SC2006 Project:

ParkNow

Presented by: **QuadCode**

Devlin, Wei Hsuen, Daryl, Zheng Wei
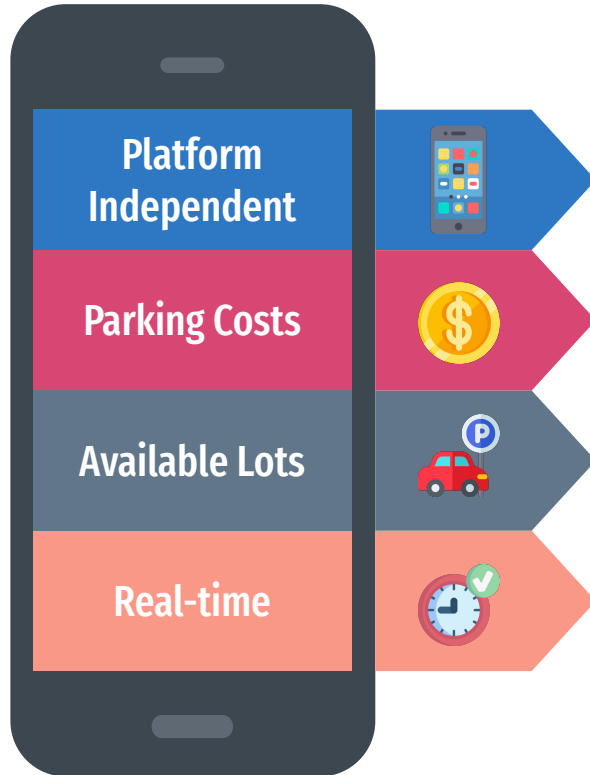
Live Demo
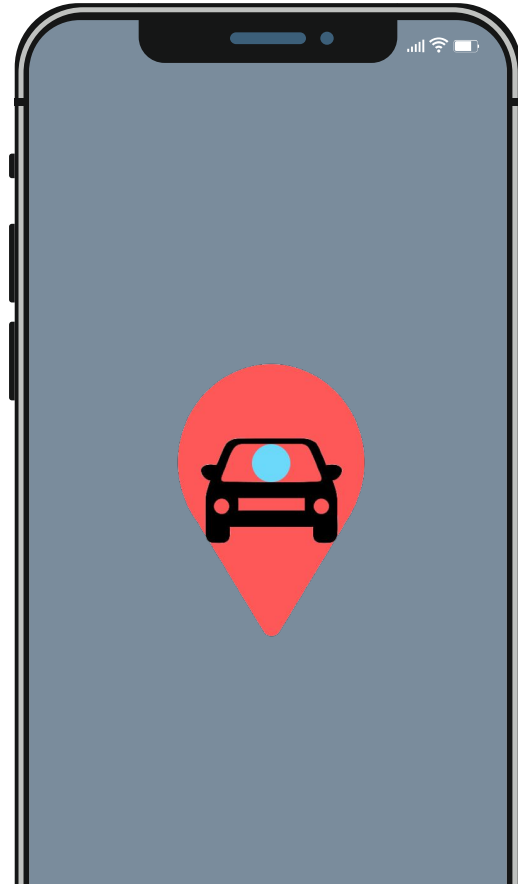
Wagile Approach?
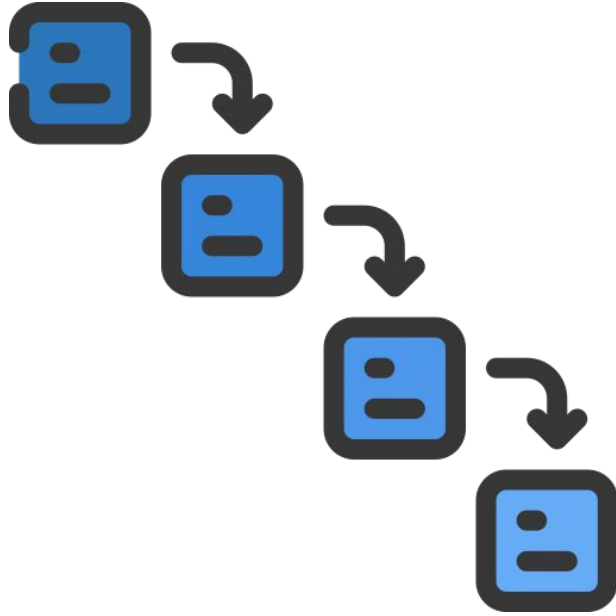
Wagile Approach?
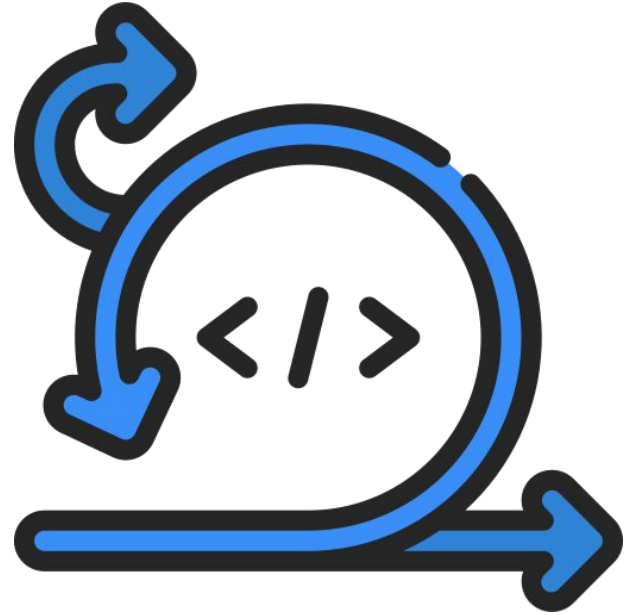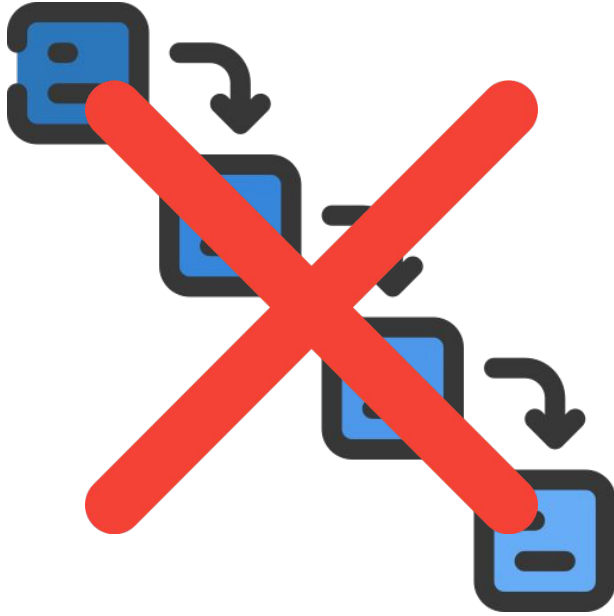
Agile Approach

Emphasis on working software and collaboration

Lightweight use cases focusing on essential scenarios that are important to understand the user's perspective and guide development decisions

Agile Methods

Incremental Delivery

Incremental Development based on user's feedback and specifications

Embrace Change

Loose Coupling and high cohesion for future extensibility

Maintain Simplicity

Clearly defined use cases, reduce complexity of system design
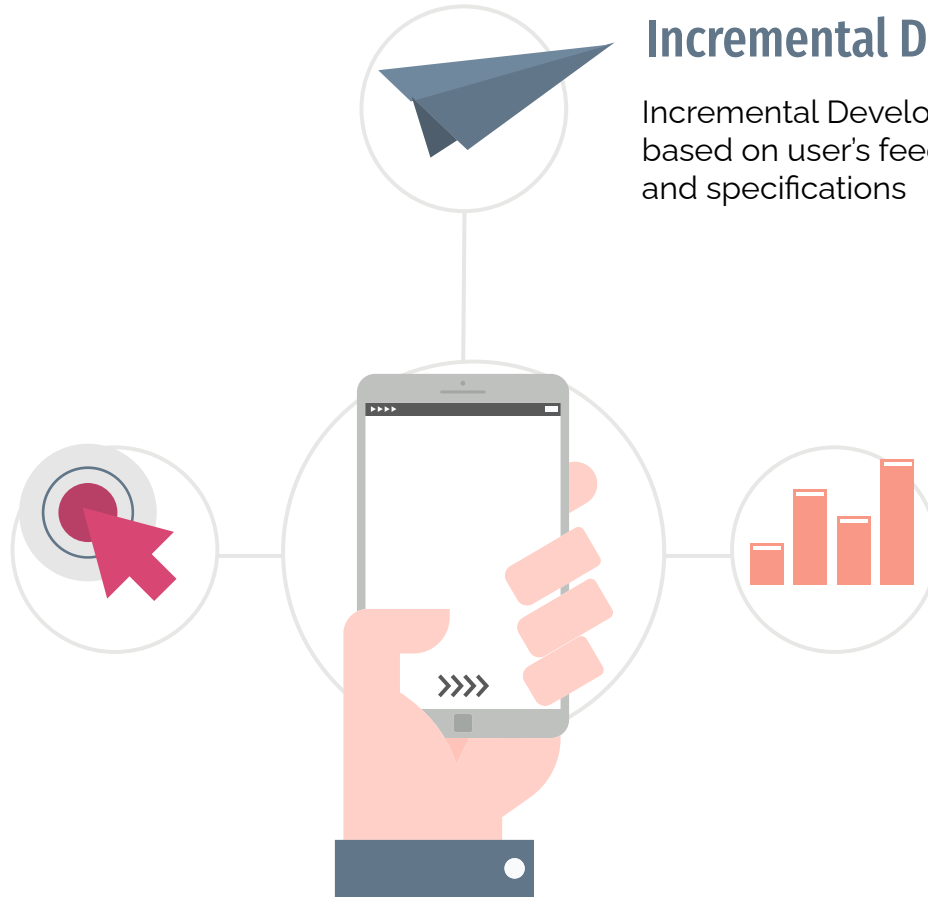
Agile Scrum Methodology
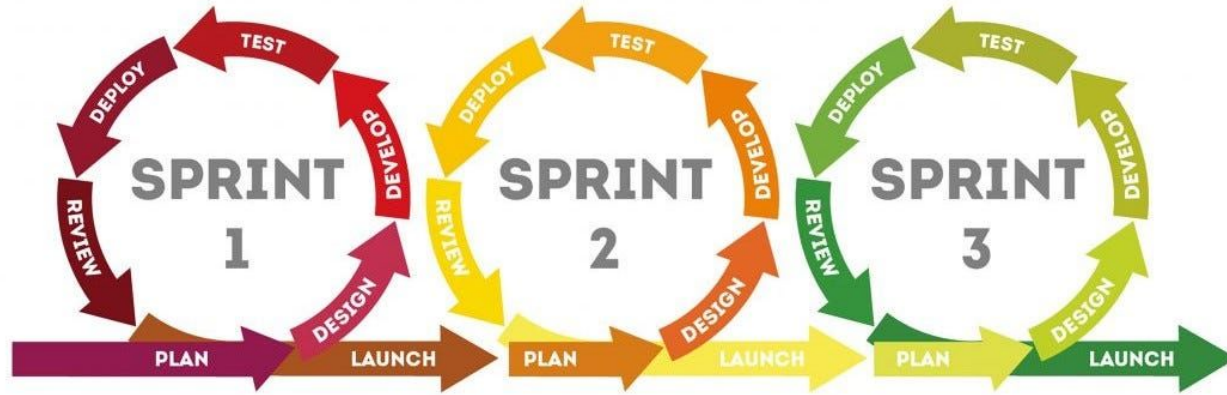
Map, Search Bar, Location Services

Markers, Database

Parking Costs Calculator, Car Park Availability Information

System Design

Modularity

Distinct, manageable components with specific responsibilities.

01

Scalability

Scale internationally with other countries' car park APIs

02

Adaptability

System not affected by new functionalities in the future

03

Documentation

Describes the system's overall design, readable to experts and non-experts.

04

Class Diagram

<<Boundary>> RequestLocationServices

- firstLogin:bool

+ promptLocationPermission(MapController):bool
+ updatePermission(bool promptUpdate,MapController): void
+ promptUpdate(): bool

<<Boundary>> MapInterface

- placedPressedCoords: int[]
- placePressed:bool

+ displayCurrentLocation(MapController):void
+ displayDestination(MapController):void
+ promptDestination(MapController): String
+ getPlacePressedCoords(): int[]
+ displayCarParkMarker(MapController): void
+ displayMap(MapController):void
+ displayRoute(MapController):void

<<Boundary>> CarParkInformationPreview

+ PromptDuration(MapController): int
+ displayParkingCost(MapController):void

<<Boundary>> CarParkDetailsTab

- navigatePressed:bool
- moreDetailsPressed:bool
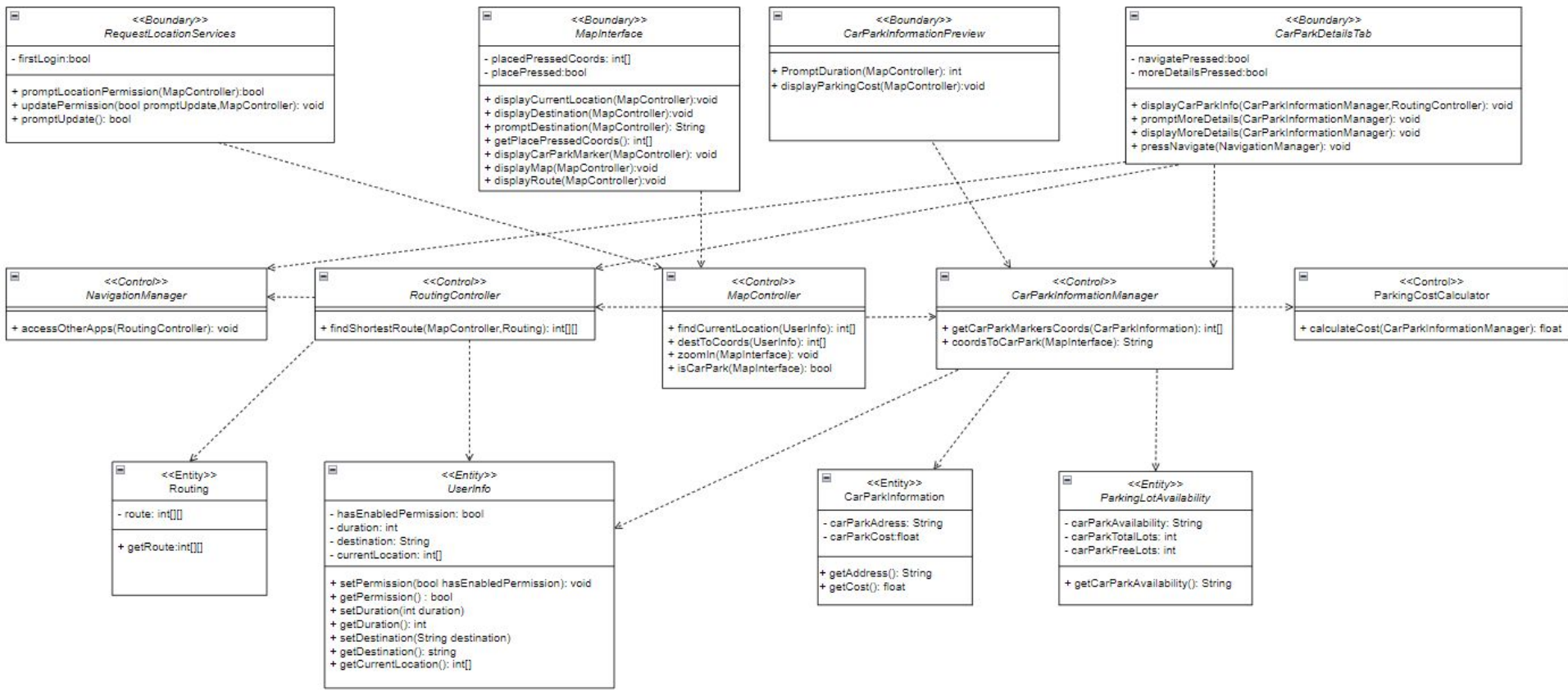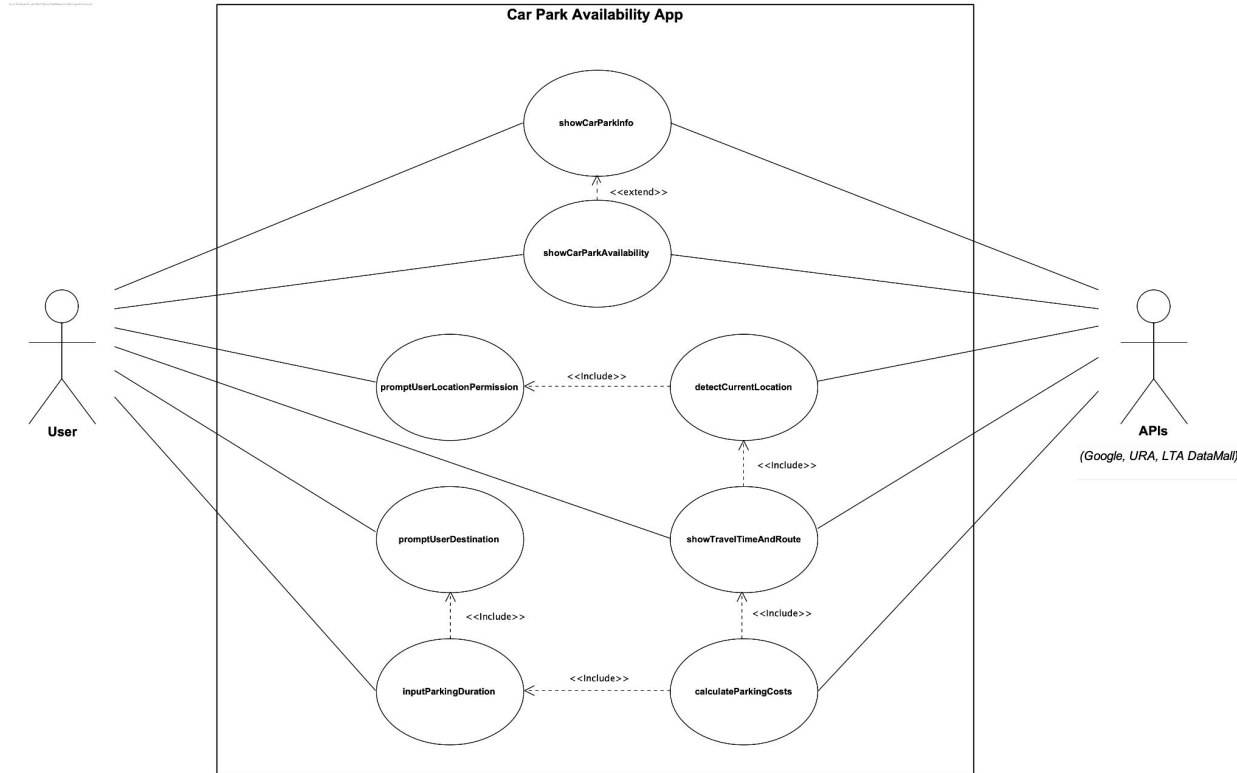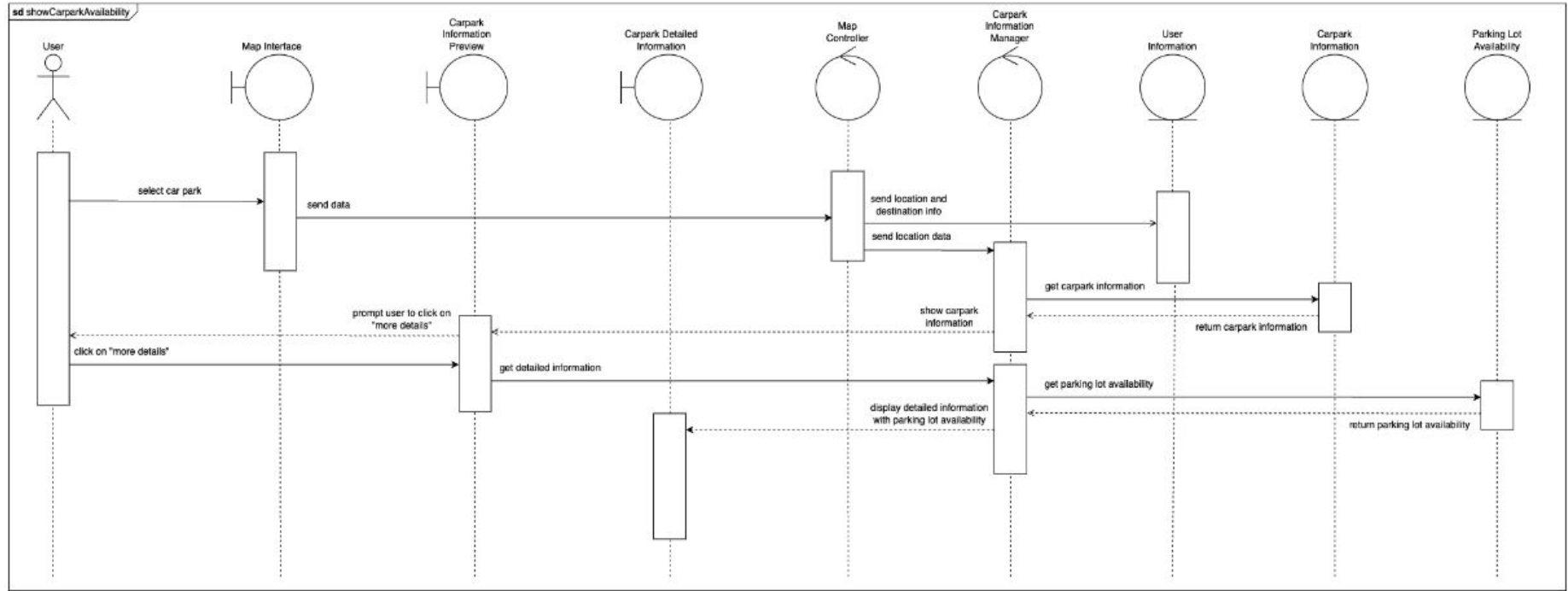
+ displayCarParkInfo(CarParkInformationManager,RoutingController): void
+ promptMoreDetails(CarParkInformationManager): void
+ displayMoreDetails(CarParkInformationManager): void
+ pressNavigate(NavigationManager): void

<<Control>> NavigationManager

+ accessOtherApps(RoutingController): void

<<Control>> RoutingController

+ findShortestRoute(MapController,Routing): int[][]

<<Control>> MapController

+ findCurrentLocation(UserInfo): int[]
+ destToCoords(UserInfo): int[]
+ zoomIn(MapInterface): void
+ isCarPark(MapInterface): bool

<<Control>> CarParkInformationManager

+ getCarParkMarkersCoords(CarParkInformation): int[]
+ coordsToCarPark(MapInterface): String

<<Control>> ParkingCostCalculator

+ calculateCost(CarParkInformationManager): float

<<Entity>> Routing

- route: int[][]

+ getRoute:int[][]

<<Entity>> UserInfo

- hasEnabledPermission: bool
- duration: int
- destination: String
- currentLocation: int[]

+ setPermission(bool hasEnabledPermission): void
+ getPermission() : bool
+ setDuration(int duration)
+ getDuration(): int
+ setDestination(String destination)
+ getDestination(): string
+ getCurrentLocation(): int[]

<<Entity>> CarParkInformation

- carParkAdress: String
- carParkCost:float

+ getAddress(): String
+ getCost(): float

<<Entity>> ParkingLotAvailability

- carParkAvailability: String
- carParkTotalLots: int
- carParkFreeLots: int

+ getCarParkAvailability(): String

Use Case Diagram

Car Park Availability App

showCarParkInfo

<<extend>>

showCarParkAvailability

promptUserLocationPermission

<<Include>>

detectCurrentLocation

<<Include>>

promptUserDestination

showTravelTimeAndRoute

<<Include>>

<<Include>>

inputParkingDuration

<<Include>>

calculateParkingCosts

User

APIs

(Google, URA, LTA DataMall)

Use Case: showCarParkAvailability

Displays Car Park

01

02

03

04

Name

Address

Availability

Rates

Sequence Diagram

sd showCarparkAvailability

User | Map Interface | Carpark Information Preview | Carpark Detailed Information | Map Controller | Carpark Information Manager | User Information | Carpark Information | Parking Lot Availability

select car park
send data
send location and destination info
send location data
get carpark information
prompt user to click on "more details"
show carpark information
return carpark information
click on "more details"
get detailed information
get parking lot availability
display detailed information with parking lot availability
return parking lot availability

Use Case: CalculateParkingCosts

Calculates a good estimate for parking costs using:

01 — Input Parking Duration

02 — Current Time

03 — Estimate travel time

04 — Pricing Structure

Sequence Diagram

System Design in Use Case

```
const DetailsButton = () => {
  return (
    <View style={styles.detailsContainer1}>
      <TouchableOpacity
        onPress={toggleMoreDetailsOverlay}
        style={styles.detailsContainer2}
      >
        <Feather
          name="more-vertical"
          size={30}
          color="black"
          style={styles.detailsIcon}
        />
      </TouchableOpacity>
    </View>
  );
};
```

```
<DetailsButton />
```

```
<NavigateButton />
```

```
const NavigateButton = () => {
  return (
    <View style={styles.navigateContainer1}>
      <TouchableOpacity
        onPress={openNavigateOverlay}
        style={styles.navigateContainer2}
      >
        <Text style={styles.navigateLabel}>Navigate</Text>
      </TouchableOpacity>
    </View>
  );
};
```
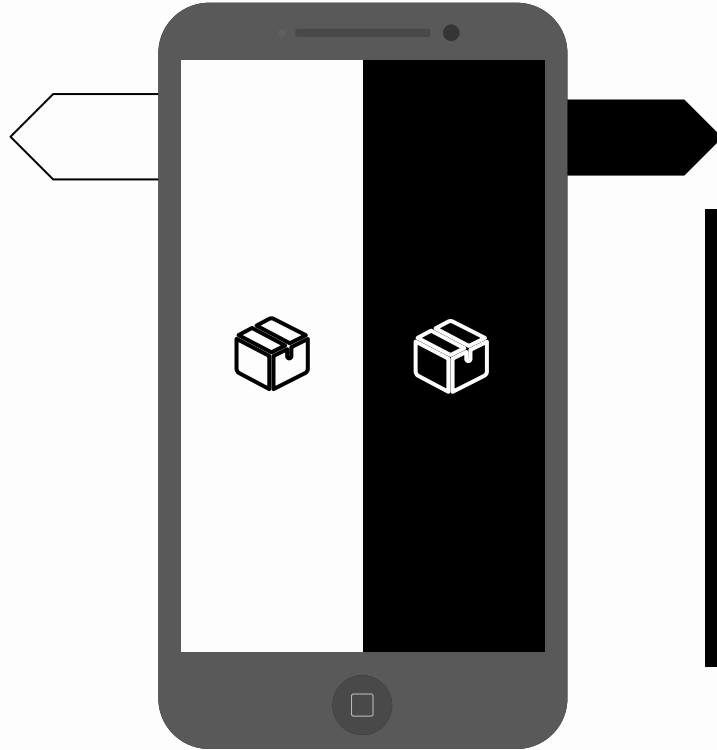
Modularity

Extensibility

Single Responsibility Components

Scale Internationally

Testing

White Box Testing

1. API retrieval fail
   - error message
2. User did not enable location access
   - displayCarParkMarker() unaffected

Black Box Testing

1. Car Park Information Manager Control Class
   - Valid Equivalence Classes for 0 mins to 23h45mins

   `LOG` `Parking Time Input of 0h 15min is valid`

   - Invalid Equivalence Classes below 0 mins and above 23h45mins

   `LOG` `Parking Time Input of 25h 0min is invalid`
   `LOG` `Parking Time Input of 0h -15min is invalid`

Innovative Aspects

Car Park
Availability
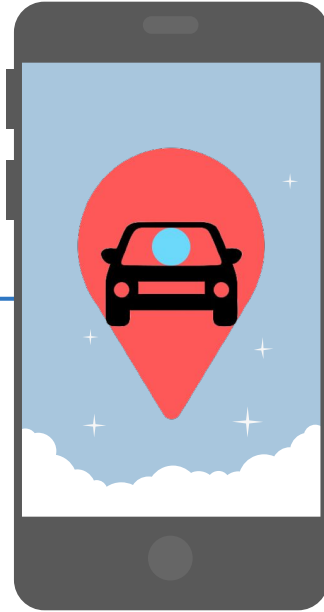
Estimated
Parking Cost

Redirect to
Navigation Apps

Thank You!