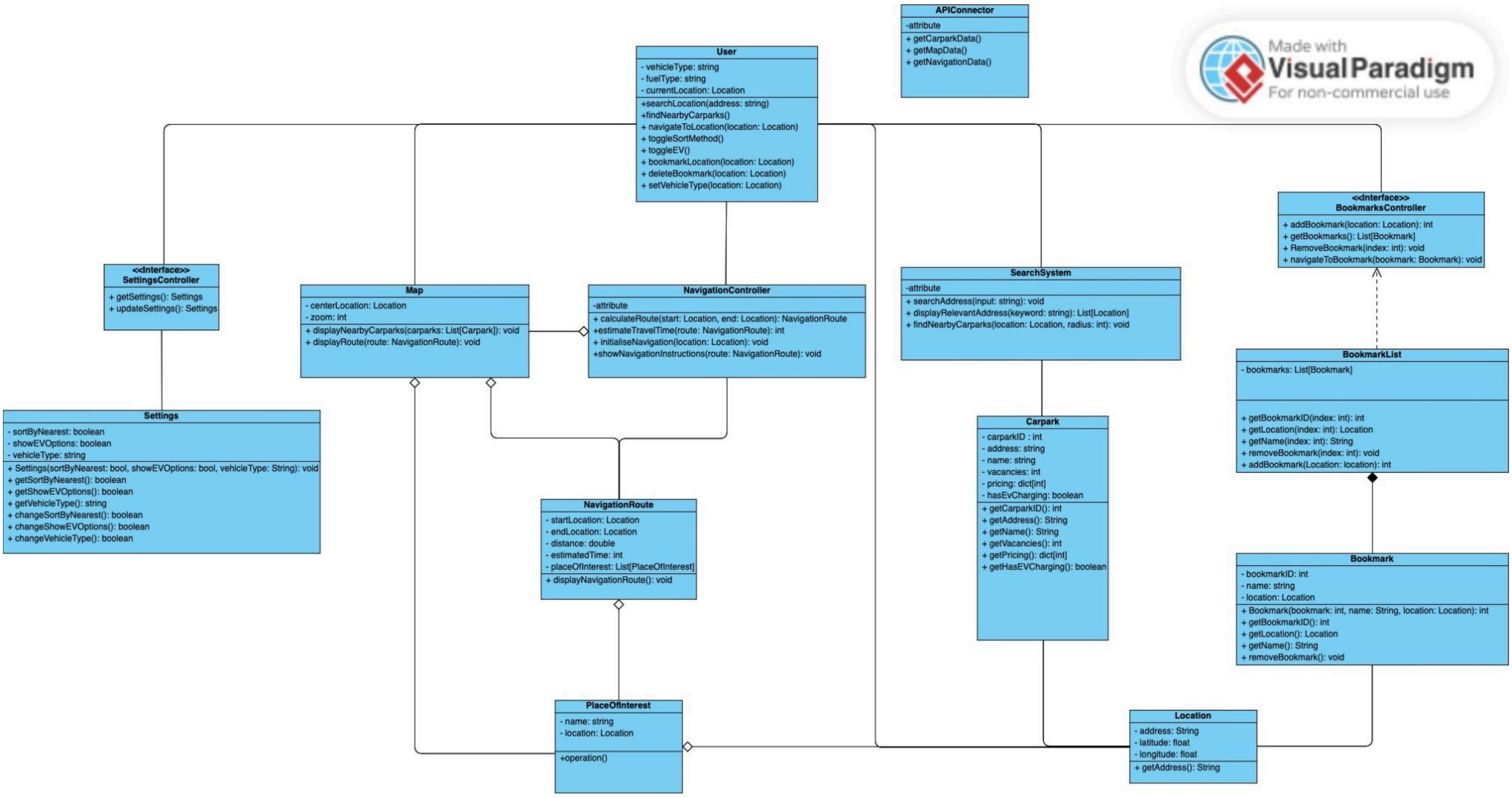
Use case 3: Settings

Actors	User (Initial Actor)
Description	Drivers can **personalise** the app to be more suited for their journeys
Entry Conditions	1. App user must have internet access on their phone 2. App user clicks on “Settings” tab
Exit Conditions	1. App user clicks on “Leave” tab 2. App crashes due to abnormal issues
Priority	Medium
Frequency of Use	Likely only used when the user first adopts the application, which changes averaging once every a few months Important because settings affect the correctness of the program
Flow of Events	1. User selects “Settings” on the UI 2. System displays the settings previously customised by the User. a. If first time entry, or for options not set by the user, System will display the default setting options. This includes Language: English, Vehicle: Car 3. User changes their vehicle type or language according to their preference 4. System updates the app functionalities according to user preference. a. This includes changing the display language, changing the routes and carpark rates and carpark availability and ERP prices for motorcycles and car.
Alternative Flows	1. System encounters an error or network disconnects while updating preferences 2. System informs app user that changes could not be saved due to a system error and prompts the user to try again later 3. System returns to normal flow from step #3 after network and system are restored
Exceptions	1. User does not have Network connectivity
Includes	1. Change Vehicle Type 2. Change Fuel Type

Use case 4: Bookmarking Locations

Actor	User (initial actor)
Description	Users can **bookmark locations** on the app for search convenience
Entry conditions	1. App user must have internet access on their phone 2. App user click on “Bookmark Tab”
Exit conditions	1. User clicks on ‘leave’ tab 2. App crashes due to abnormal issues
Priority	Low
Frequency of use	User may use this feature once every month on average Not as important as it does not affect correctness of the application. It mainly impacts the user-friendliness and efficiency of the application.
Flow of events	1. User selects “Bookmark” tab on the UI. 2. System retrieves the bookmarks saved by the user from server-side database and display it on the UI. 3. User selects a chosen location. 4. User selects “Add Bookmark” button on the UI. 5. System saves the bookmarks added by the user to server-side database and displays the new bookmark on the UI. 6. User selects “Remove Bookmark” button on the UI. 7. System removes the bookmarks removed by the user from the UI and updates the information on the server-side database 8. User selects “Navigate To” button on the UI. 9. System performs the SearchLocation function from step 2) to 4) (see above)
Alternative flows	1. User has no bookmarks 2. System displays message indicating that no bookmarks are available and prompts the user to add new bookmarks 3. System returns to normal flow from step #2
Exceptions	1. User does not have Network connectivity 2. Database error when retrieving bookmark information
Includes	1. Add Bookmark 2. Remove Bookmark

Class diagram of entity classes


APIConnector
-attribute
+ getCarparkData()
+ getMapData()
+ getNavigationData()
Made with Visual Paradigm
For non-commercial use
User
- vehicleType: string
- fuelType: string
- currentLocation: Location
+searchLocation(address: string)
+findNearbyCarparks()
+ navigateToLocation(location: Location)
+ toggleSortMethod()
+ toggleEV()
+ bookmarkLocation(location: Location)
+ deleteBookmark(location: Location)
+ setVehicleType(location: Location)
<<Interface>>
BookmarksController
+ addBookmark(location: Location): int
+ getBookmarks(): List[Bookmark]
+ RemoveBookmark(index: int): void
+ navigateToBookmark(bookmark: Bookmark): void
<<Interface>>
SettingsController
+ getSettings(): Settings
+ updateSettings(): Settings
Map
- centerLocation: Location
- zoom: int
+ displayNearbyCarparks(carparks: List[Carpark]): void
+ displayRoute(route: NavigationRoute): void
NavigationController
-attribute
+ calculateRoute(start: Location, end: Location): NavigationRoute
+estimateTravelTime(route: NavigationRoute): int
+ initialiseNavigation(location: Location): void
+showNavigationInstructions(route: NavigationRoute): void
SearchSystem
-attribute
+ searchAddress(input: string): void
+ displayRelevantAddress(keyword: string): List[Location]
+ findNearbyCarparks(location: Location, radius: int): void
BookmarkList
- bookmarks: List[Bookmark]
+ getBookmarkID(index: int): int
+ getLocation(index: int): Location
+ getName(index: int): String
+ removeBookmark(index: int): void
+ addBookmark(Location: location): int
Settings
- sortByNearest: boolean
- showEVOptions: boolean
- vehicleType: string
+ Settings(sortByNearest: bool, showEVOptions: bool, vehicleType: String): void
+ getSortByNearest(): boolean
+ getShowEVOptions(): boolean
+ getVehicleType(): string
+ changeSortByNearest(): boolean
+ changeShowEVOptions(): boolean
+ changeVehicleType(): boolean
Carpark
- carparkID : int
- address: string
- name: string
- vacancies: int
- pricing: dict[int]
- hasEvCharging: boolean
+ getCarparkID(): int
+ getAddress(): String
+ getName(): String
+ getVacancies(): int
+ getPricing(): dict[int]
+ getHasEVCharging(): boolean
NavigationRoute
- startLocation: Location
- endLocation: Location
- distance: double
- estimatedTime: int
- placeOfInterest: List[PlaceOfInterest]
+ displayNavigationRoute(): void
Bookmark
- bookmarkID: int
- name: string
- location: Location
+ Bookmark(bookmark: int, name: String, location: Location): int
+ getBookmarkID(): int
+ getLocation(): Location
+ getName(): String
+ removeBookmark(): void
PlaceOfInterest
- name: string
- location: Location
+operation()
Location
- address: String
- latitude: float
- longitude: float
+ getAddress(): String

Key boundary classes and control classes

HomeInterface

Variables	Description
MapLocation	Points on the map for the user to toggle.
SearchBar	Text input field for User to input Carpark name to search
SearchBarButton	Button, when toggled triggers the system to search for Carpark based on SearchBar input field
NavigateCarparkButton	Button within a Carpark’s description when a user toggles on a Carpark on the map or a searched Carpark.

HomeController

Method	**Description**
SearchLocation()	Upon toggling on a MapLocation, or toggling the SearchBarButton after inputting text into SearchBar, the system searches for the carpark name, and fetches its description to the user. Under the description, there will be a NavigateCarparkButton.
NavigateLocation()	Upon clicking on NavigateCarparkButton, the system searches for the navigation route and is shown to the user. The user can choose to either complete the navigation journey or terminate in advance.

SettingsInterface

Method	**Description**
SortingByButton	Denotes the current Sorting By preference setting set by the user
IsEVButton	Denotes the current Fuel Type preference setting set by the user
ChangeVehicleTypeButton	Denotes the current Vehicle Type preference setting set by the user

SettingsController

Method	**Description**
ChangeSortingBy()	Upon toggling the SortingByButton, the system changes the setting to the toggled value on the button.
IsEV()	Upon toggling the IsEVButton, the system changes the setting to the toggled value on the button.
ChangeVehicleType()	Upon toggling the ChangeVehicleTypeButton, the system changes the setting to the toggled value on the button.

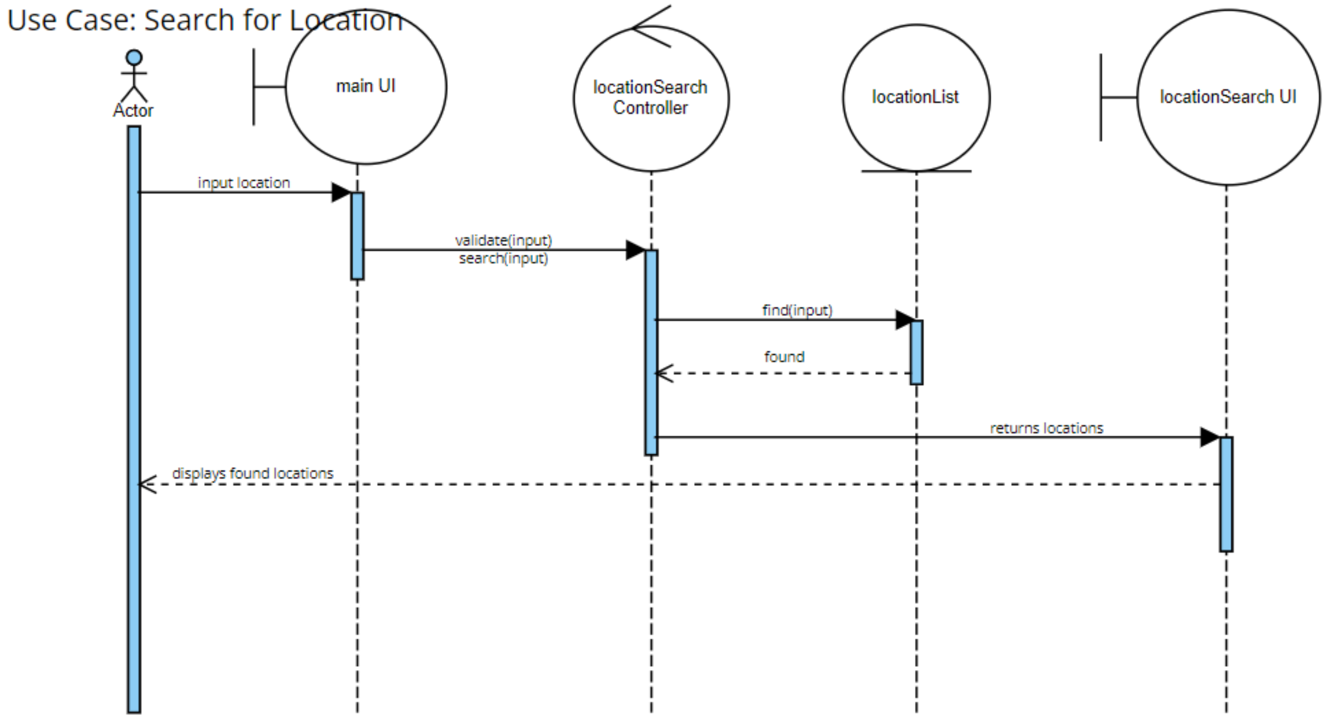
BookmarksInterface

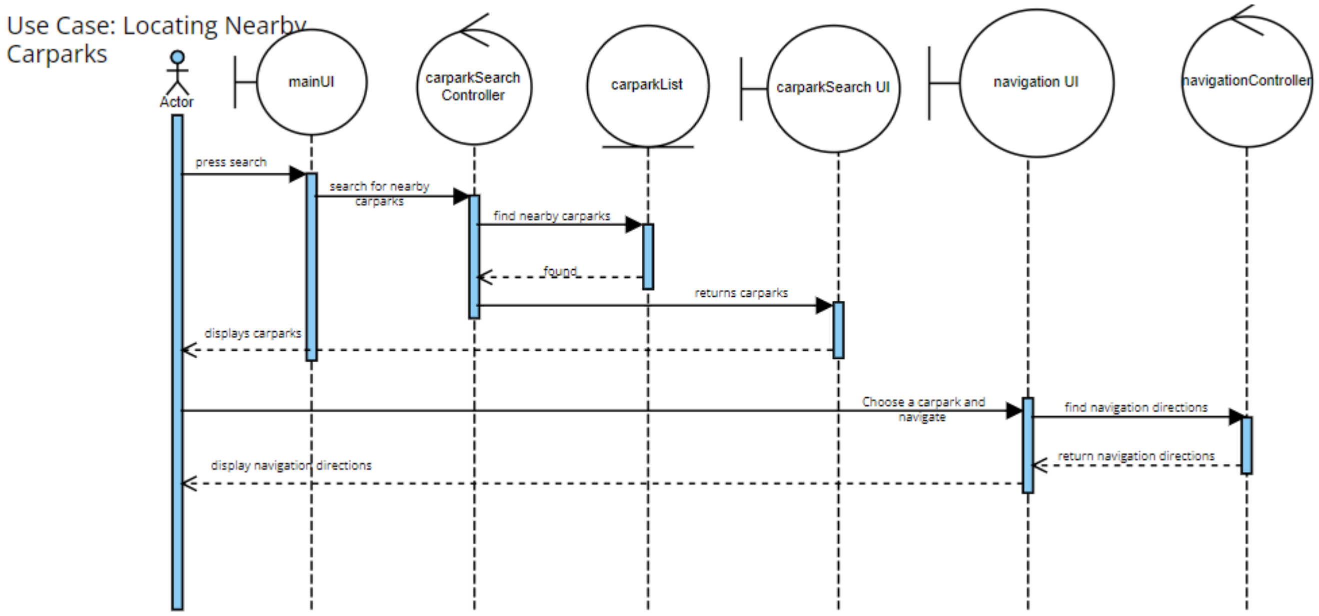
Method	Description
BookmarkObject	Denotes the location of the bookmark, as well as its description
RemoveBookmarkButton	A button for the user to toggle if they wish to remove the chosen bookmark
NavigateBookmarkButton	A button for the user to toggle if they wish to navigate to the chosen bookmark

BookmarksController

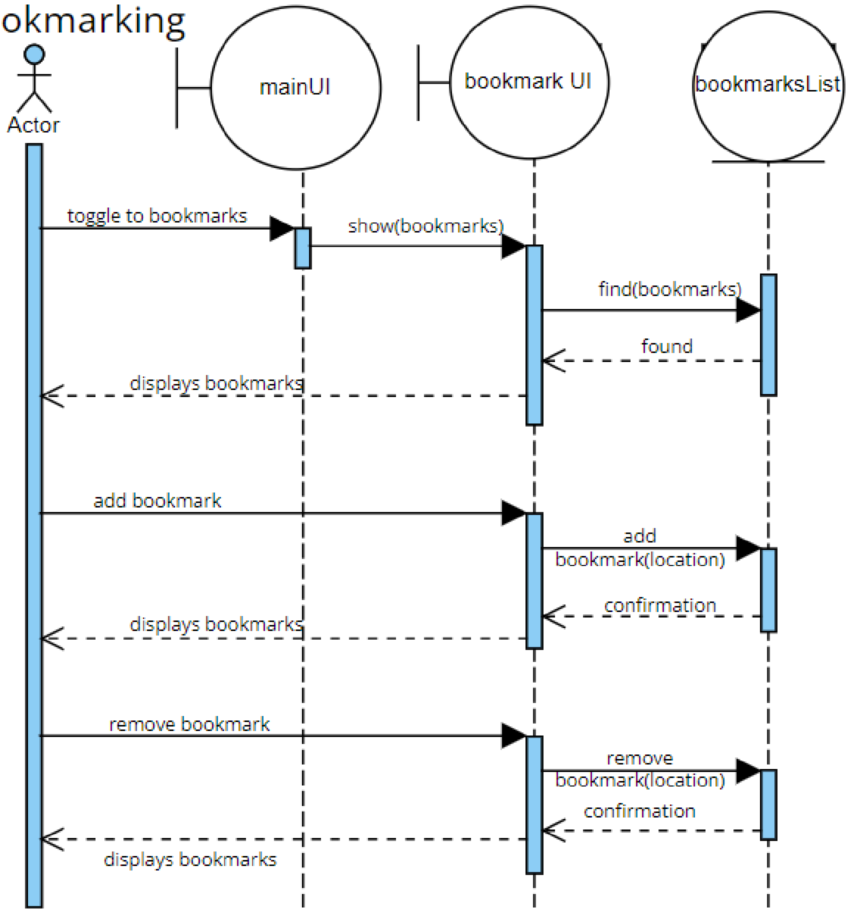
Method	Description
ViewBookmark()	Upon clicking on BookmarkObject, a drop-down description of the bookmark will be shown to the user. Under the description, there will be a NavigateBookmarkButton and a RemoveBookmarkButton.
NavigateBookmark()	Upon clicking on NavigateBookmarkButton, the user will be redirected to the MainInterface, where NavigateCarpark() will be run.
RemoveBookmark()	Upon clicking on RemoveBookmarkButton, the bookmark will be removed from the local system.

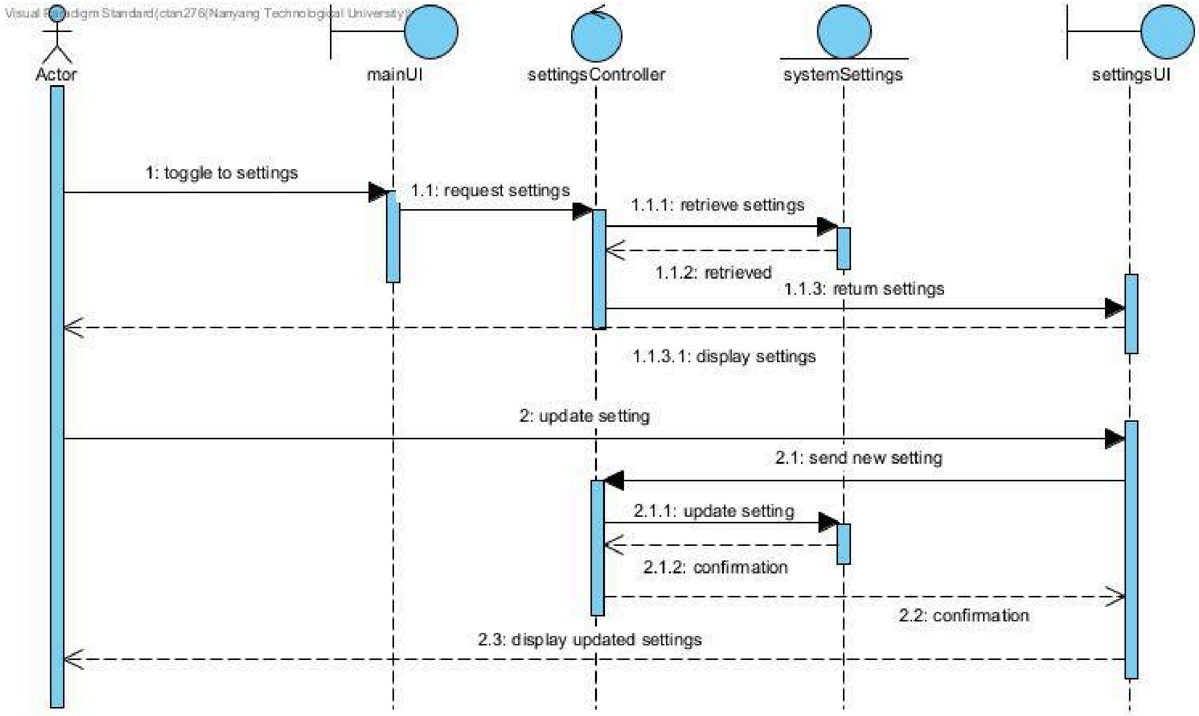
Sequence diagrams



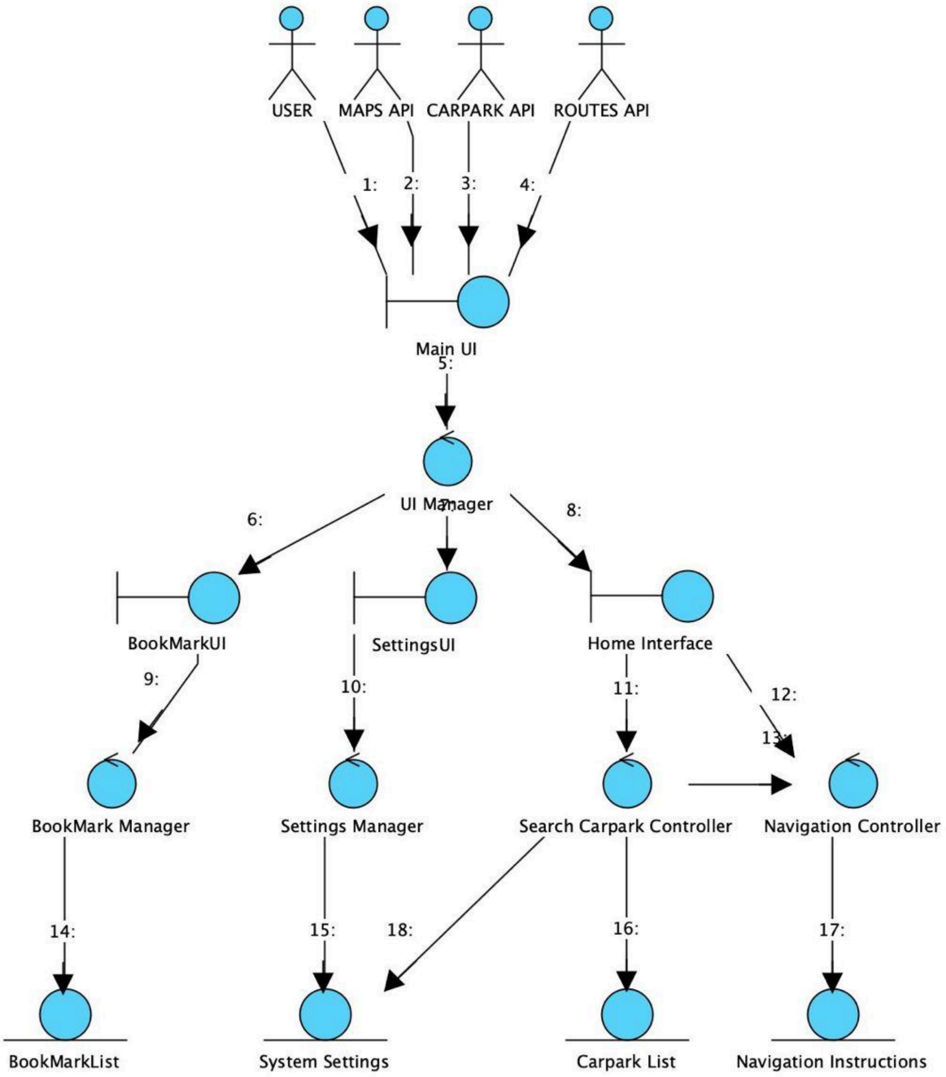

Use Case: Locating Nearby Carparks
Actor
mainUI
carparkSearch Controller
carparkList
carparkSearch UI
navigation UI
navigationController
press search
search for nearby carparks
find nearby carparks
found
returns carparks
displays carparks
Choose a carpark and navigate
find navigation directions
return navigation directions
display navigation directions

Use Case: Bookmarking Locations

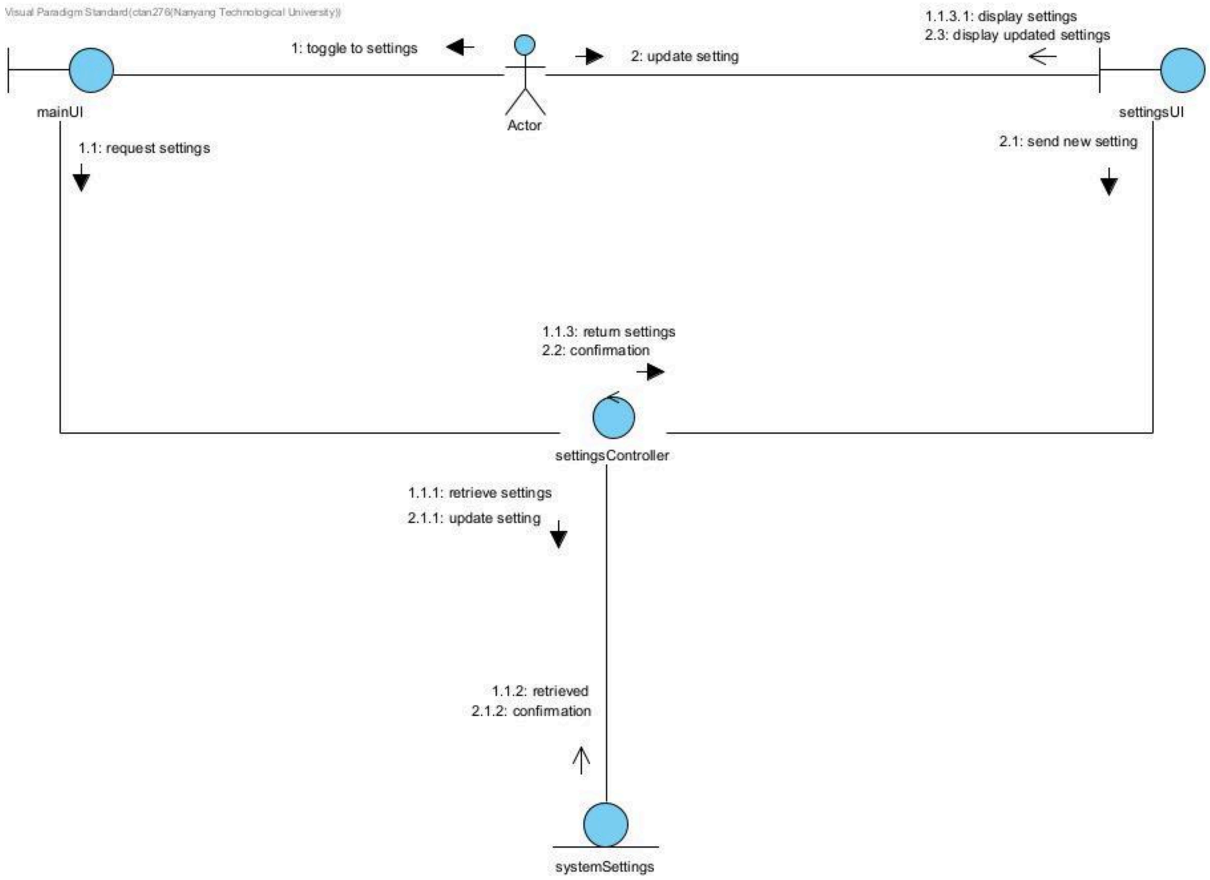




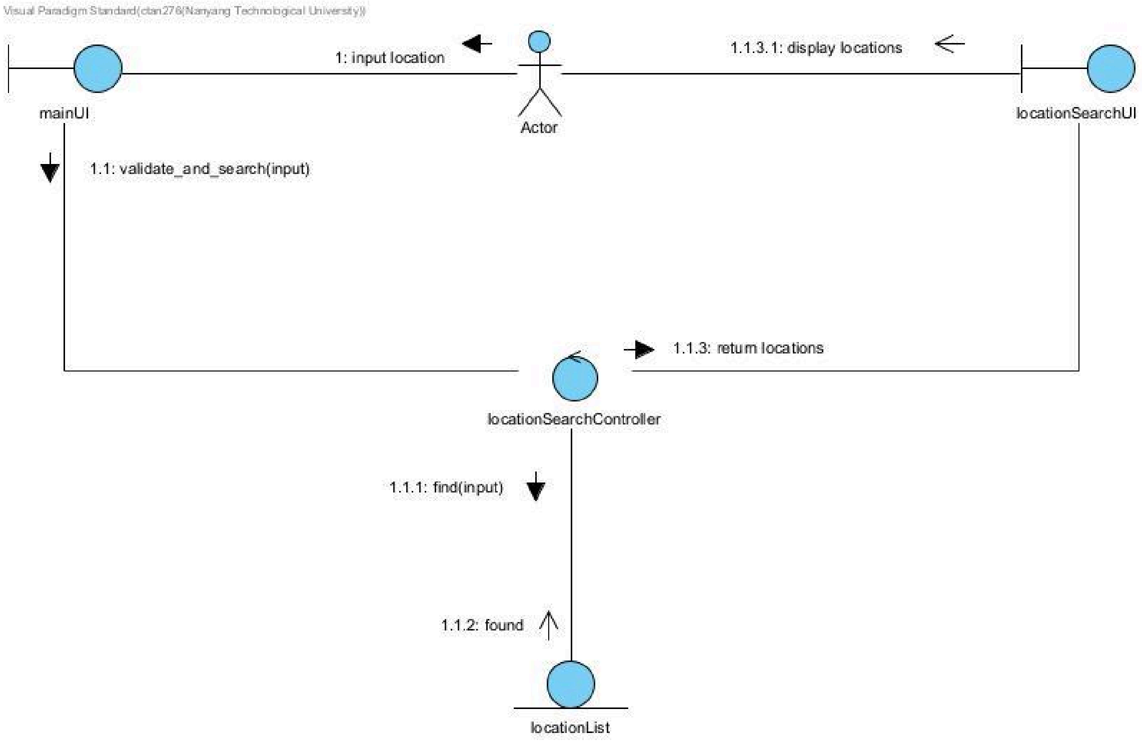
Overall Communication diagram



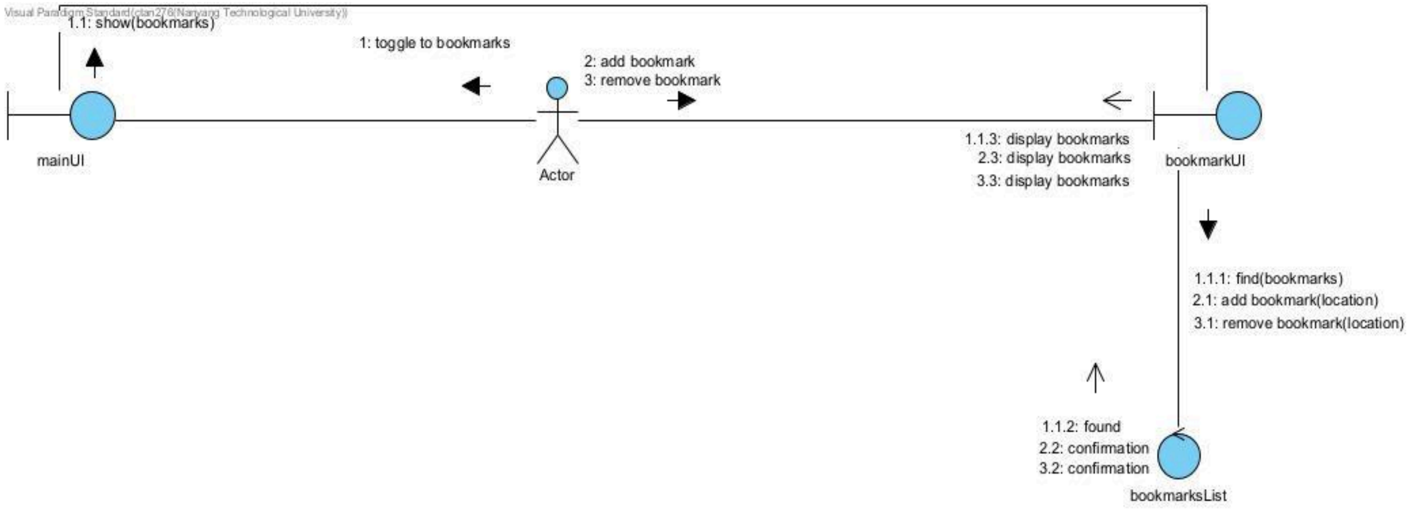
Settings communication diagram



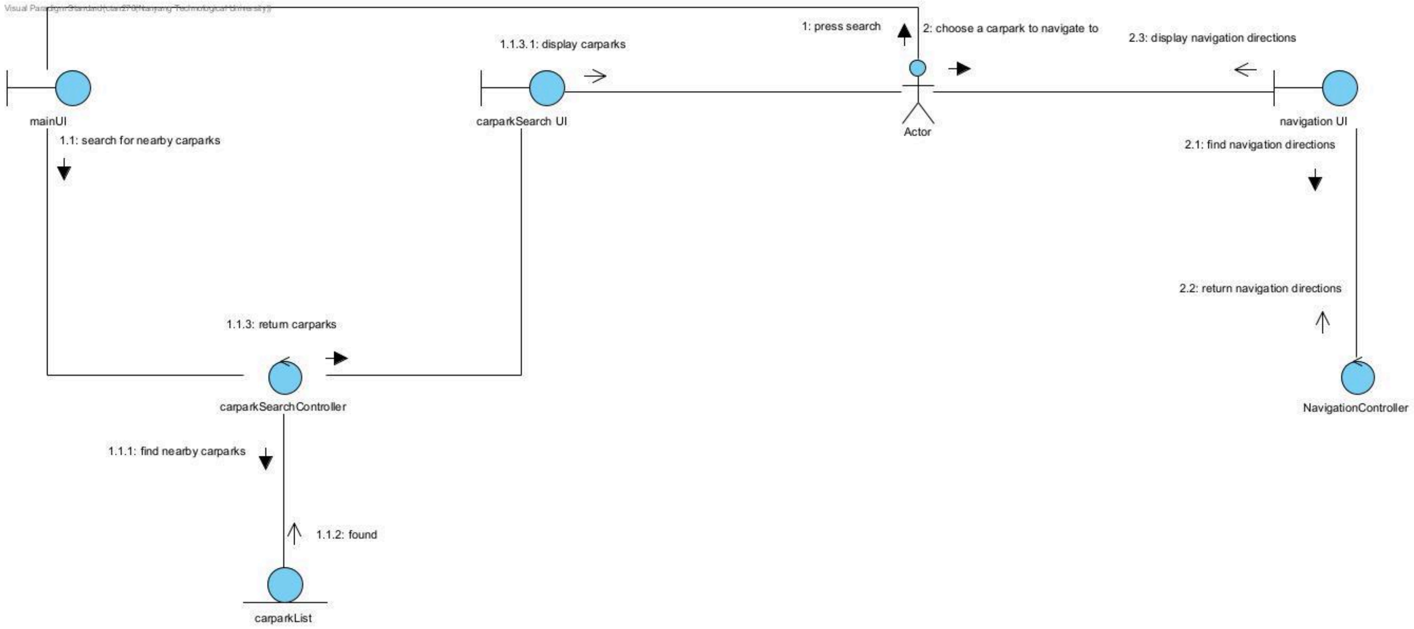
Location Search communication diagram



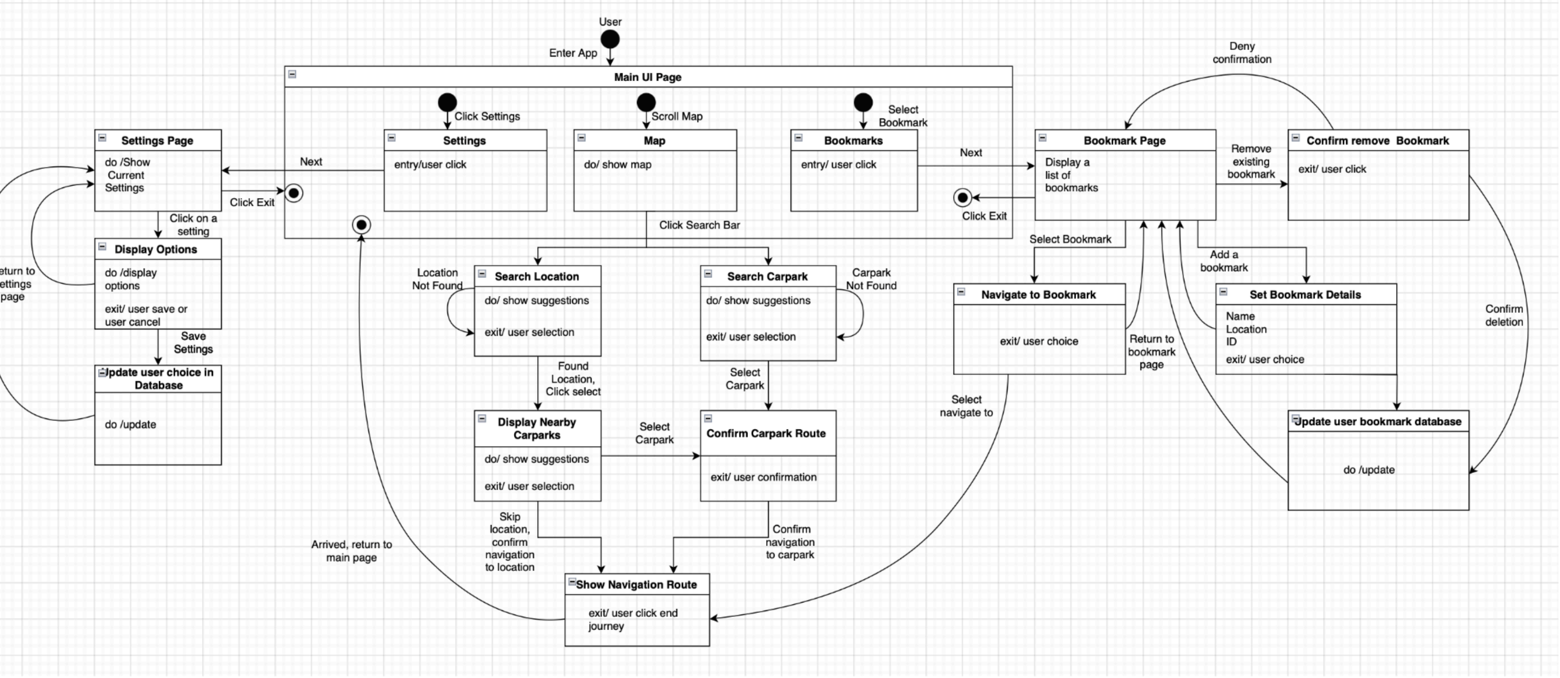
Bookmark communication diagram



Carpark Search communication diagram


1: press search
2: choose a carpark to navigate to
1.1.3.1: display carparks
2.3: display navigation directions
mainUI
carparkSearch UI
Actor
navigation UI
1.1: search for nearby carparks
2.1: find navigation directions
2.2: return navigation directions
1.1.3: return carparks
carparkSearchController
NavigationController
1.1.1: find nearby carparks
1.1.2: found
carparkList

Initial Dialog Map



Appendix: Notes

https://www.cs.sjsu.edu/~pearce/modules/patterns/enterprise/ecb/ecb.htm?ref=blog.ippon.fr#:~:text=Entities%20are%20objects%20representing%20system,commands%20coming%20from%20the%20boundary.

https://home.iscte-iul.pt/~hro/RUPSmallProjects/core.base_rup/guidances/guidelines/analysis_class_7E97273E.html#:~:text=A%20control%20class%20is%20a,encapsulate%20use-case%20specific%20behavior

https://stackoverflow.com/questions/683825/in-uml-class-diagrams-what-are-boundary-classes-control-classes-and-entity-cl