Use Case diagram

Class diagram

Page *(abstract)*
+displayPage() : void

<<boundary>> SidebarPage
+displayPage() : void

<<boundary>> FavouritePage
+FavouritePage()
+displayCarparkInfo() : void
+displayFavouriteCarparks() : void
+displayPage() : void

<<boundary>> SearchPage
-searchResult : Carpark[]
-selectedCarpark : Carpark
+SearchPage()
+SearchPage(searchResult : Carpark[])
+displaySearchResult() : void
+displayMap(map : Map) : void
+getCarparkDetail(carpark : Carpark) : void
+handleSearch(searchLocation : string) : Carpark[]
+setSearchResult(searchResult : Carpark[]) : void
+getSearchResult() : Carpark[]
+displayPage() : void

<<boundary>> DetailsPage
+getCarparkRates() : double
+getCarparkNum() : string
+getCarparkAddress() : string
+displayCarparkRate() : void
+displayCarparkNum() : void
+displayCarparkAddress() : void
+displayCarpark() : void
+displayPage() : void

<<boundary>> NavigatePage
+NavigatePage()
+displayRoute(route : Route) : void
+displayMap() : void
+displayPage() : void

<<control>> FavouritesHandler
+FavouritesHandler()
+getFavourites() : Carpark[]
+addFavouriteCarpark(carpark : Carpark) : void
+removeFavouriteCarpark(carpark : Carpark) : void

<<control>> SelectionHandler
+SelectionHandler()
+selectCarpark() : void
+unselectCarpark() : void

<<control>> SearchHandler
-searchLocation : string
-nearbyCarparks : Carpark[]
+SearchHandler()
+SearchHandler(searchLocation : string)
+setSearchLocation(searchLocation : string) : void
+getSearchLocation() : string
+search() : void
+addToNearbyCarparks() : void
+sortNearbyCarparks() : void

<<control>> SortHandler
-sortSelection : string
+SortHandler()
+setSortSelection(sortSelection : string) : void
+applySort() : void

<<control>> CarparkHandler
+CarparkHandler()
+retrieveCarparkDetails() : void

<<control>> RouteHandler
-givenRoute : Route
+RouteHandler()
+setRoute(route : Route) : void
+removeRoute() : void
+setLocationPin() : void
+setCarparkLocationPin() : void
+removeLocationPin() : void
+removeCarparkLocationPin() : void
+getRoute() : Route
+computeRoute() : void

<<entity>> UserInfo
-favourites : Carpark[]
-userLocation : Location
-sessionID : int
+UserInfo()
+getFavourites() : Carpark[]
+addFavouriteCarpark(carpark : Carpark)
+getUserLocation() : Location
+setUserLocation(location : Location) : void
+getSessionID() : int

<<control>> URAAPI
-apiKey : string
+URAAPI()
+setApiKey(apiKey : string) : void
+getApiKey() : string
+retrieveCarparkRates() : double
+getCarparkAddress() : string
+retrieveVehicleType() : void
+getVehicleType() : void
+getUserLocation() : string

<<entity>> Carpark
-carparkNum : string
-location : Location
-capacity : int
-availability : int
-lotType : char[]
+Carpark()
+getCarparkNum() : string
+getLocation() : Location
+getCapacity() : int
+getAvailability() : int
+updateAvailability(availability : int) : void

<<entity>> CarparkList
-carparkList : Carpark[]
+CarparkList()
+getCarparkList() : Carpark[]
+setCarparkList() : void

<<control>> DataGovAPI
-lastUpdate : DateTime
-apiKey : string
+DataGovAPI()
+DataGovAPI(apiKey : string)
+setApiKey(apiKey : string) : void
+getApiKey() : string
+updateTime(dateTime : DateTime) : void
+getLastUpdate() : DateTime
+retrieveApiAvailability(carparkList : CarparkList) : void

<<control>> GoogleMapAPI
-map : Map
-apiKey : string
+GoogleMapAPI(map : Map)
+GoogleMapAPI(map : Map, apiKey : string)
+getMap() : Map
+setApiKey(apiKey : string) : void
+getApiKey() : string
+createLocation(coords : double[]) : Location
+getUserLocation() : double[]
+computeDistanceBetween(source : double[], target : double[]) : double
+isInRadius(radius : boolean, source : double[], target : double[]) : boolean
+searchLocation(name : string) : double[]
+markLocation(location : Location) : void
+navigateToPoint() : void
+getRoute() : Route[]
+displayMap() : void

<<control>> GPSModule
-usePreciseLocation : boolean
-permissionGranted : boolean
+GPSModule()
+getPermission() : boolean
+getUserLocation() : double[]

<<control>> NotificationsHandler
-notification : string
+NotificationsHandler()
+createNotification(carpark : Carpark) : string
+sendNotification() : void

<<control>> LocationFormatConverter
+getCoordinates()
+conversionFormat() : void

<<entity>> Location
-latitude : double
-longtitude : double
-address : string
-streetName : string
+Location()
+getLatitude() : double
+getLongtitude() : double
+setLatitude(lat : double) : void
+setLongtitude(long : double) : void

<<entity>> Map
-zoomLevel : int
-markers : Location[]
+Map()
+addMarker(marker : Location) : void
+removeMarker() : void
+displayMarkers() : void
+showRadius(radius : float) : void
+displayMap() : void
+setZoomLevel(level : int) : void

<<entity>> Route
-initialLocation : Location
-endLocation : Location
-junctions : Location[]
-distance : float
-estimatedTime : Time
+calculateDistance() : float
+calculateEstimatedTime() : Time
+getDirections() : string[]

<<entity>> Proximity
-location : Location
+Proximity()
+calculateDistance(otherLocation : Location) : double
+getCurrentLocation() : Location
+findMarkersWithRadius(map : Map, radius : double) : Location[]
+setLocation(location : Location) : void

Sequence diagrams

Use Case 1: Search Location
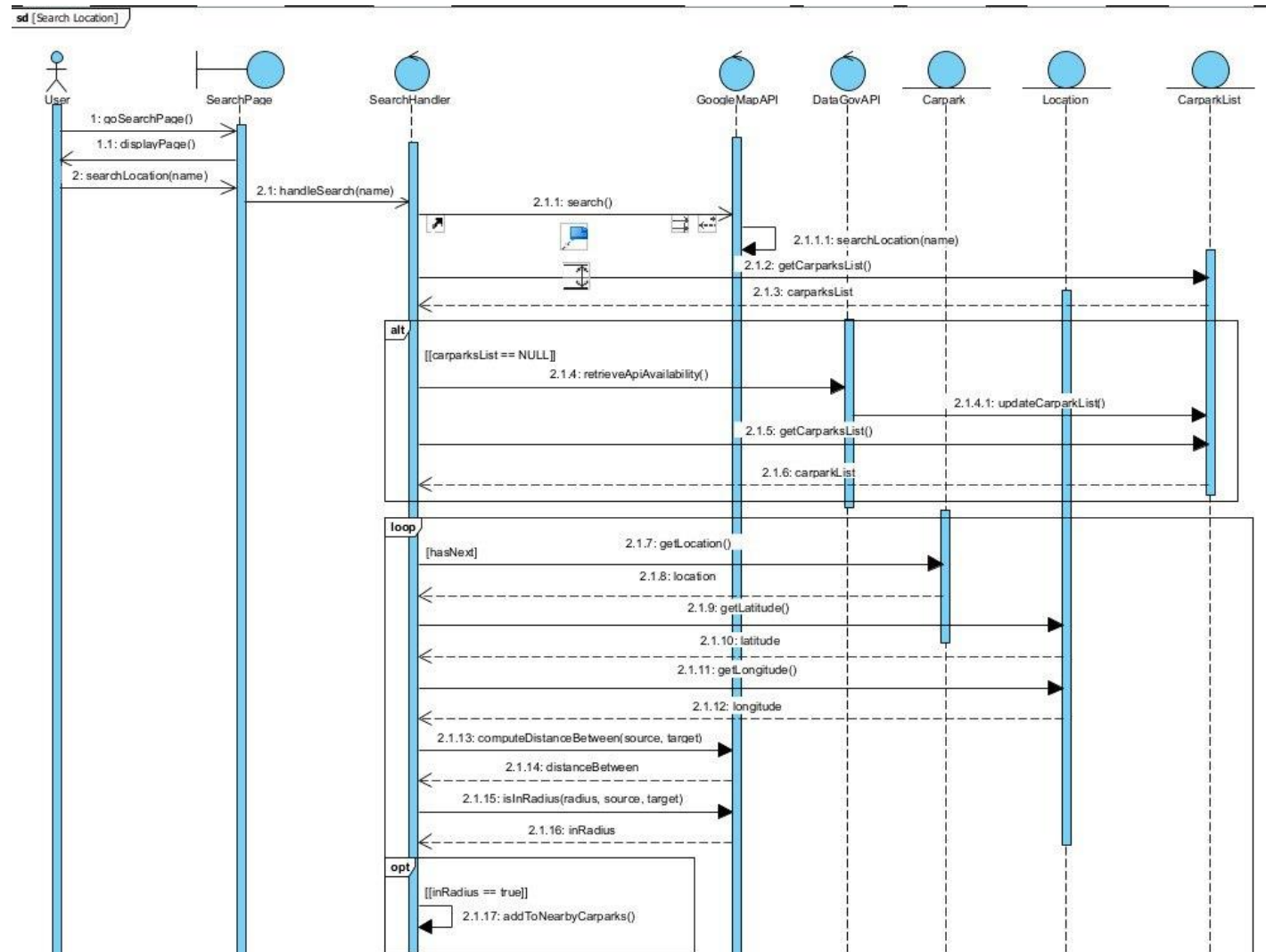
Use Case 2: Favourite Carpark

sd [Favourite Carpark]

User   DetailsPage   FavouritesHandler   UserInfo

1: User selects a carpark

1.1: getCarparkRates()

1.2: getCarparkID()

1.3: getCarparkAddress()

1.4: displayPage()

2: User clicks on the hollow star icon

2.1: addFavouriteCarpark()
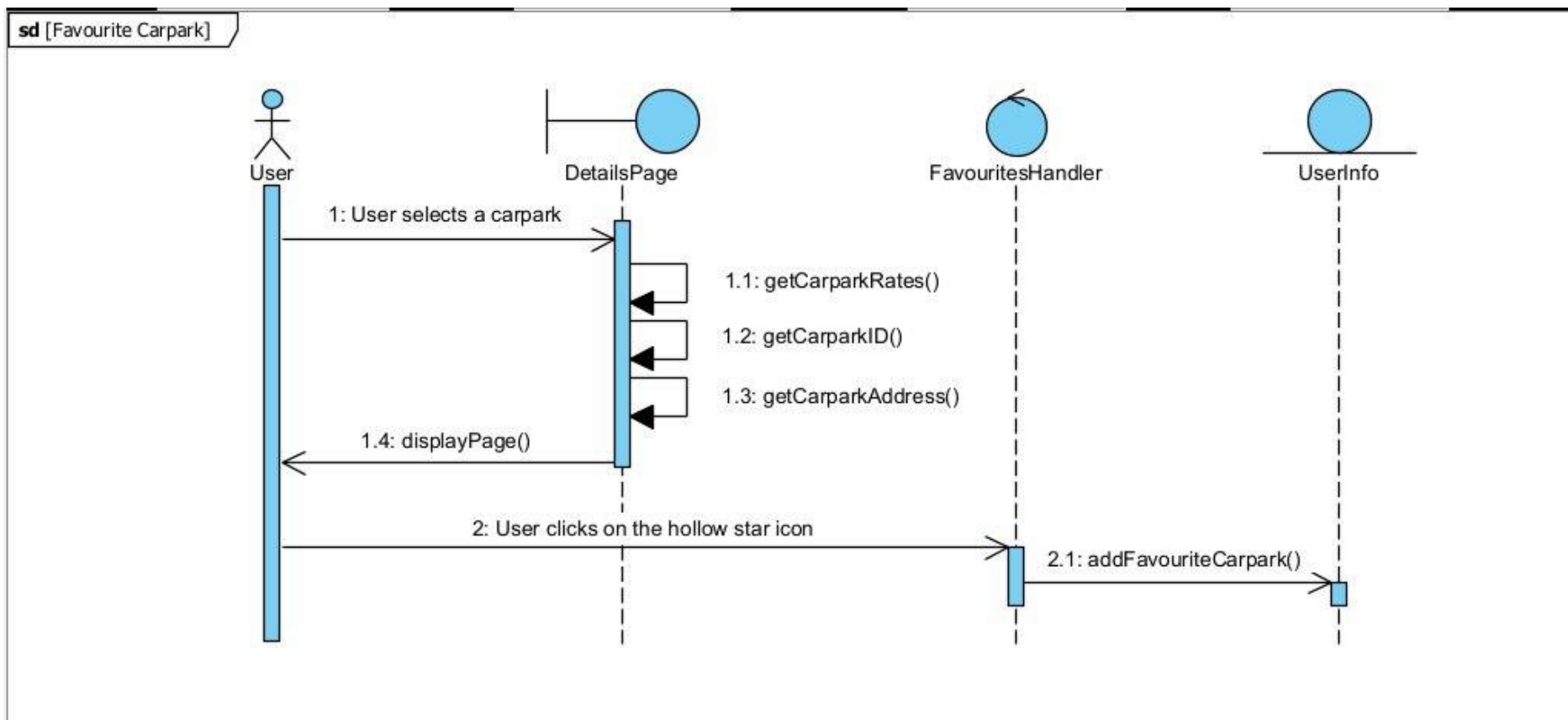
Use Case 3: View Favourite Carpark

sd [View Favourite Carpark]

User — FP : FavouritePage — FH : FavouritesHandler — U : UserInfo

1: clicks on View Favourite Page

1.1: getFavourites() : List<Carpark>

1.1.1: getFavourites() : List<Carpark>

1.1.2: favourites

1.2: favourites

1.3: display User's favourite carparks
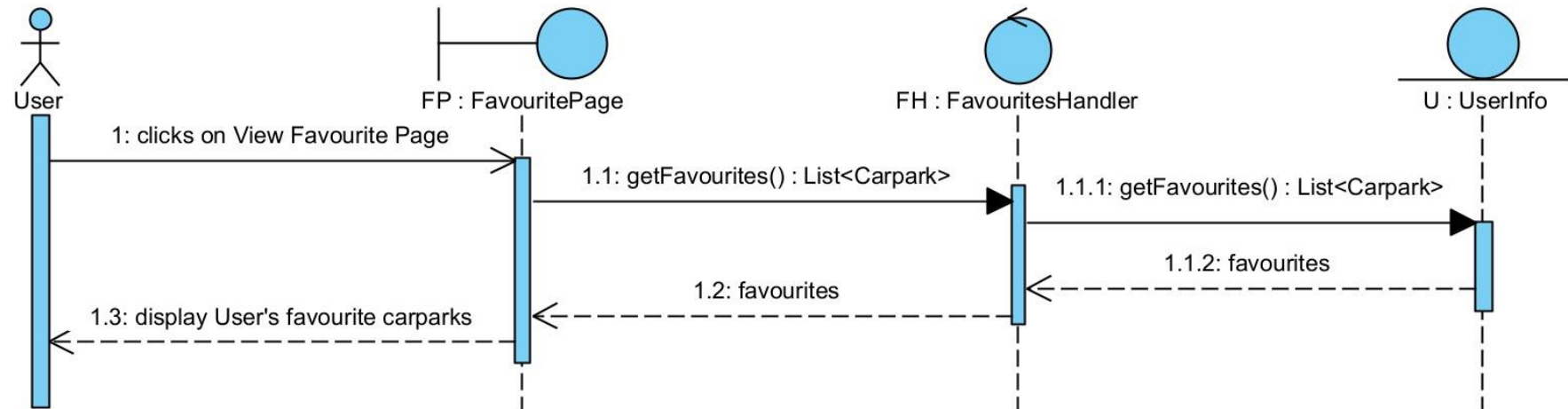
Use Case 4: Remove Favourite Carpark

sd [Remove Favourite Carparks]
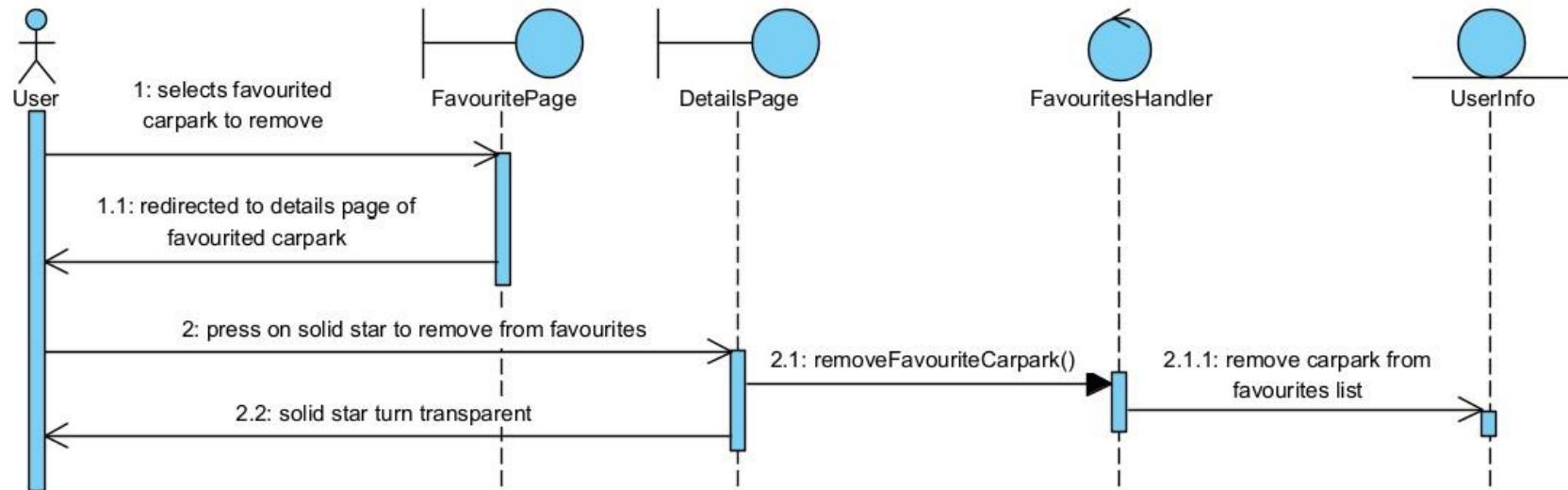
Use Case 5: Retrieve Carpark Location

sd [Retrieve Carpark Locations]

Google Maps API

1: query API for locations of carparks using coordinates

GoogleMapsAPI

2: returns carparks locations

3: mark locations

4: display markers on map

Map
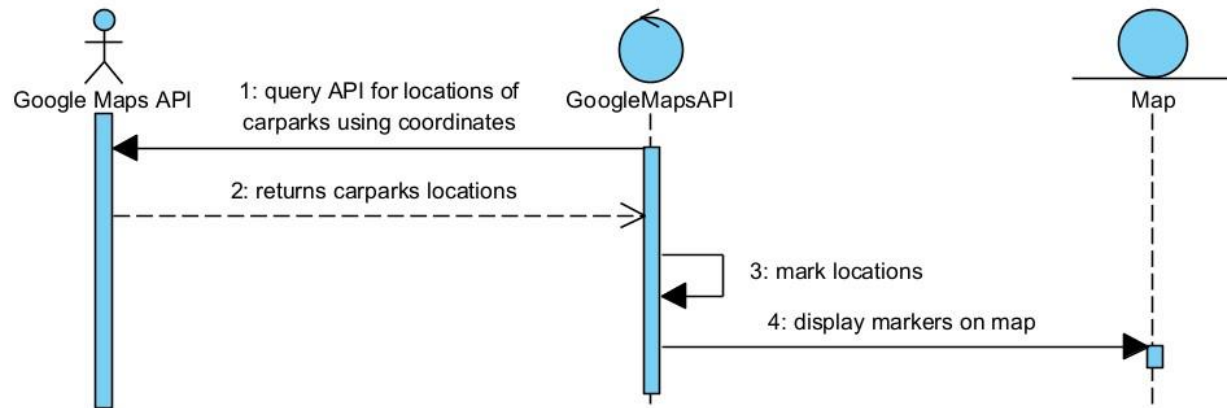
Use Case 6: Retrieve Carpark Information
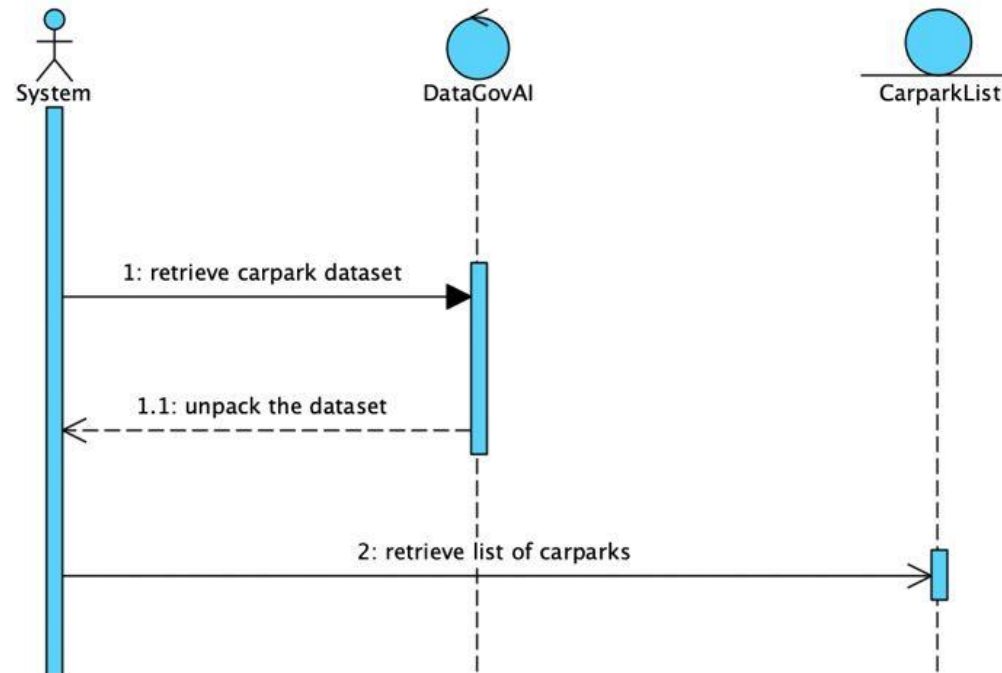
Use Case 7: Unselect Carpark to Visit

sd [Unselect Carpark to Visit]

User

1: press unselect
carpark button

ResultsPage

1.1: unselectCarpark()

SelectionHandler

1.2: green tick indicating
selected is removed
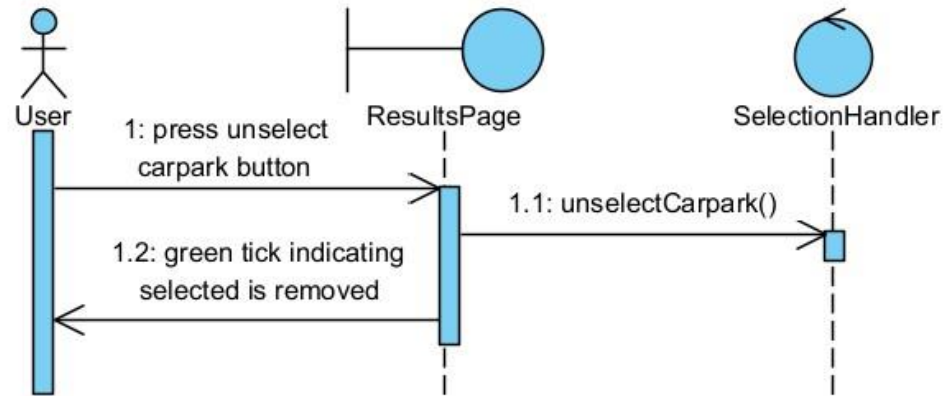
Use Case 8: Check Radius
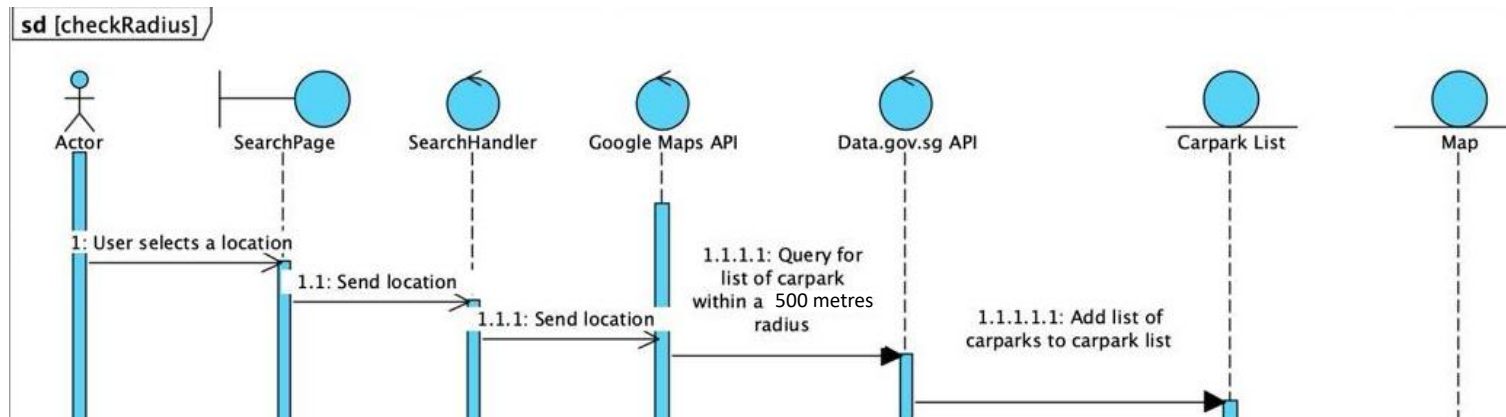
Use Case 9: Recommend Nearby Carparks

sd [Recommend Nearby Carparks]

User	SearchPage	SearchHandler	GoogleMapAPI

1: searchLocation(name)

1.1: handleSearch(name)

2: search()

2.1: searchLocation(name)

loop

3: computeDistanceBetween(source, target)

3.1: distanceBetween

4: isInRadius(radius, source, target)

4.1: inRadius

5: addToNearbyCarparks()

6: sortNearbyCarparks()

7: nearbyCarparks

7.1: setSearchResult(nearbyCarparks)

7.2: displaySearchResult()
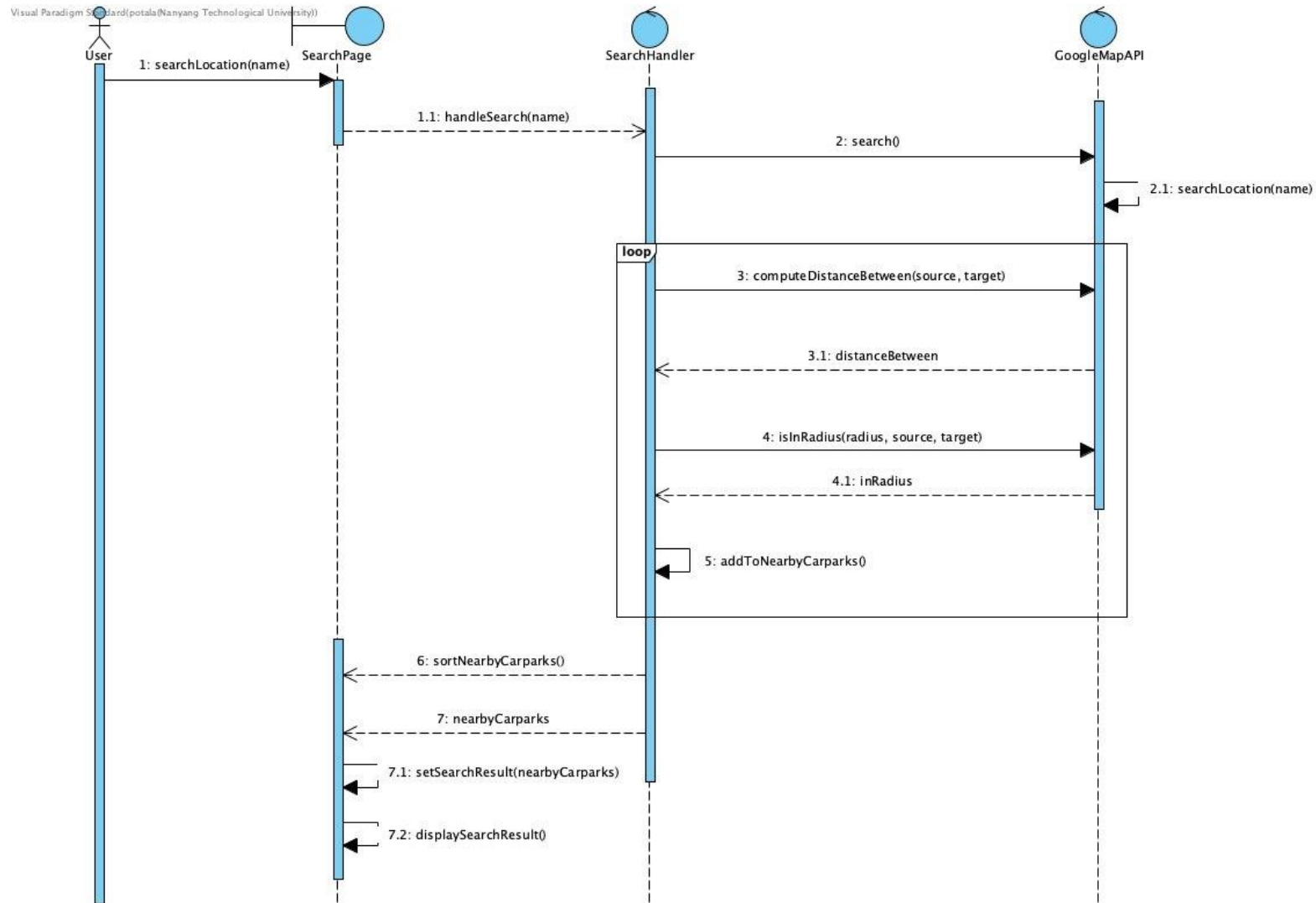
Use Case 10: Select Carpark To Visit
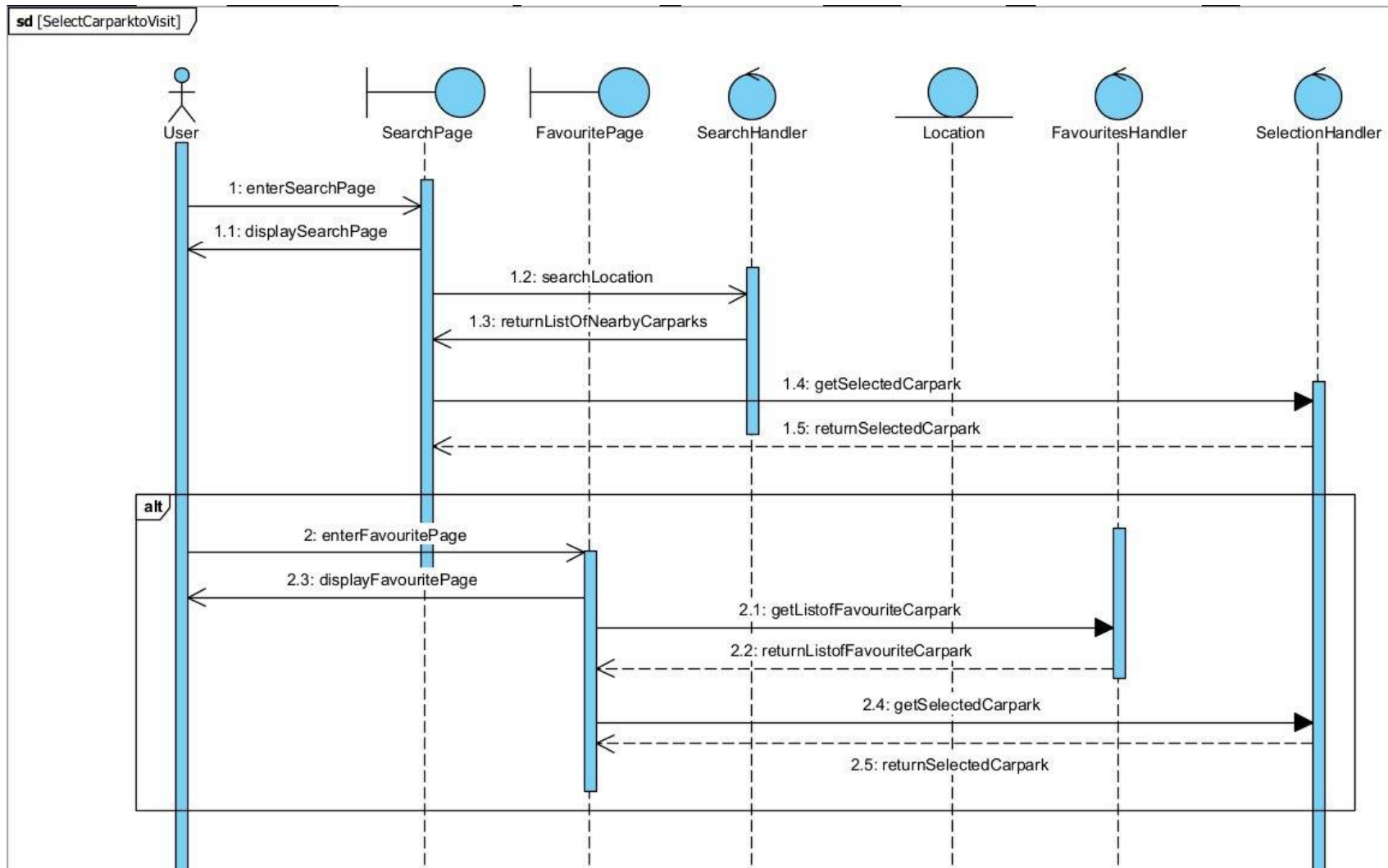
Use Case 11: Low Availability Warning

sd [Low availability warning]

User   SearchPage

1: select carpark

alt

[if CarparkAvailability < 5]

1.1: Print Low availability message
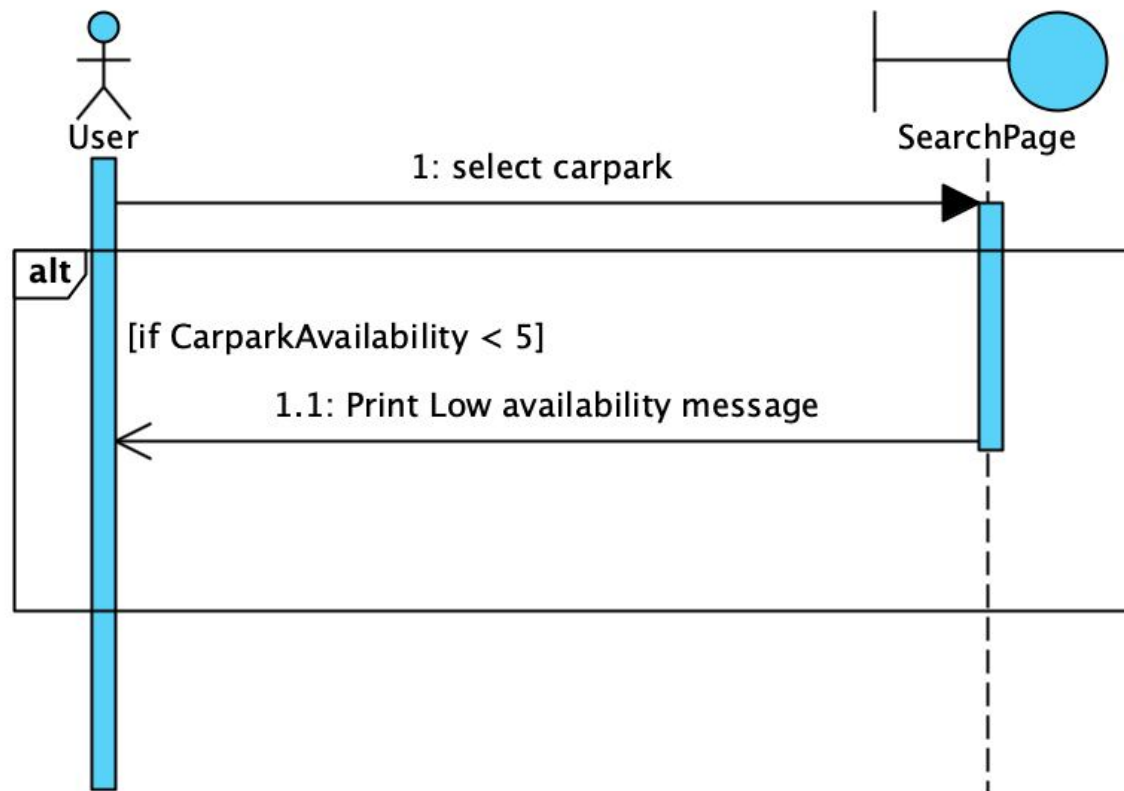
Use Case 12: Retrieve Carpark Availability

sd [Retrieve Carpark Availability]

Data.gov.sg API   DataGovAPI   CarparkList

1: query API for latest
Carpark Availability data

2: returns Carpark Availability data

3: update corresponding
carpark availability
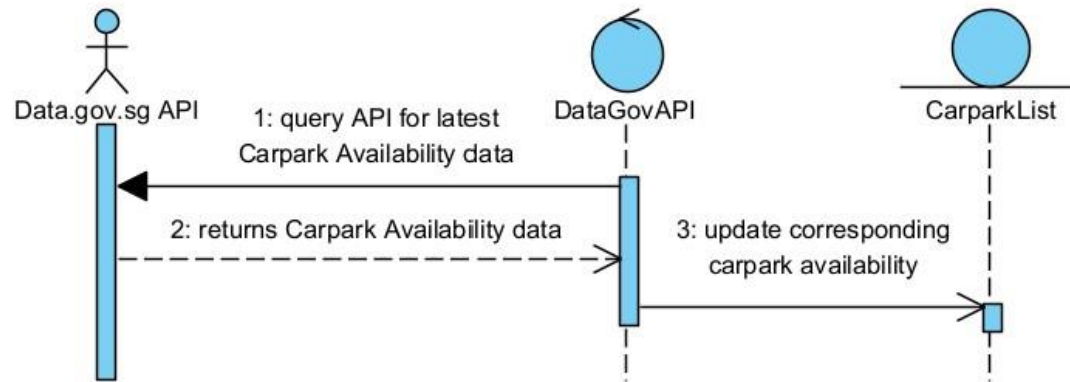
Use Case 13: Get Location of User

sd [Get Location of User]

GoogleMapAPI   GPSModule   Map

1: retrieve location of user

1.1: return user's coordinates

2: search location

3: mark location on map
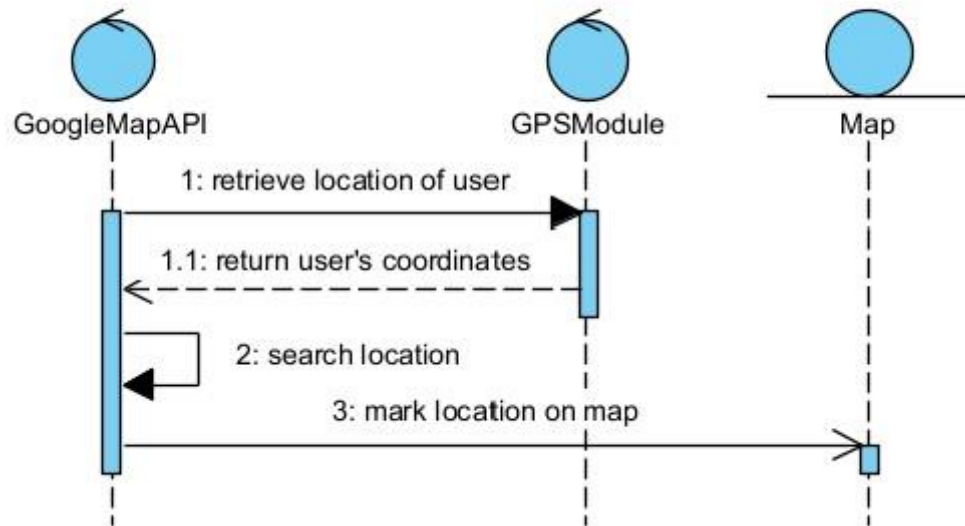
Use Case 14: Navigate Carpark
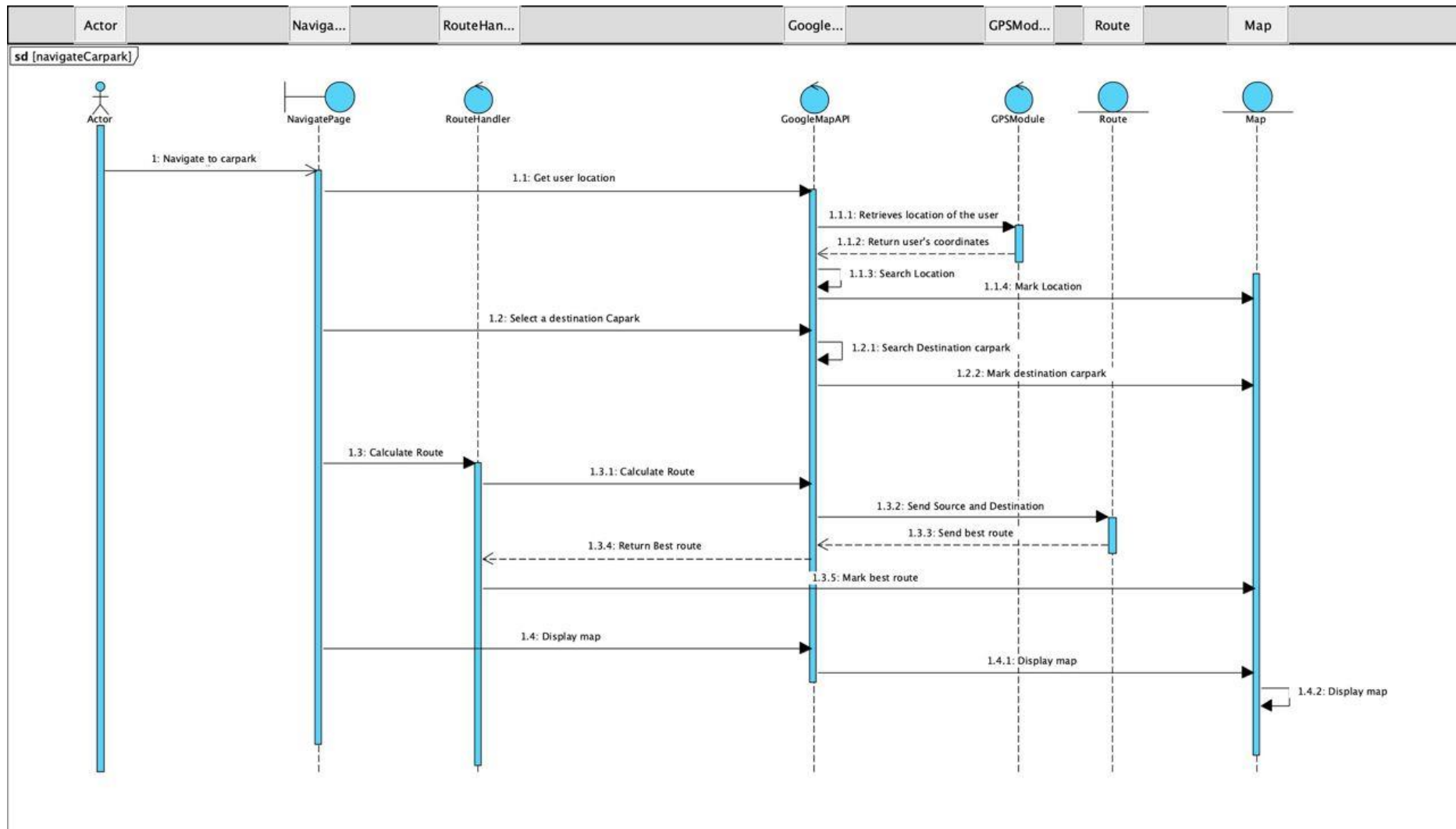
Use Case 15: Display Route

sd [displayRoute]

GoogleMapsAPI   RouteHandler   Maps

1: getRoute()

1.1: computeRoutet()

1.2: returns the best route

2: DisplayMap()

Use Case 16: Sort Results

sd [Sort Results]

User   Search Page   SortHandler   SearchHandler

1: User selects the dropdown and chooses sort type

1.1: setSortSelection()

1.1.1: applySort()

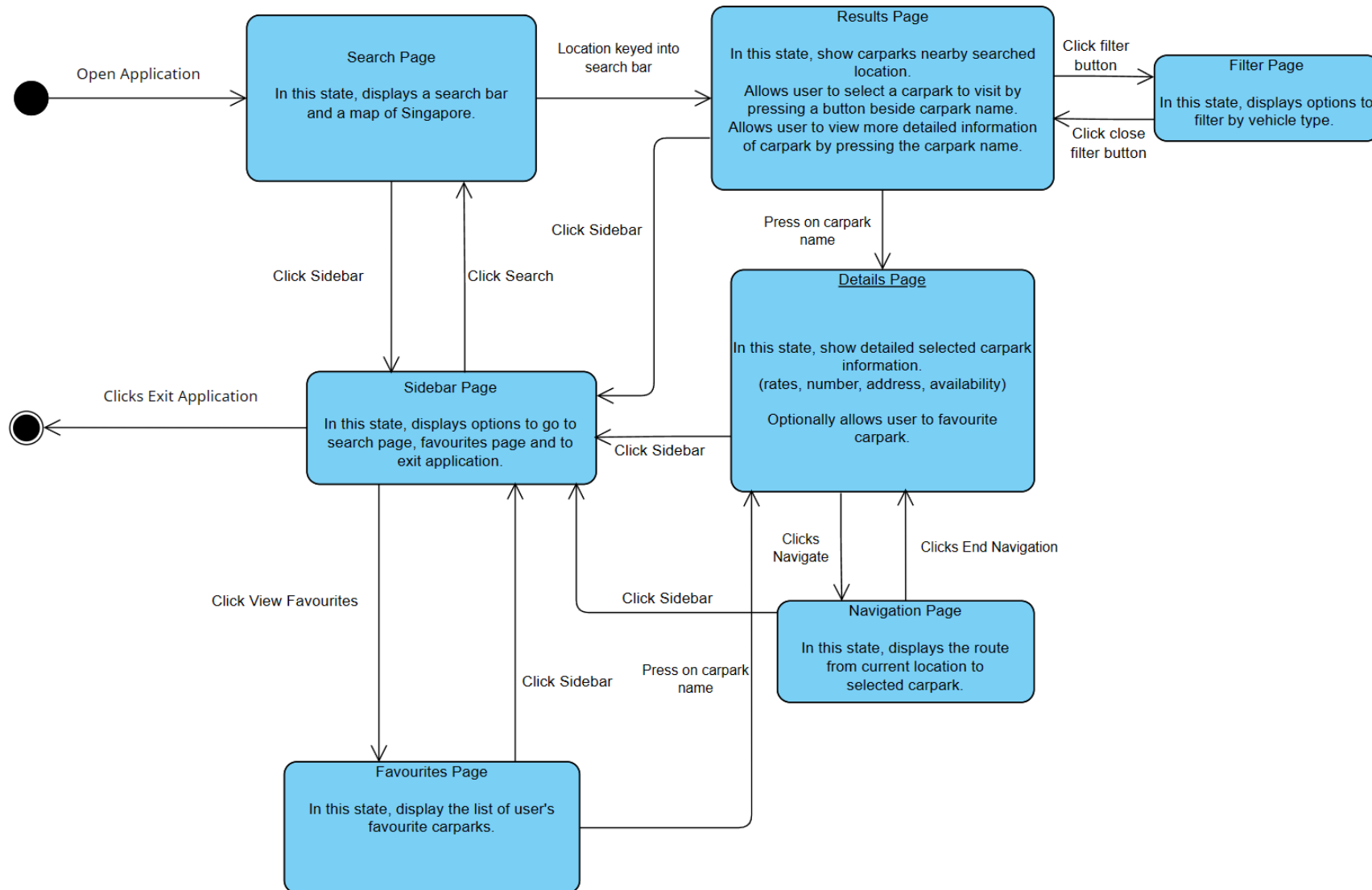1.1.2: Sort the results accordingly to filter
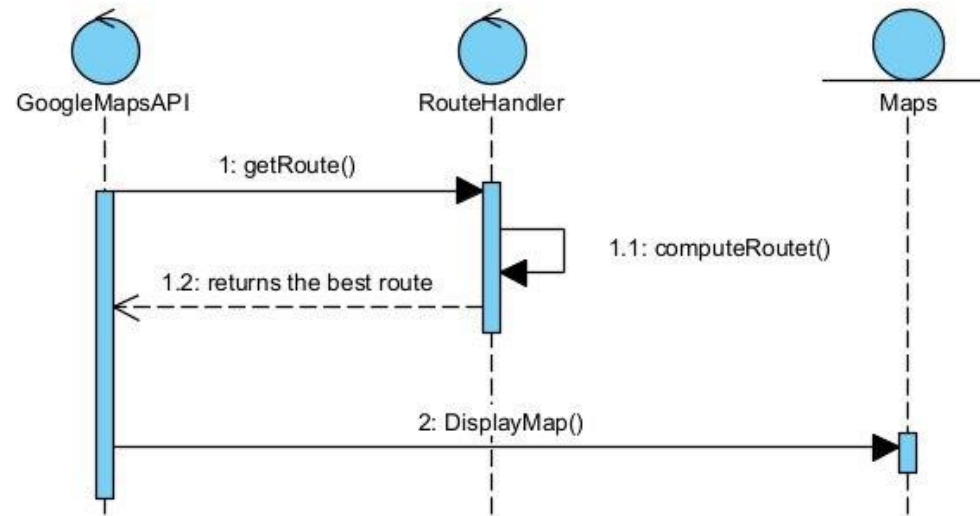
1.1.2.1: sortNearbyCarparks()
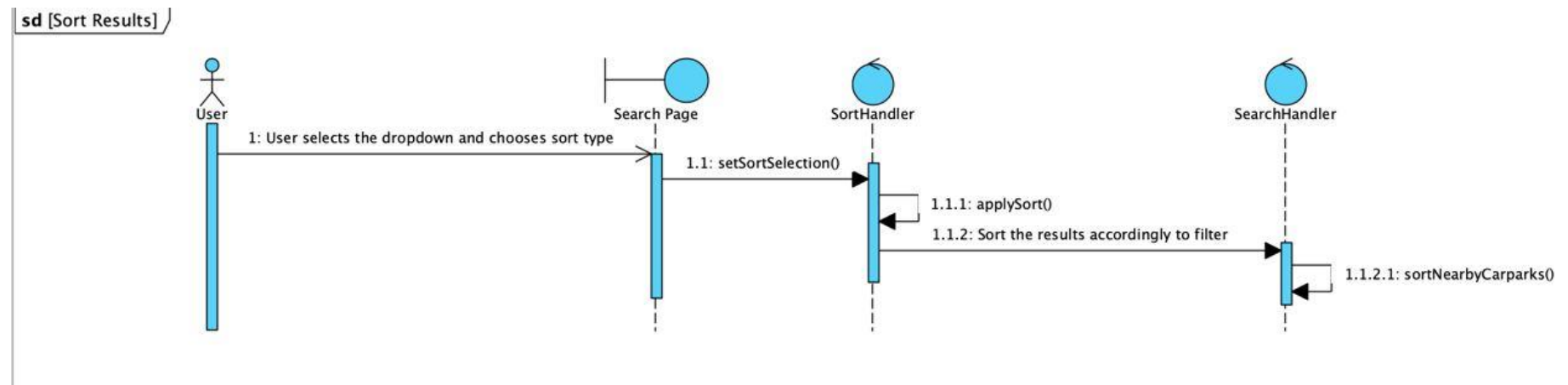
Use Case 17: Get Carpark Rates

sd [Get Carpark Rates]
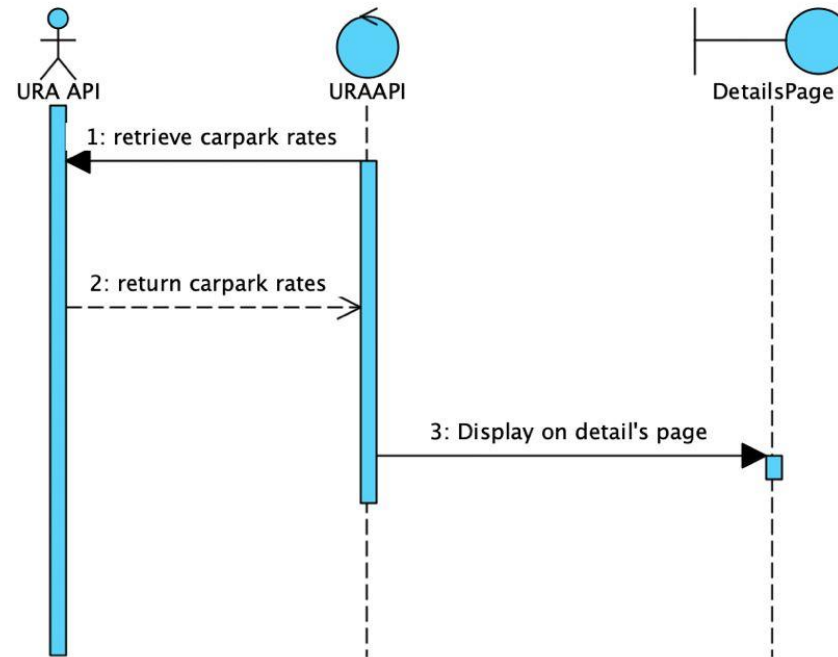
Dialog Map (State Machine Diagram)

System Architecture