CZ2006 Software Engineering

Get Going

Prepared by SoftwareFun

Isabel Thomander	N1801169D
Surabhi Malani	U1721308A
Grace Ong Yong Han	U1721575H
Shubhankar Agrawal	N1801537B
Eugene Teo Yu-Jie	U1721688J

Lab Group: SS3

Table of Contents

1. 1.1 Introduction

1.1 Purpose

The software requirements described in this document are related to the android application Get Going, release number 1.0.0. The SRS will cover the entire system, including internal storage.

1.2 Document Conventions

We tried to fulfill each of our functional requirements for a fuller and thorough user experience.

1.3 Intended Audience and Reading Suggestions

The main target users for our application are students, office workers and tourists— basically anyone who is a frequent or newbie traveller.

1.4 Product Scope

The objective of Get Going is to ease the user's day to day travels. This application will do so by having the user submit their travel details, giving them the possibility to follow their journey in real time on a map and sending them an alert when they are approaching the destination. Our main target groups are people who have difficulties navigating where they are and who require assistance in their travels - senior citizens and people who are travelling to places they've never been before. Get Going will help them get where they want to go, and let them relax while doing so.

1.5 References

All references will be attached to this SRS document as appendices:

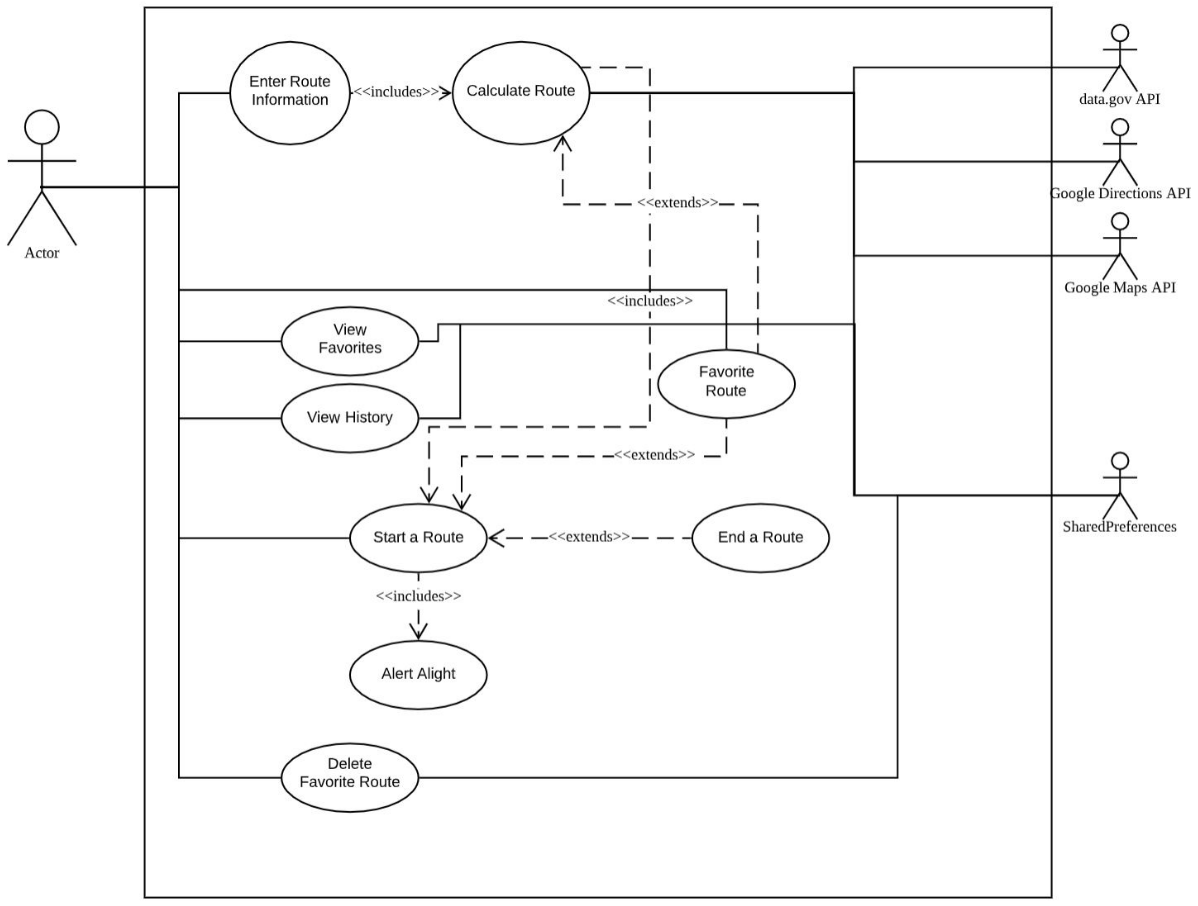
Appendix A: Data Dictionary
Appendix B: Sequence Diagrams
Appendix C: Use Case Models

2. Overall Description

2.1 Product Perspective

Get Going is a new product which resulted from a competition to elicit applications that would use publicly available government data to change the ways people live.

2.2 Product Functions



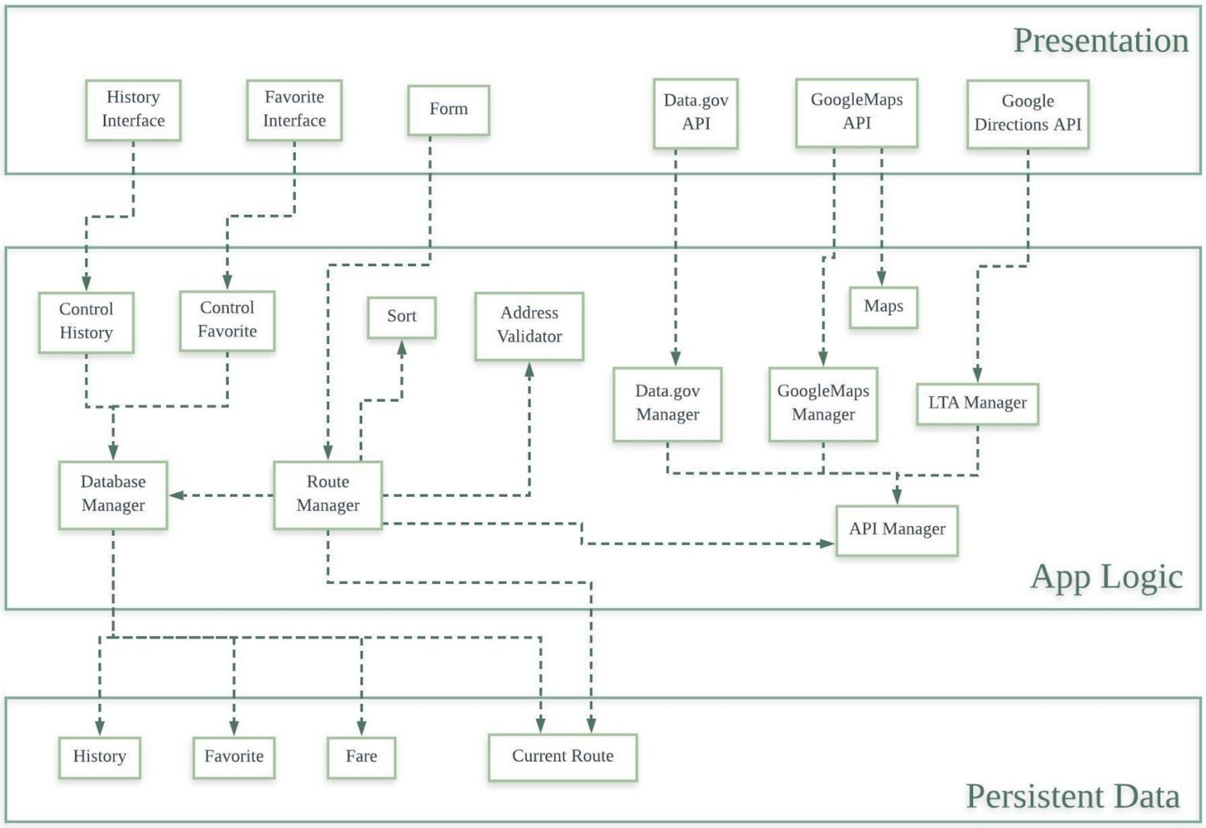
The user must be able to:
- Enter journey details
- Start a route
- View history of routes
- Manage favorite routes (View / Add / Delete)

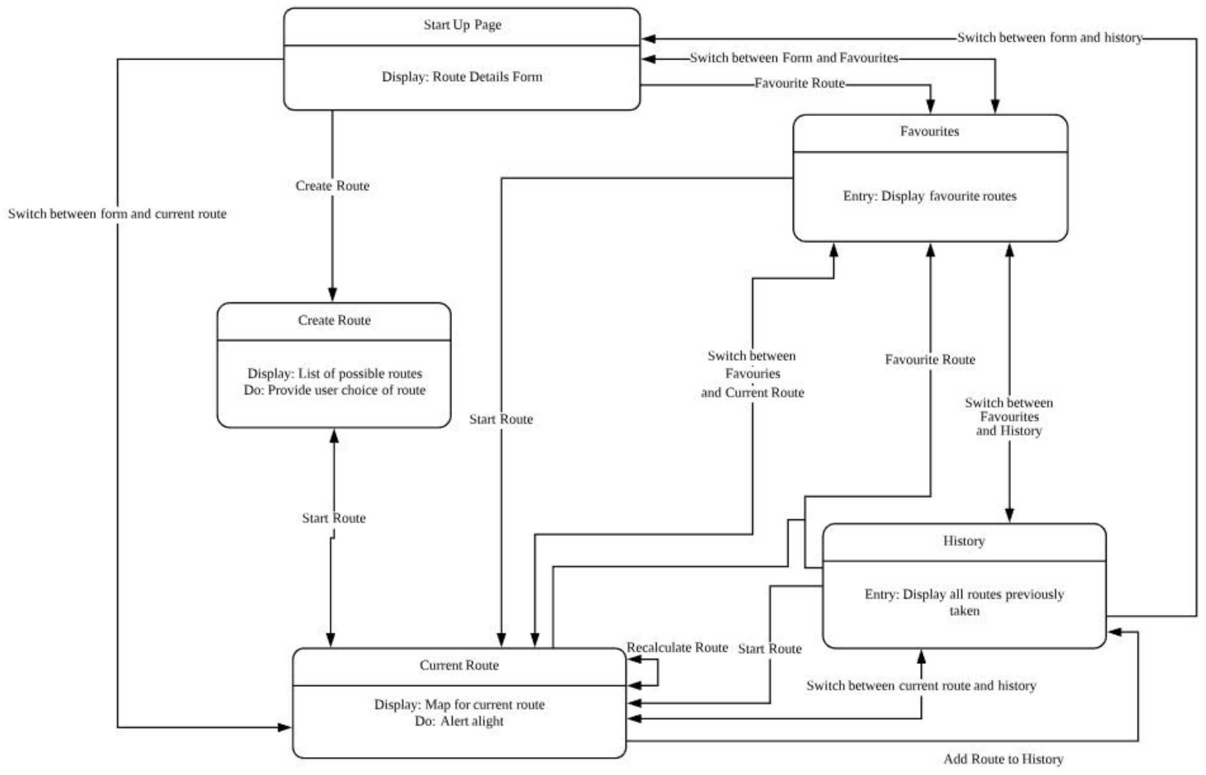
The product must be able to:
- Find routes and fares
- Calculate fastest/cheapest route
- Alert the user when he or she is approaching the destination

- Recalculate route if the user misses the destination

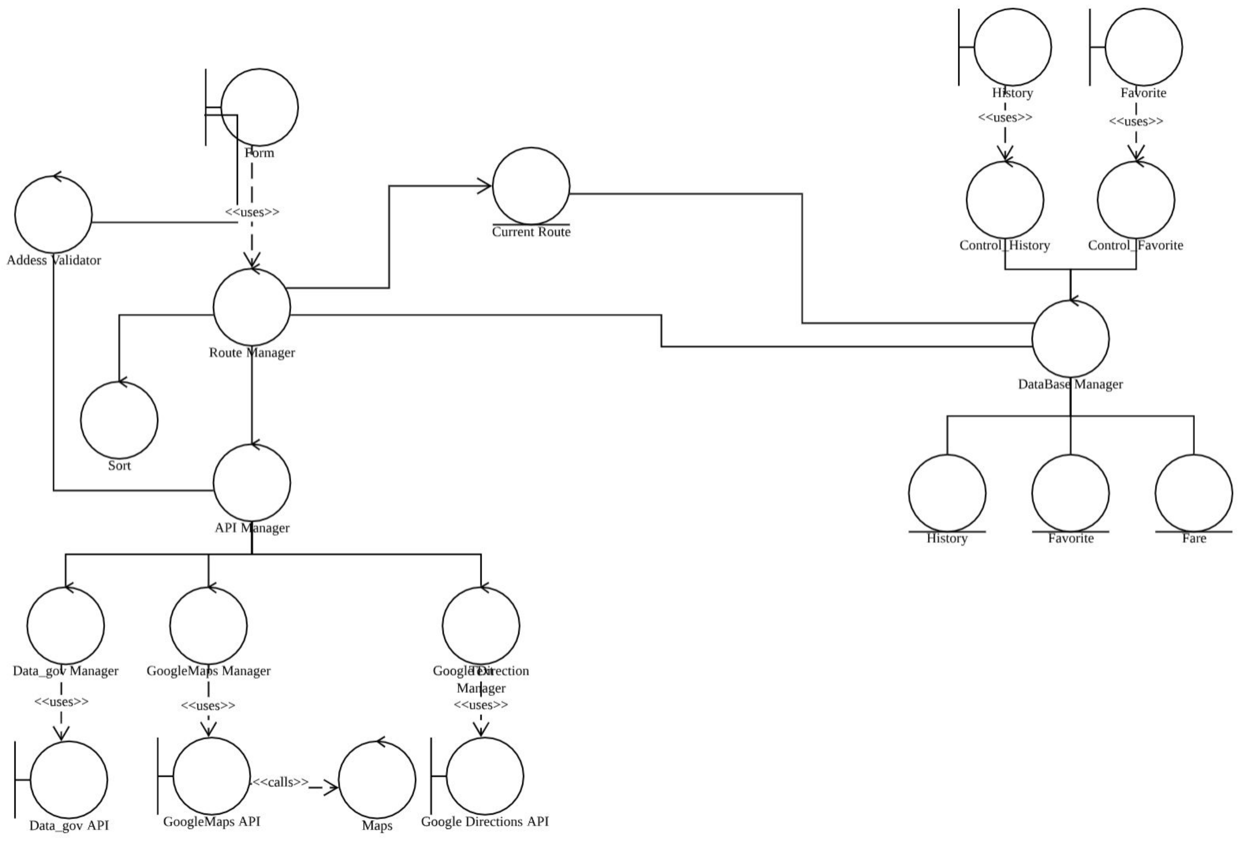
2.3 User Classes and Characteristics

Owing to the simplistic nature of our application, we believe there would only be two types of users that access the app. To begin with, we have the general user who uses our app and represents the entire target audience of our application. These are the citizens of Singapore who use public transport for commute. They have the ability to search for routes, pick a route, add to history, favourites and follow a route during the commute.
The admin user which is the development team cannot access any of the user's data. The admin team can only change functionalities of the application to improve performance and provide additional features during updates of the app.

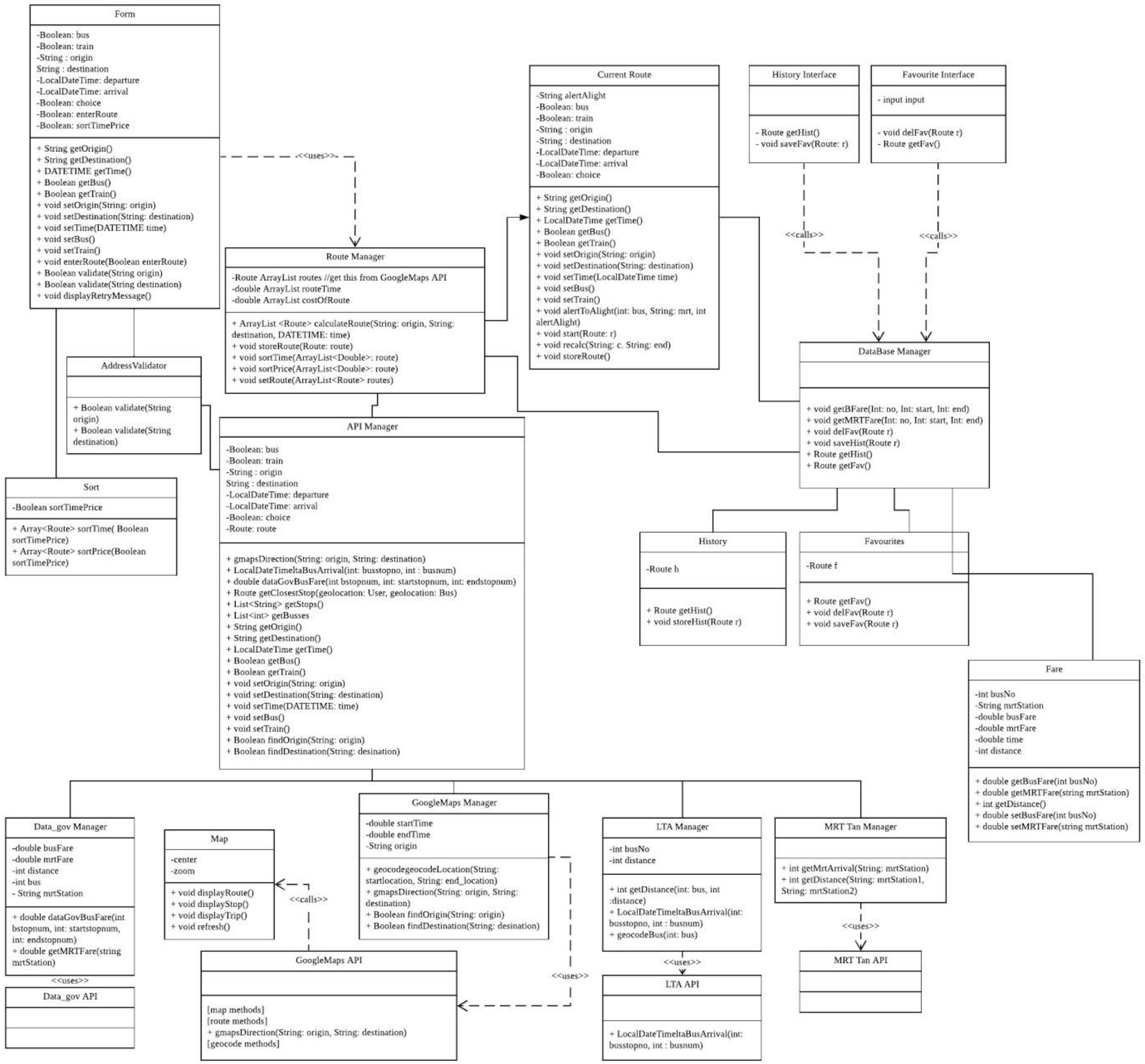



Start Up Page
Display: Route Details Form
Switch between form and history
Switch between Form and Favourites
Favourite Route
Favourites
Entry: Display favourite routes
Create Route
Switch between form and current route
Create Route
Display: List of possible routes
Do: Provide user choice of route
Start Route
Switch between Favourites and Current Route
Favourite Route
Switch between Favourites and History
Start Route
History
Entry: Display all routes previously taken
Recalculate Route
Start Route
Switch between current route and history
Current Route
Display: Map for current route
Do: Alert alight
Add Route to History

Dialog Map of the Application



Class Diagram


Form
-Boolean: bus
-Boolean: train
-String : origin
String : destination
-LocalDateTime: departure
-LocalDateTime: arrival
-Boolean: choice
-Boolean: enterRoute
-Boolean: sortTimePrice
+ String getOrigin()
+ String getDestination()
+ DATETIME getTime()
+ Boolean getBus()
+ Boolean getTrain()
+ void setOrigin(String: origin)
+ void setDestination(String: destination)
+ void setTime(DATETIME time)
+ void setBus()
+ void setTrain()
+ void enterRoute(Boolean enterRoute)
+ Boolean validate(String origin)
+ Boolean validate(String destination)
+ void displayRetryMessage()
<<uses>>
Route Manager
-Route ArrayList routes //get this from GoogleMaps API
-double ArrayList routeTime
-double ArrayList costOfRoute
+ ArrayList <Route> calculateRoute(String: origin, String: destination, DATETIME: time)
+ void storeRoute(Route: route)
+ void sortTime(ArrayList<Double>: route)
+ void sortPrice(ArrayList<Double>: route)
+ void setRoute(ArrayList<Route> routes)
Current Route
-String alertAlight
-Boolean: bus
-Boolean: train
-String : origin
-String : destination
-LocalDateTime: departure
-LocalDateTime: arrival
-Boolean: choice
+ String getOrigin()
+ String getDestination()
+ LocalDateTime getTime()
+ Boolean getBus()
+ Boolean getTrain()
+ void setOrigin(String: origin)
+ void setDestination(String: destination)
+ void setTime(LocalDateTime time)
+ void setBus()
+ void setTrain()
+ void alertToAlight(int: bus, String: mrt, int alertAlight)
+ void start(Route: r)
+ void recalc(String: c. String: end)
+ void storeRoute()
History Interface
- Route getHist()
- void saveFav(Route: r)
Favourite Interface
- input input
- void delFav(Route r)
- Route getFav()
<<calls>>
<<calls>>
DataBase Manager
+ void getBFare(Int: no, Int: start, Int: end)
+ void getMRTFare(Int: no, Int: start, Int: end)
+ void delFav(Route r)
+ void saveHist(Route r)
+ Route getHist()
+ Route getFav()
AddressValidator
+ Boolean validate(String origin)
+ Boolean validate(String destination)
Sort
-Boolean sortTimePrice
+ Array<Route> sortTime(Boolean sortTimePrice)
+ Array<Route> sortPrice(Boolean sortTimePrice)
API Manager
-Boolean: bus
-Boolean: train
-String : origin
String : destination
-LocalDateTime: departure
-LocalDateTime: arrival
-Boolean: choice
-Route: route
+ gmapsDirection(String: origin, String: destination)
+ LocalDateTimeltaBusArrival(int: busstopno, int : busnum)
+ double dataGovBusFare(int bstopnum, int: startstopnum, int: endstopnum)
+ Route getClosestStop(geolocation: User, geolocation: Bus)
+ List<String> getStops()
+ List<int> getBusses
+ String getOrigin()
+ String getDestination()
+ LocalDateTime getTime()
+ Boolean getBus()
+ Boolean getTrain()
+ void setOrigin(String: origin)
+ void setDestination(String: destination)
+ void setTime(DATETIME: time)
+ void setBus()
+ void setTrain()
+ Boolean findOrigin(String: origin)
+ Boolean findDestination(String: desination)
History
-Route h
+ Route getHist()
+ void storeHist(Route r)
Favourites
-Route f
+ Route getFav()
+ void delFav(Route r)
+ void saveFav(Route r)
Fare
-int busNo
-String mrtStation
-double busFare
-double mrtFare
-double time
-int distance
+ double getBusFare(int busNo)
+ double getMRTFare(string mrtStation)
+ int getDistance()
+ double setBusFare(int busNo)
+ double setMRTFare(string mrtStation)
Data_gov Manager
-double busFare
-double mrtFare
-int distance
-int bus
- String mrtStation
+ double dataGovBusFare(int bstopnum, int: startstopnum, int: endstopnum)
+ double getMRTFare(string mrtStation)
<<uses>>
Data_gov API
Map
-center
-zoom
+ void displayRoute()
+ void displayStop()
+ void displayTrip()
+ void refresh()
<<calls>>
GoogleMaps Manager
-double startTime
-double endTime
-String origin
+ geocodegeocodeLocation(String: startlocation, String: end_location)
+ gmapsDirection(String: origin, String: destination)
+ Boolean findOrigin(String: origin)
+ Boolean findDestination(String: desination)
<<uses>>
GoogleMaps API
[map methods]
[route methods]
+ gmapsDirection(String: origin, String: destination)
[geocode methods]
LTA Manager
-int busNo
-int distance
+ int getDistance(int: bus, int :distance)
+ LocalDateTimeltaBusArrival(int: busstopno, int : busnum)
+ geocodeBus(int: bus)
<<uses>>
LTA API
+ LocalDateTimeltaBusArrival(int: busstopno, int : busnum)
MRT Tan Manager
+ int getMrtArrival(String: mrtStation)
+ int getDistance(String: mrtStation1, String: mrtStation2)
<<uses>>
MRT Tan API

2.4 Operating Environment

The application will operate on Android and will make use of internal storage. The application will interact with external APIs such as Google's Directions API, Google Maps API and data.gov API.

2.5 Design and Implementation Constraints

There are a few constraints owing to the data sources that the application if using currently. The Google Maps API limits the number of routes it returns and so the user has a limited option of routes to select from. In addition, the application only supports English language for now. It is built only for the Singapore public transport public system now. The application can only be used on devices which enable location services. Without that, the app will still be usable to a certain extent, but certain functionalities will be restricted.

2.6 User Documentation

A user manual with detailed explanation of the functionalities of the various components of the application for the user to get hands-on access for is attached along with the deliverables.

2.7 Assumptions and Dependencies

As the application depends on information available through external APIs it is assumed that this information will remain available and in the current format. Any changes to these two factors would affect the applications ability to retain data and provide the user with route options and/or route fares.

3. External Interface Requirements

3.1 User Interfaces

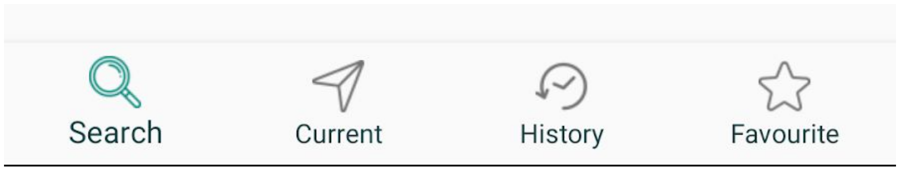


Figure 1 - Navigation bar

Figure 1 shows the navigation bar. The navigation bar is situated at the bottom of the screen and is present in every user interface. It enables the user to navigate to a search page, to see current location on map, to view history of routes travelled and to view routes saved to favorites.

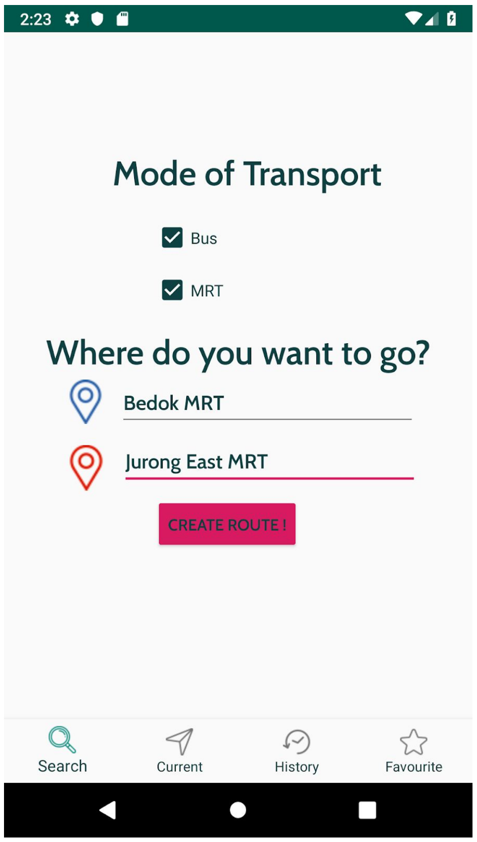
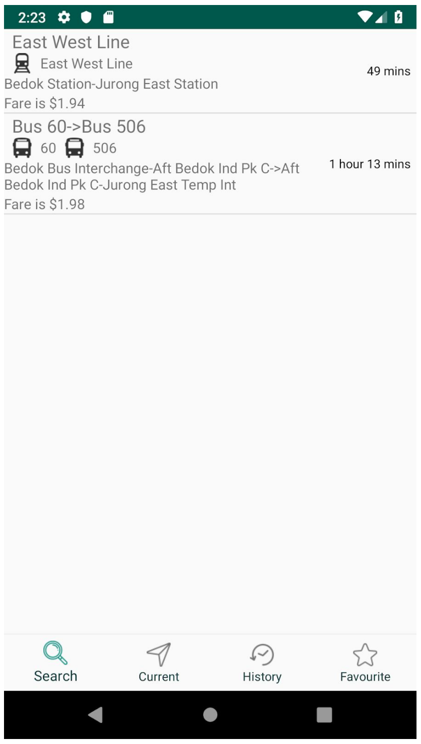


Figure 2 - Search page

Figure 2 shows the interface for entering route details. It enables the user to choose desired mode of transportation and enter origin and destination. Furthermore, it gives the option to choose whether the route should be planned based on time of departure or arrival. Lastly, it presents the user with the ability choose time and date. The “Create Route” button will transfer the user to the next page.


2:23

East West Line
East West Line
Bedok Station-Jurong East Station
Fare is $1.94
49 mins

Bus 60->Bus 506
60 506
Bedok Bus Interchange-Aft Bedok Ind Pk C->Aft Bedok Ind Pk C-Jurong East Temp Int
Fare is $1.98
1 hour 13 mins

Figure 3 - Route alternatives

Figure 3 shows the interface for route alternatives. The interface shows the search results from which the user can select one route. Each result contains information about route and the fare. To proceed, the user can simply click one one route and then start their journey

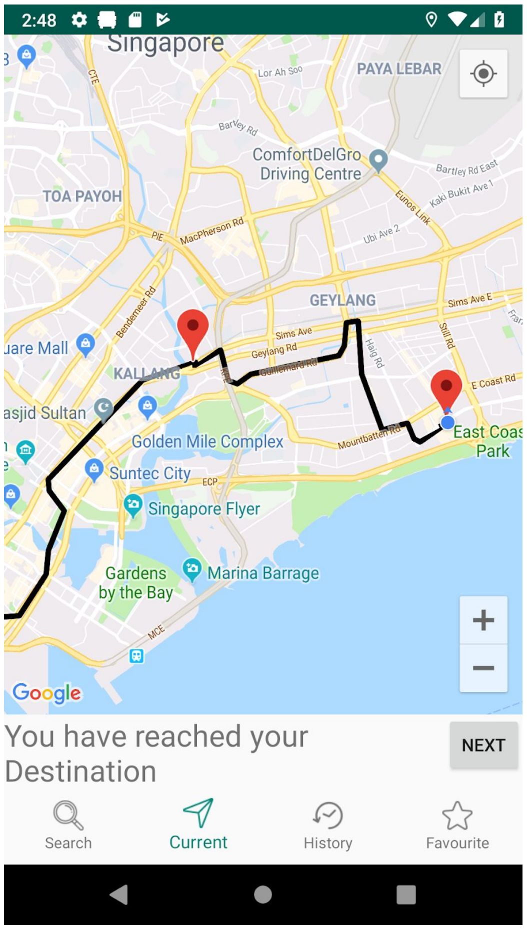
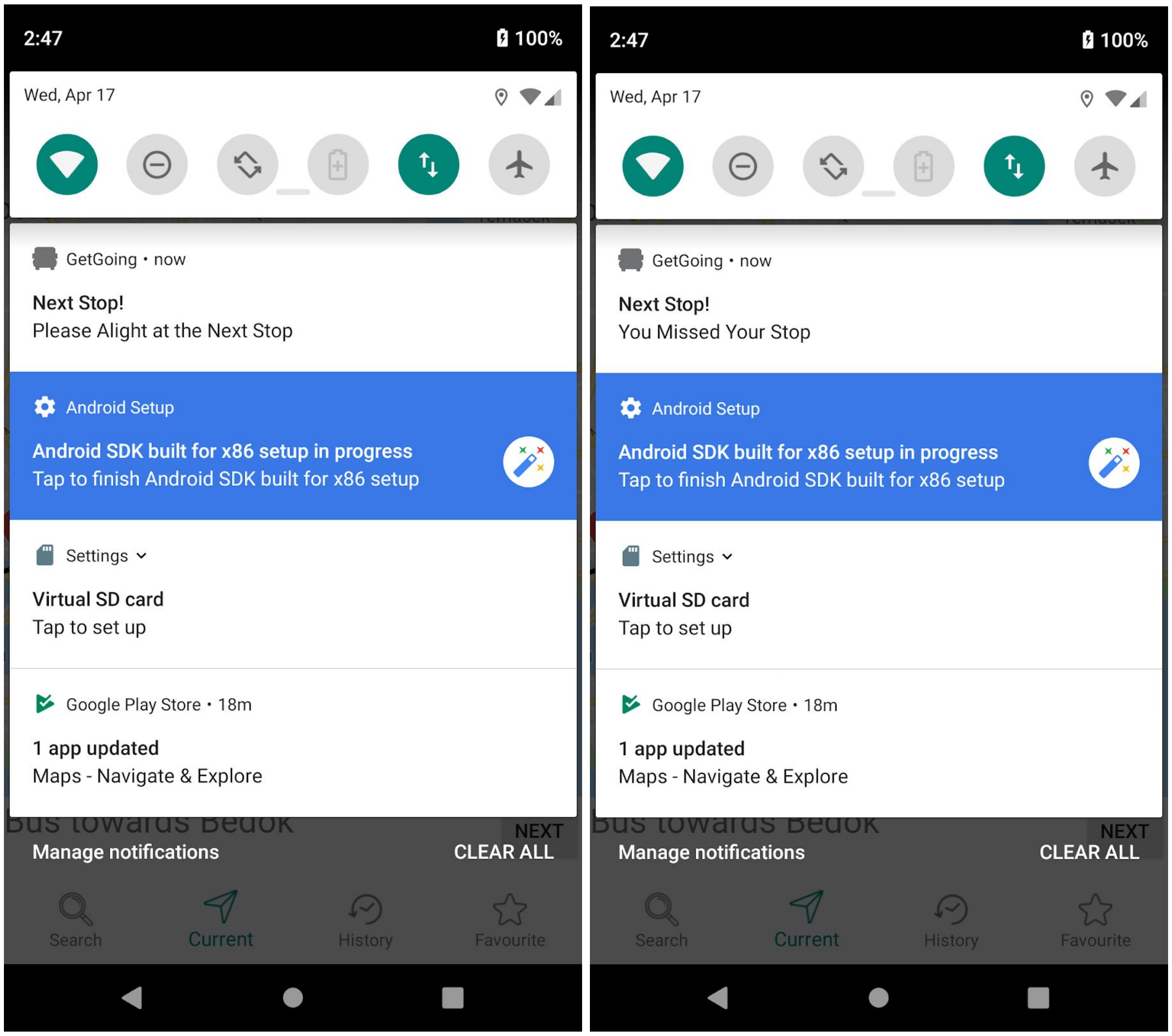


Figure 4 - Current Route

Figure 4 shows the layout of how the screen shows when the user starts a route. A descriptive map locates the user's journey. When the user has reached the destination, a pop-up as shown is displayed.



2:47 100%

Wed, Apr 17

GetGoing • now

Next Stop!
Please Alight at the Next Stop

Android Setup

Android SDK built for x86 setup in progress
Tap to finish Android SDK built for x86 setup

Settings

Virtual SD card
Tap to set up

Google Play Store • 18m

1 app updated
Maps - Navigate & Explore

Manage notifications CLEAR ALL

Search Current History Favourite

2:47 100%

Wed, Apr 17

GetGoing • now

Next Stop!
You Missed Your Stop

Android Setup

Android SDK built for x86 setup in progress
Tap to finish Android SDK built for x86 setup

Settings

Virtual SD card
Tap to set up

Google Play Store • 18m

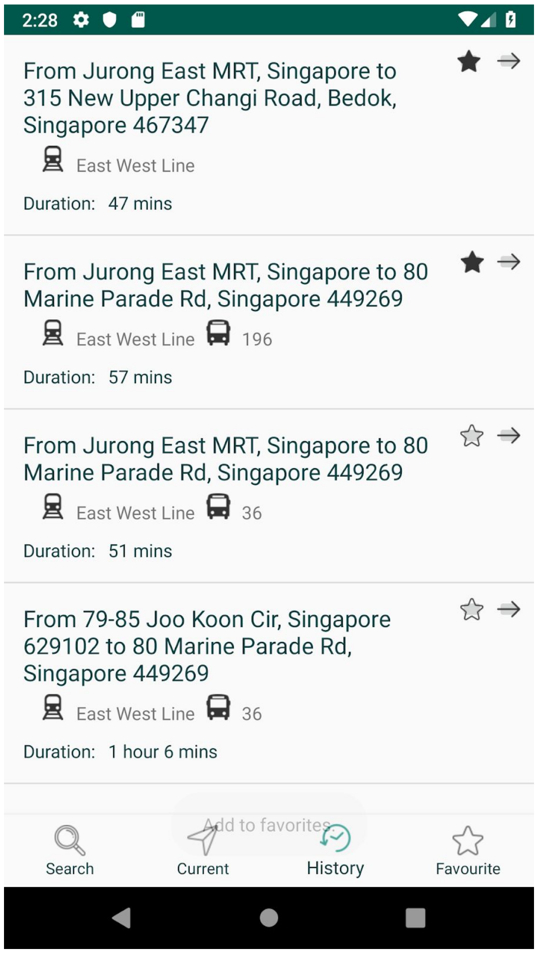
1 app updated
Maps - Navigate & Explore

Manage notifications CLEAR ALL

Search Current History Favourite

Figure 5 - Alert alight / Screen if missed alight

Figure 5 illustrates how the user is notified whether a stop has been missed or when to alight from the bus.

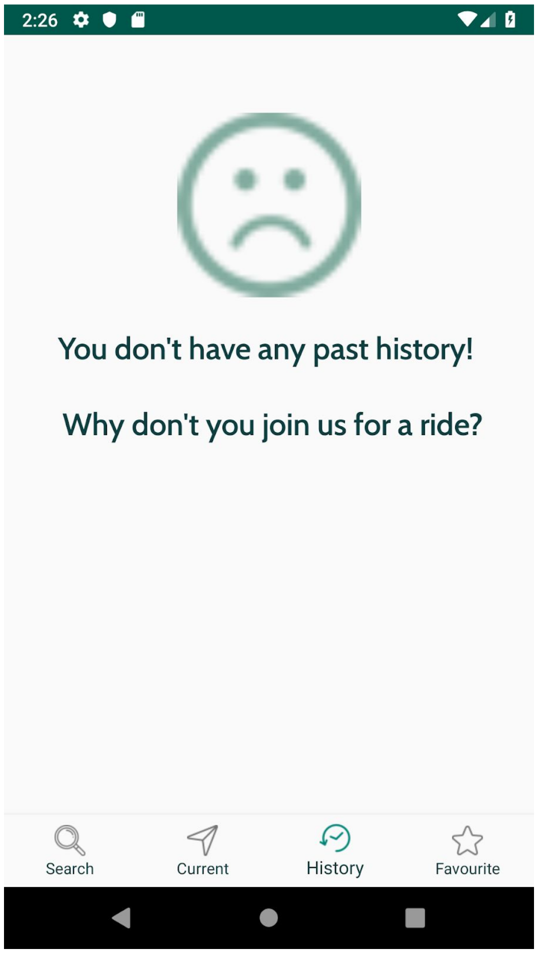

From Jurong East MRT, Singapore to 315 New Upper Changi Road, Bedok, Singapore 467347
East West Line
Duration: 47 mins

From Jurong East MRT, Singapore to 80 Marine Parade Rd, Singapore 449269
East West Line 196
Duration: 57 mins

From Jurong East MRT, Singapore to 80 Marine Parade Rd, Singapore 449269
East West Line 36
Duration: 51 mins

From 79-85 Joo Koon Cir, Singapore 629102 to 80 Marine Parade Rd, Singapore 449269
East West Line 36
Duration: 1 hour 6 mins

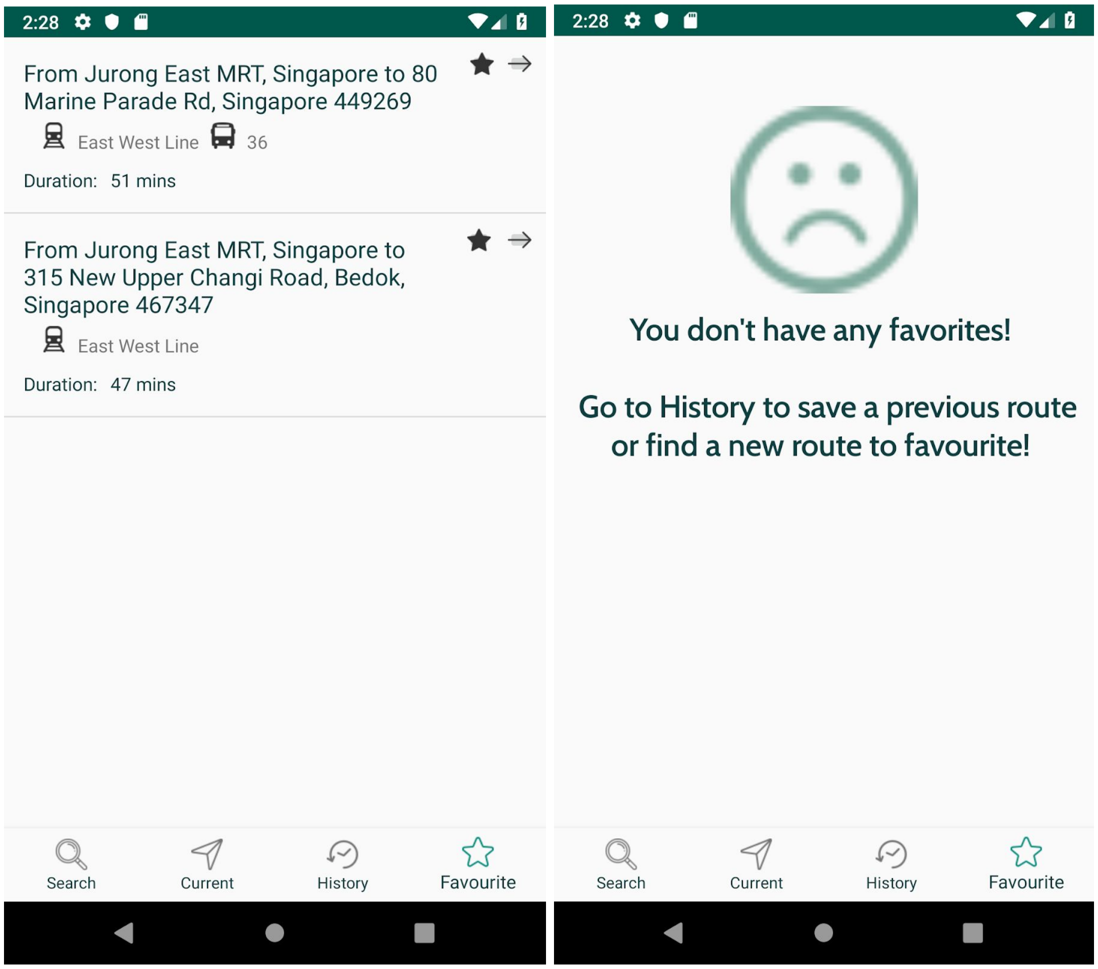
Add to favorites


You don't have any past history!

Why don't you join us for a ride?

Figure 6 - No history / History / Favorited route

Figure 6 shows the interface for routes in History. If the user hasn't travelled with the application there will be no routes saved to History and the user will be informed of this. If there are available routes, they will be displayed here and the user can see origin, destination, mode of transportation for a particular route travelled. The interface contains an arrow symbol which will redirect the user to the search page. The star symbol symbolizes whether a route is saved in Favorites or not.


2:28

From Jurong East MRT, Singapore to 80 Marine Parade Rd, Singapore 449269
East West Line 36
Duration: 51 mins

From Jurong East MRT, Singapore to 315 New Upper Changi Road, Bedok, Singapore 467347
East West Line
Duration: 47 mins

You don't have any favorites!

Go to History to save a previous route or find a new route to favourite!

Figure 7 - No favorite / Favorites / Deleted favorite

Figure 7 shows the interface for routes in Favorites. If the user haven't selected any favorite routes from History the page will inform the user of this. If there are routes saved to Favorites, they will be displayed on here. The interface contains an arrow symbol which will redirect the user to the search page. The star symbol shows that the route is a favorite - if clicked on will delete route from Favorites.

Figure 8 - Route selected from History / Favorite

Figure 8 shows the interface for when the user has selected a route from History or Favorites. The search page will be automatically filled with the information saved in History / Favorites for the selected route.

3.2 Hardware Interfaces

In the map interface, GetGoing requires to work on hardware devices with location service enabled to locate the user's current location and alert when it is time to alight. Furthermore, whenever the user has travelled a route and saved/deleted a route to/from Favorites, the software will incur changes in the data storage of the device. For the time being, the application is only runnable on a mobile device and on emulators for the same.

3.3 Software Interfaces

App version 1.0.0
Our application will access data shared by APIs of Google and Smart Nation. We send APIs requests with parameters for the type of route we require such as the origin, destination, mode of travel, time of departure or arrival. The API sends us back an object with contains all details of the route. In addition, the Smart Nation API sends us the fare for a certain route once we provide it information about the route itself. The application calls the Google Maps component to display the route the user selects onto a map. We also interact with the operating system of the device so as to display push notifications for the alert alight service. When dealing with saved routes or favourites, our app interacts with the session storage to save, load and modify locally stored user specific data.

4. System Features

4.1 Enter A Route

4.1.1 Description and Priority

The User enters the information for the journey, including mode of transportation, origin, destination, and time of departure/arrival. Before using this feature, the user must have already downloaded the application. After successful input, by default, the fastest route will be calculated and displayed on the screen.

4.1.2 Stimulus/Response Sequences

1. The user can enter route information from the "Search" page, or
2. The user can enter route information from the "Favorite" page by clicking the route desired to follow (by clicking the next arrow button). After a user selects the route, the page is redirected to the "Search" page and the information is pre-filled in the form, or
3. The user can enter route information from the "History" page by clicking the route desired to follow (by clicking the next arrow button). After a user selects the route, the page is redirected to the "Search" page and the information is pre-filled in the form

4.1.3 Functional Requirements

1. The user shall be able to choose mode of transportation:
    - 1.1. Mode of transportation includes:
        - 1.1.1. Bus
        - 1.1.2. MRT
2. The user must be able to enter origin in different formats:
    - 2.1. Formats include:
        - 2.1.1. address
        - 2.1.2. name of Bus stop

        2.1.3. name of MRT stop
3. The user must be able to enter destination in different formats:
    3.1. Formats include:
        3.1.1. address
        3.1.2. name of Bus stop
        3.1.3. name of MRT stop
4. The user shall be able to plan route by time of arrival
5. The user shall be able to plan route by time of departure
6. The user must be able to choose a time of travel
7. The user must be able to choose a date of travel
8. The user must be able to plan a trip with immediate departure
9. The user must be able to plan a trip with departure in the future

4.2 Calculate A Route

4.1.1 Description and Priority

The app registers user input from the Enter A Route form page and calls the respective Google DIrection, API for route calculation and dataGov api to calculate the fares of each route

4.1.2 Stimulus/Response Sequences

No user actions are required for this feature as it is done by the system

4.1.3 Functional Requirements

1. The system shall convert the user input into data.
    1.1. The system must parse a manual address into a geolocation using relevant API.
    1.2. If user requests live location, the system shall use location services to detect the location of the user.
    1.3. Date and time entered by the user shall be converted to the desired format for processing.
2. The system shall perform computations to calculate routes for the user.
    2.1. System shall retrieve travel fare information from Data.gov.sg - Fares for Feeder Bus Services / Fares for Express Bus Services
    2.2. System shall calculate list of available routes using Google Directions API and Google Maps API.
    2.3. System shall provide the list of routes to the user
    2.4. System shall display the cost of each route
    2.5. System shall display the time taken for each route
    2.6. During the trip, the system shall have additional functionalities.
        2.6.1. System shall be able to track a specific vessel in real time
        2.6.2. System shall display the vessel movement in real time
        2.6.3. System shall make suggestions if the stop is missed
        2.6.4. The system must alert the user when the next stop is the destination
    2.7. System must be able to store one or more “Favorite” routes

4.3 Favorite A Route

4.1.1 Description and Priority

The user should be able to favorite a route from the “History” page.

4.1.2 Stimulus/Response Sequences

1. The user goes to the “History” page
2. The user clicks the “favorite” (star) button to favorite that specific route.
3. A notification appears informing the user that the input has been registered and the route has been favorited. The star button is also shaded (to show that is in favorites).
4. The user goes to the “Favorite” page.
5. The user notices that the route they just favorited has appeared in the “Favorite” page and that the star button is shaded (to show that is in favorites).

4.1.3 Functional Requirements

1. The system must warn the user if there are no routes saved in history
2. The user must be able to view all routes travelled in history
3. The user must not be able to delete a route from history
4. The user shall be able to select a route as a favorite
5. The system must be able to save a route as a favorite
6. The system must be able to show that a route has been favorited
7. The user must be able to start a selected route from history
8. The system must be able to redirect the user to the search page
9. The system must be able to pass saved route information to search page when initiated by user

4.4 Unfavorite A Route

4.1.1 Description and Priority

The user should be able to unfavorite a route from the “Favorites” page. This requires that the user has already selected a favorite from the History page.

4.1.2 Stimulus/Response Sequences

6. The user goes to the “Favorite” page
7. The user clicks the “favorite” (star) button to unfavorite that specific route.
8. A notification appears informing the user that the input has been registered and the route has been unfavorited. The route is deleted from the “Favorites” page.
9. The user goes to the “History” page.
10. The user notices that the route they just unfavorited has an unshaded star button (to show that is not in favorites).

4.1.3 Functional Requirements

1. The system must be able to warn the user when there are no favorite routes saved
2. The user should be able to view favorited routes
3. The user should be able to un-favorite a route
4. The system must be able to delete a route
5. The system must be able to update the status of deleted favorite in the routes saved to history

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

1. 99 % of users must be able to enter a simple query within 2 minutes of starting the system
2. After a system reboot, the restoration to full system functionality must not take longer than 3 minutes.
3. When the user is on the MRT/bus, the system shall to detect if the next stop is the destination within 1 minute.
4. System shall be able to calculate cheapest/ fastest route within 1 minute
5. Installation of the application must not take more than 5 minutes.
6. On missing the transport, the app should recalculate a new route within 2 minutes.
7. System response time must not exceed 1 minute.

5.2 Safety Requirements

Our application does not concern any additional safety breaches and does not have any additional safety requirements from the user.

5.3 Security Requirements

Data APIs used as public by the various organizations which provide data without any security concerns and no user authentications required. In the current version, there is no database created as all details are stored in local storage, thereby there can be no breach of security unless there are external factors causing loss of information in the user’s device. No location information is stored in a centralised database for the application.

5.4 Software Quality Attributes

As many users should be able to concurrently access the app as many requests are allowed by our APIs to make concurrent requests to. The route should be displayed in the right manner on the Maps component and to scale so as to not misdirect the user. The admin team should maintain the app to update API calls when APIs are updated. Location services should be accurate on the map with respect to the route displayed. The alert alight should be notified on time to make full use of the application.

6. Future Goals

6.1 Adopt a route planning feature

For example, user wants to plan a route to leave an hour later :
Step 1: Enter Route Information
Step 2: System suggests list of possible routes
Step 3: If user wants to take a route, he/she can set a reminder and the app will inform the user when to leave when appropriate.

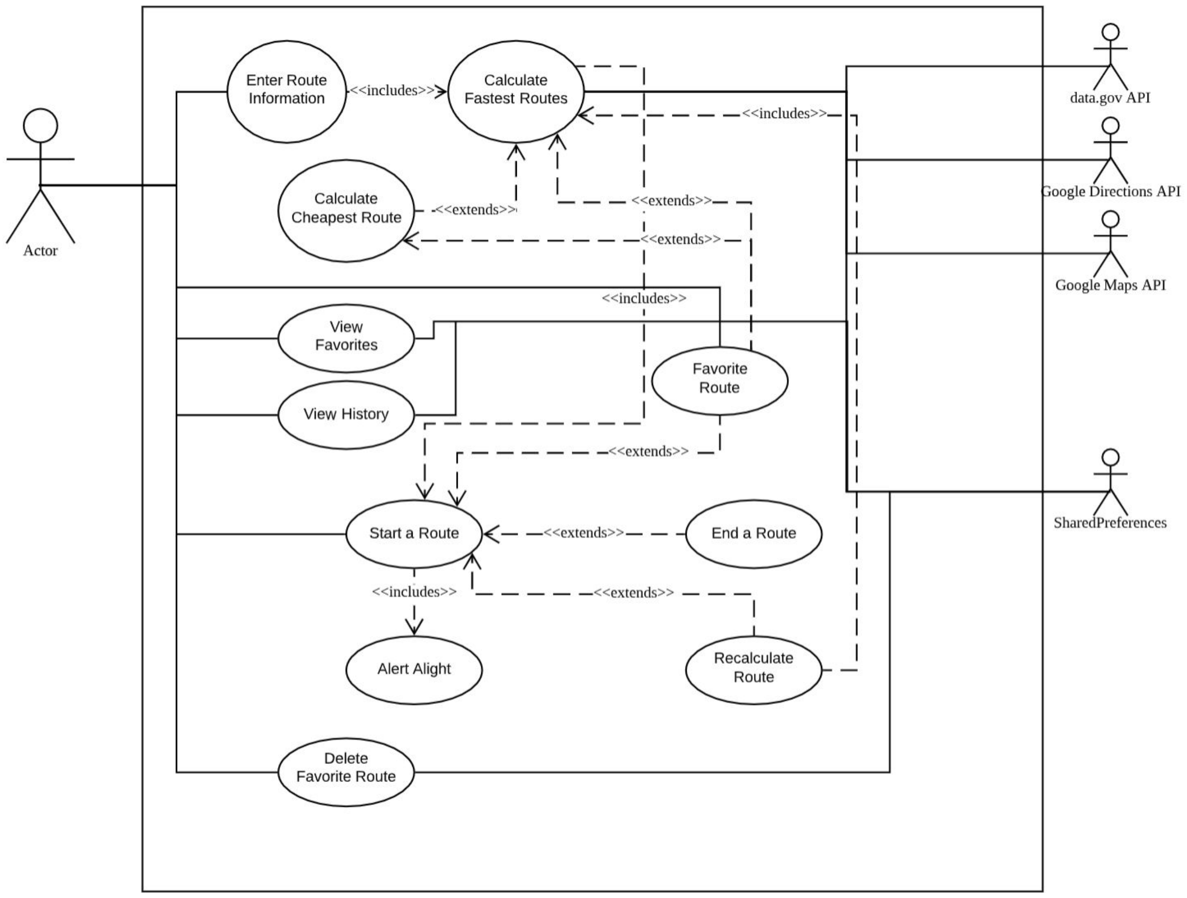
6.2 Adopt a sort by price/duration feature

Step 1: Enter Route Information
Step 2: System suggests list of possible routes
Step 3: User can choose to sort the given suggestion routes by price or duration. (new!)
Step4: System displays list of routes but sorted by price/duration.

6.3 Implement search Suggest/autocomplete

Step1: System would provide suggestion to users as they enter their search queries for origin/destination.
Step2: If user enters origin/destination that would not be understood by the API, system would convert it to a format understandable by API

This would allow easier searching of routes and users can simply enter postal codes as opposed to now when they want to search 637719 which is a postal code in singapore, no results would be returned by the API but need precise typing like Singapore 637719.

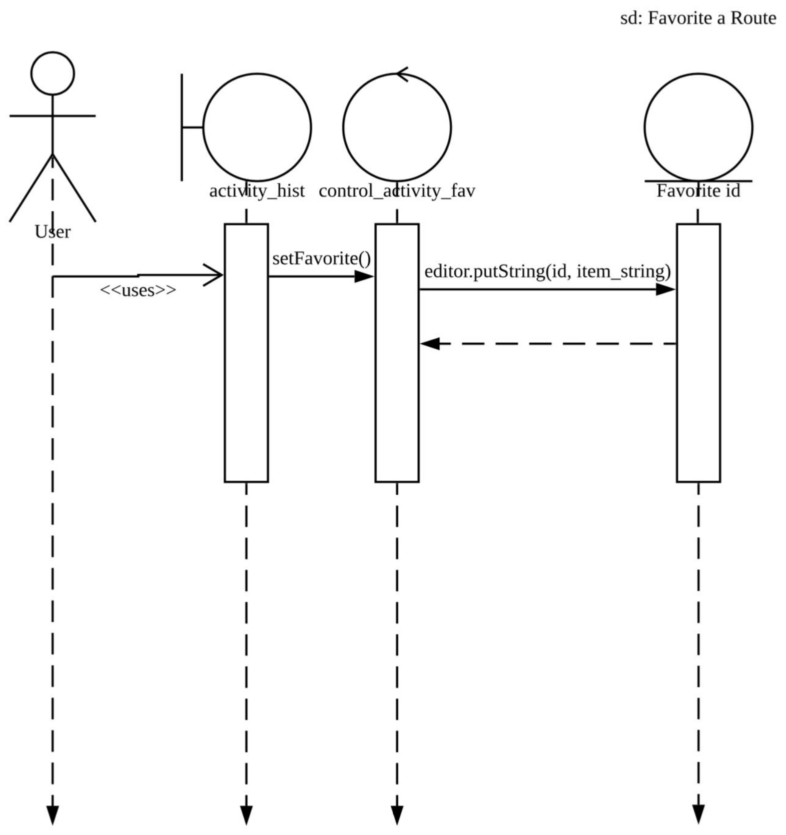

Actor
Enter Route Information
<<includes>>
Calculate Fastest Routes
<<includes>>
Calculate Cheapest Route
<<extends>>
<<extends>>
<<extends>>
<<includes>>
View Favorites
View History
Favorite Route
<<extends>>
Start a Route
<<extends>>
End a Route
<<includes>>
<<extends>>
Alert Alight
Recalculate Route
Delete Favorite Route
data.gov API
Google Directions API
Google Maps API
SharedPreferences

Appendix A: Data Dictionary

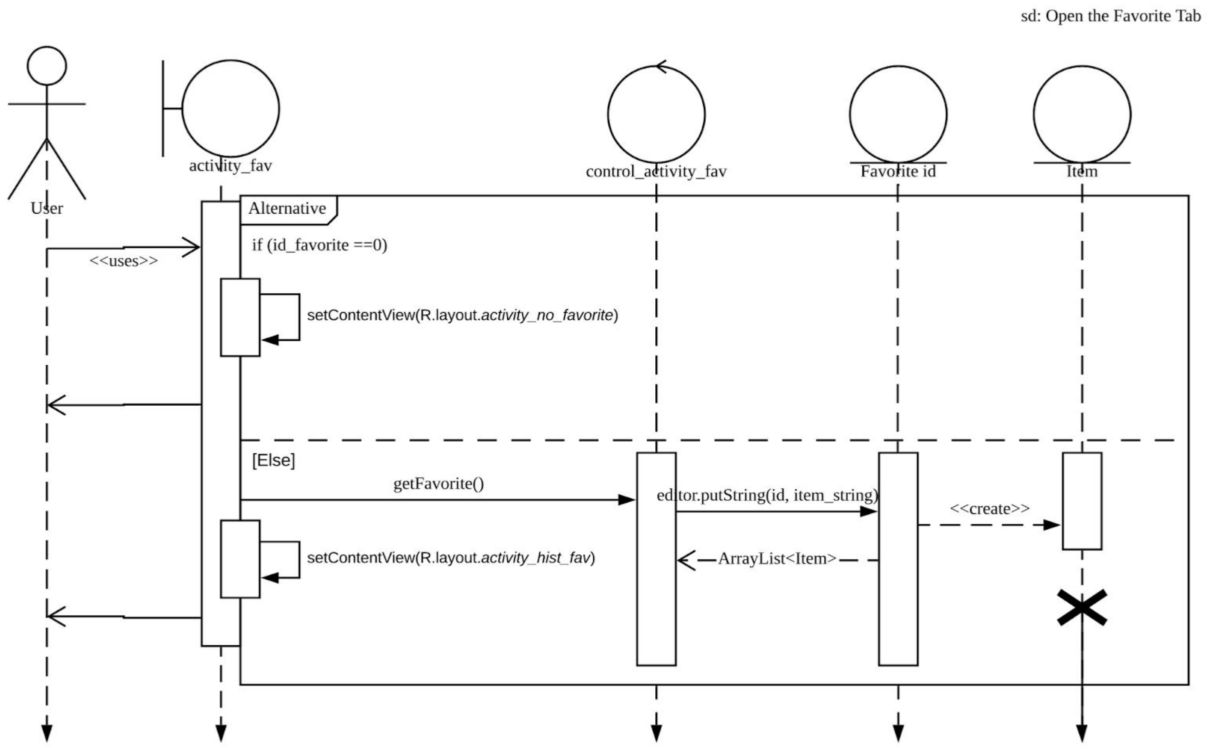
Term	Definition
Bus	A large motor vehicle carrying passengers by road, serving the public on a fixed route for a fare.
Bus Stop	A place where the bus regularly stops, usually marked by a sign. The bus stop can be represented by its name or code.
Mass Rapid Transport (MRT)	Fast moving commuter train that runs on a rail, bringing commuters from one station to another.
MRT Stop	A place where the MRT regularly stops.
User	People in Singapore who are trying to make use of public transport to get from one place to another.
System / Product	The mobile application.
Input	Information of different forms. This information could for example the name of a bus stop or departure time.
Output	The result of the query created by the user from the database
Origin	The bus stop or MRT station from which the user would like to travel.
Destination	The bus stop or MRT station the user would like to travel to.
Route	The route is a geographical description of how the user will get from the origin to the destination.
Possible routes	The different routes a user can take to get from origin to destination.
Favorites	An option available to users which enables them to save specific routes.
Mode of transportation	The users will be able to travel by bus or MRT. As such, these are two types of modes of transportation.
Departure Time	The point in time when the vessel will leave from the origin.
Arrival Time	The point in time when the vessel will reach the destination.

User Location	The coordinates of the user .This location is used to track the users movements and alert when they are approaching the destination.
Distance	Distance travelled during trip.
Fare	The price for a specific trip.
Price	The price the user will have to pay for their trip.
Cheapest route	The route holding the cheapest price the user can pay for getting from origin to destination.
Fastest route	The route representing the shortest travel time

Appendix B: Sequence Diagrams

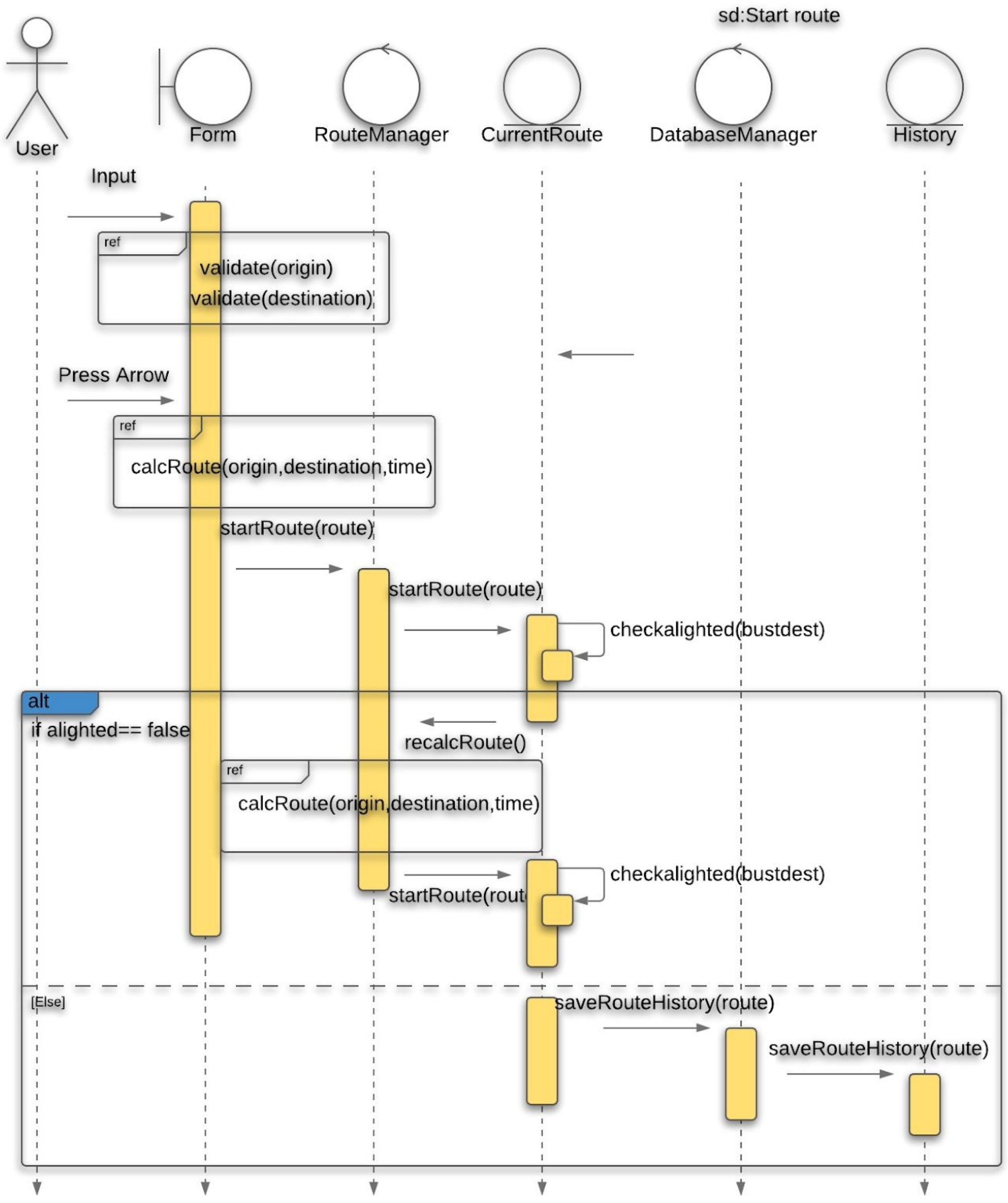


The figure on the left illustrates a sequence diagram of how a route is favorited


sd: Open the Favorite Tab
User
activity_fav
control_activity_fav
Favorite id
Item
Alternative
if (id_favorite ==0)
<<uses>>
setContentView(R.layout.*activity_no_favorite*)
[Else]
getFavorite()
editor.putString(id, item_string)
<<create>>
setContentView(R.layout.*activity_hist_fav*)
ArrayList<Item>

The figure above illustrates a sequence diagram of how the “Favorite” page is opened — i.e how the route is extracted from SharedPreferences— our storage unit under Android Studio— and is displayed on the user’s screen.

3) Starting a route



Appendix C: Use Case Descriptions

Route-1: Enter route information

Use Case ID:	ROUTE-1		
Use Case Name:	Enter route information		
Created By:	Isabel	Last Updated By:	Grace
Date Created:	2019-02-25	Date Last Updated:	2019-04-16

Actor:	User
Description:	The User enters the information for the journey, including mode of transportation, origin, destination, and time of departure/arrival.
Preconditions:	The user has downloaded the application.
Postconditions:	By default, the fastest route will be calculated.
Priority:	High
Frequency of Use:	Once a week
Flow of Events:	1. The user checks the desired box/boxes to choose mode of transportation , enters origin and destination information in the respective fields. 2. The user then clicks on a button to calculate the route from origin to destination. 3. System checks if at least one mode of transportation has been chosen 4. System checks if origin fill is not empty 5. System checks that destination is not empty
Alternative Flows:	AFS-3: If the user did not choose at least one mode of transport

	1. System display the message requesting the user to at least choose one mode of transport 2. System returns to step 1. AFS-4/5: If the user did not enter any origin/destination 1. System displays the message requesting the user to enter a origin/destination 2. System returns to step 1.
Exceptions:	
Includes:	ROUTE-2 Calculate Fastest Routes
Special Requirements:	1. 99 % of users must be able to complete ROUTE-1, ROUTE-2, and possibly ROUTE-3 within 2 minutes of starting the system
Assumptions:	
Notes and Issues:	

Route-2: Calculate Routes

Use Case ID:	ROUTE-2		
Use Case Name:	Calculate Routes		
Created By:	Grace	Last Updated By:	Isabel
Date Created:	2019-02-25	Date Last Updated:	2019-03-12

Actor:	System
Description:	Displaying for the user up to 4 different routes to get from origin to destination by bus and/or MRT
Preconditions:	The user must have completed Use Case ROUTE-1
Postconditions:	Display the various routes and their timing needed
Priority:	High
Frequency of Use:	Once a week
Flow of Events:	1. System queries Google Directions API and returns up to 4 alternative route information 2. The system reads the estimated duration time required for each possible route. 3. System extracts the bus and mrt that need to be taken 4. System displays Route information such as the duration needed for each route, the train/bus taken.
Alternative Flows:	AFS2: if no route could be found
Exceptions:	
Includes:	This use case includes ROUTE-3

Special Requirements:	1. 99 % of users must be able to complete ROUTE-1, ROUTE-2, and possibly ROUTE-3 within 2 minutes of starting the system 2. System shall be able to identify the fastest route within 1 minute
Assumptions:	System would always be able to find a route from users origin to their destination
Notes and Issues:	If the user is planning a journey when MRT/bus services are not available (e.g outside of operating hours), the system will show the next possible route (e.g when services start operating again).

Route-3: Calculating fares of Routes

Use Case ID:	ROUTE-3		
Use Case Name:	Calculate Cheapest Route		
Created By:	Eugene	Last Updated By:	Isabel
Date Created:	2019-02-25	Date Last Updated:	2019-04-14

Actor:	User
Description:	Displaying for price for each individual route
Preconditions:	Route1 and Route2
Postconditions:	Display the various routes and their associated travel time
Priority:	High
Frequency of Use:	Once a week
Flow of Events:	1.System extracts the distances information gotten from google directions API 2. System calls DataGov API to calculate the price of each route 3. System display fare information
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	1. 99 % of users must be able to complete ROUTE-1, ROUTE-2, and possibly ROUTE-3 within 2 minutes of starting the system 2. System shall be able to identify the cheapest route within 1 minute

Assumptions:	
Notes and Issues:	

Route-4: Add a favorite route

Use Case ID:	ROUTE-4		
Use Case Name:	Add favorite route		
Created By:	Surabhi	Last Updated By:	Surabhi
Date Created:	2019-03-10	Date Last Updated:	2019-04-15

Actor:	User (initiating) & System
Description:	Allows the user to save routes as favorites for future use
Preconditions:	This use case extends ROUTE-2 and ROUTE-3. It is initiated when the user clicks on symbol next to route. The user must have travelled the route so that it is added to history.
Postconditions:	Route is added to Favourite list Route saved showed as saved in History and Favorites
Priority:	Medium
Frequency of Use:	Once a month
Flow of Events:	1. The user clicks the “save” button (a star button) next to the desired route to save from “History”. 2. The system saves the origin, destination, mode of transport and to internal storage 3. System displays a message or indication to the user to inform that the route has been added to favorites. 4. The route favorited is shown as saved in “History” and “Favorites” 5. The favorited route is now viewable in “Favorites”
Alternative Flows:	
Exceptions:	
Includes:	

Special Requirements:	System response time must not exceed 30 seconds
Assumptions:	
Notes and Issues:	

Route-5: Delete Favorite Route

Use Case ID:	ROUTE-5		
Use Case Name:	Delete favorite route		
Created By:	Surabhi	Last Updated By:	Surabhi
Date Created:	2019-03-10	Date Last Updated:	2019-04-15

Actor:	User
Description:	Allowing the user to delete routes from Favorites
Preconditions:	This use case extends ROUTE-4. The user must have at least one favorite route saved.
Postconditions:	Route is deleted from Favorites. Route deleted shows as unsaved in ”History”
Priority:	Low
Frequency of Use:	Once a month
Flow of Events:	1. The user clicks on “Favorites” in the navigation bar 2. The system shows a list of routes the user has favorited 3. The user selects a route to be deleted by clicking on a the unsave button 4. The system deletes the route from “Favorites” 5. System displays a message “Route deleted from favourites” 6. If user views “History”, the route just unfavorited is shown as unsaved.
Alternative Flows:	AFS 2-4: If there are no favorites to display 1. The system will display a page informing the user that there are no available favorite routes
Exceptions:	

Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	Users cannot delete a route from History.

Route-6: Start a Route

Use Case ID:	ROUTE-6		
Use Case Name:	Start a Route		
Created By:	Shub	Last Updated By:	Grace
Date Created:	2019-03-10	Date Last Updated:	2019-04-16

Actor:	User
Description:	Display information for selected route
Preconditions:	The user must have completed use case ROUTE-1 or completed use case Route-10 or completed use case Route-11. (ie user either just calculated a route, or start a route from one of the routes in the favourites and history
Postconditions:	
Priority:	High
Frequency of Use:	Daily
Flow of Events:	1. The user selects their preferred route from list of routes. 2. The system calls Google Maps API and displays a map showing the user current location as well the polyline of the chosen route 3. System constantly updates the map shown based on changes in user location. 4. If the route involves taking a bus, when the user nears the bus stop they need to alight, the system prompts the user to alight (use case Route-7 Alert Alight)

Alternative Flows:	
Exceptions:	EX1: User decides to cancel route 1. System displays " You have chosen to end the route"
Includes:	This use case includes ROUTE-7
Special Requirements:	System response time must not exceed 1 minute
Assumptions:	GPS tracking is fairly accurate
Notes and Issues:	

Route-7: Alert alight

Use Case ID:	ROUTE-7		
Use Case Name:	Alert alight		
Created By:	Shubh	Last Updated By:	Isabel
Date Created:	2019-03-10	Date Last Updated:	2019-03-12

Actor:	User
Description:	Real time alerting of user when to alight from the bus
Preconditions:	Route chosen involves at least one bus taken.
Postconditions:	User has alighted at the correct stop
Priority:	High
Frequency of Use:	Daily
Flow of Events:	1. The application constantly queries for the user current location 2. The application uses the user's location to determine if the user is approaching the intended bus stop 3. When application determines the user is approaching the intended bus stop, the application issues a notification to the user 4. User successfully alights and click next for the next instruction to do (ie take another bus 1 from this stop)
Alternative Flows:	AF-S4: User misses the stop and did not click next 1. The system sends a notification to the user telling them that they have missed their bus stop AF-S4: User has completed the route

	1. Application notifies user “You have arrived at Your Destination!”
Exceptions:	
Includes:	
Special Requirements:	On missing the transport, the app should recalculate a new route within 2 minutes.
Assumptions:	GPS tracking is fairly accurate
Notes and Issues:	

Route-8: End a Route

Use Case ID:	ROUTE-8		
Use Case Name:	End a Route		
Created By:	Surabhi	Last Updated By:	Isabel
Date Created:	2019-03-12	Date Last Updated:	2019-03-12

Actor:	User
Description:	Allows the user to end a route.
Preconditions:	This use case extends ROUTE-6.
Postconditions:	Route has ended
Priority:	High
Frequency of Use:	Daily
Flow of Events:	1. User has completed the route. 2. The system shows feedback “Route has been completed. Thank you! Have a nice day” 3. The system adds the route to history of journeys in database.
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	From here, the user can start a new route

Route-9: Recalculate Route

Use Case ID:	ROUTE-9		
Use Case Name:	Recalculate Route		
Created By:	Surabhi	Last Updated By:	Isabel
Date Created:	2019-03-10	Date Last Updated:	2019-03-12

Actor:	User
Description:	User has missed a stop.
Preconditions:	User must be on a route and has not followed directions. This use case extends ROUTE-6
Postconditions:	User is assigned a new route
Priority:	High
Frequency of Use:	Daily
Flow of Events:	1. If a user has missed a route, system calculates a new route to destination. 2. The system displays new instructions for the user to follow.
Alternative Flows:	
Exceptions:	
Includes:	This use case includes ROUTE-2.
Special Requirements:	On missing the transport, the app should recalculate a new route within 2 minutes.
Assumptions:	
Notes and Issues:	

Route-10: View History

Use Case ID:	ROUTE-10		
Use Case Name:	View History		
Created By:	Isabel	Last Updated By:	Isabel
Date Created:	2019-04-14	Date Last Updated:	2019-04-14

Actor:	User (initiating) & System
Description:	Allows the user to view all the routes he/she has travelled using the application.
Preconditions:	The user must have travelled the route so that it is added to history.
Postconditions:	The user can view all the routes previously travelled.
Priority:	Low
Frequency of Use:	Once a month
Flow of Events:	The user clicks on "History" on the navigation bar The system displays routes travelled by the user The system shows the origin, destination and mode of transport for each route
Alternative Flows:	AFS 2-3: There are no saved routes to display 1. The system will display a page informing the user that there are no available favorite routes
Exceptions:	
Includes:	
Special Requirements:	System response time must not exceed 1 minute
Assumptions:	
Notes and Issues:	The user cannot delete a route from History

Route-11: View Favourites

Use Case ID:	ROUTE-11		
Use Case Name:	View Favourites		
Created By:	Grace	Last Updated By:	Grace
Date Created:	2019-04-14	Date Last Updated:	2019-04-16

Actor:	User (initiating) & System
Description:	Allows the user to view all the routes he/she has favourited
Preconditions:	The user must have travelled the route so that it is added to history.
Postconditions:	The user can view all their favourite routes.
Priority:	Low
Frequency of Use:	Once a month
Flow of Events:	1. The user clicks on “Favourite” on the navigation bar 2. The system display routes that user previously favourited 3. The system shows the origin, destination and mode of transport for each route.
Alternative Flows:	AFS 2-3: There are no saved routes to display 2. The system will display a page informing the user that there are no available favorite routes
Exceptions:	
Includes:	
Special Requirements:	System response time must not exceed 1 minute

Assumptions:	
Notes and Issues:	The user cannot delete a route from History

Testing

Black Box Testing

1. **Entering route information and then calculating it**

a) Generic Cases

Test ID	**Scenario**	**Expected Results**	**Actual Results**
1	User clicks calculate route after filling up a valid origin, valid destination having selected at least one mode of transport	The system displays a list of up to 4 route information for the user to view and select preferred route	The system displays a list of up to 4 route information for the user to view and select preferred route
2	User clicks calculate route without filling up the required field	The system prompts the user to fill up the required field before they would be able to proceed	The system prompts the user to fill up the required field before they would be able to proceed
3	User clicks calculate route but origin and destination filled are invalid	The system display an empty page indicating to the users that no route could be found	The system display an empty page indicating to the users that no route could be found

b) Specific Cases(combinations)

Mode of transport	**Origin**	**Destination**	**Expected Results**	**Actual Results**
bus, train	jurong east mrt station	jurong point	Displaying up to 4 alternate route information that can be a combination of taking the bus or mrt / either one	Displayed up to 4 route information that can be a combination of taking the bus or mrt / either one
bus	Jurong east mrt	jurong point	Displaying up to	Displayed up to

	station		4 alternate bus routes information	4 alternate bus routes information
train	jurong east mrt	jurong point	Displaying up to 4 alternate mrt routes information	Displayed up to 4 alternate mrt routes information
Empty(“ did not select any”)	jurong east mrt	jurong point	Please select at least one mode of transport	Please select at least one mode of transport
bus, mrt	Empty(“”)	jurong point	Please enter the origin	Please enter the origin
bus,mrt	Jurong east mrt	Empty(“”)	Please enter the destination	Please enter the destination

2. Alerting Alight

a) to alert alight at 1.30327, 103.906

Changing the GPS of AVD to see if the application would alert the user to alight when they have reached the destination bus stop

Test input (user GPS) Latitude Longitude		**Expected Results**	**Actual Results**
1.340327	103.906	No notification since not near stop	No notification since not near stop
1.30325	103.906	Please Alight at the next stop	Please Alight at the next stop

b) to inform the user after they have missed their bus stop

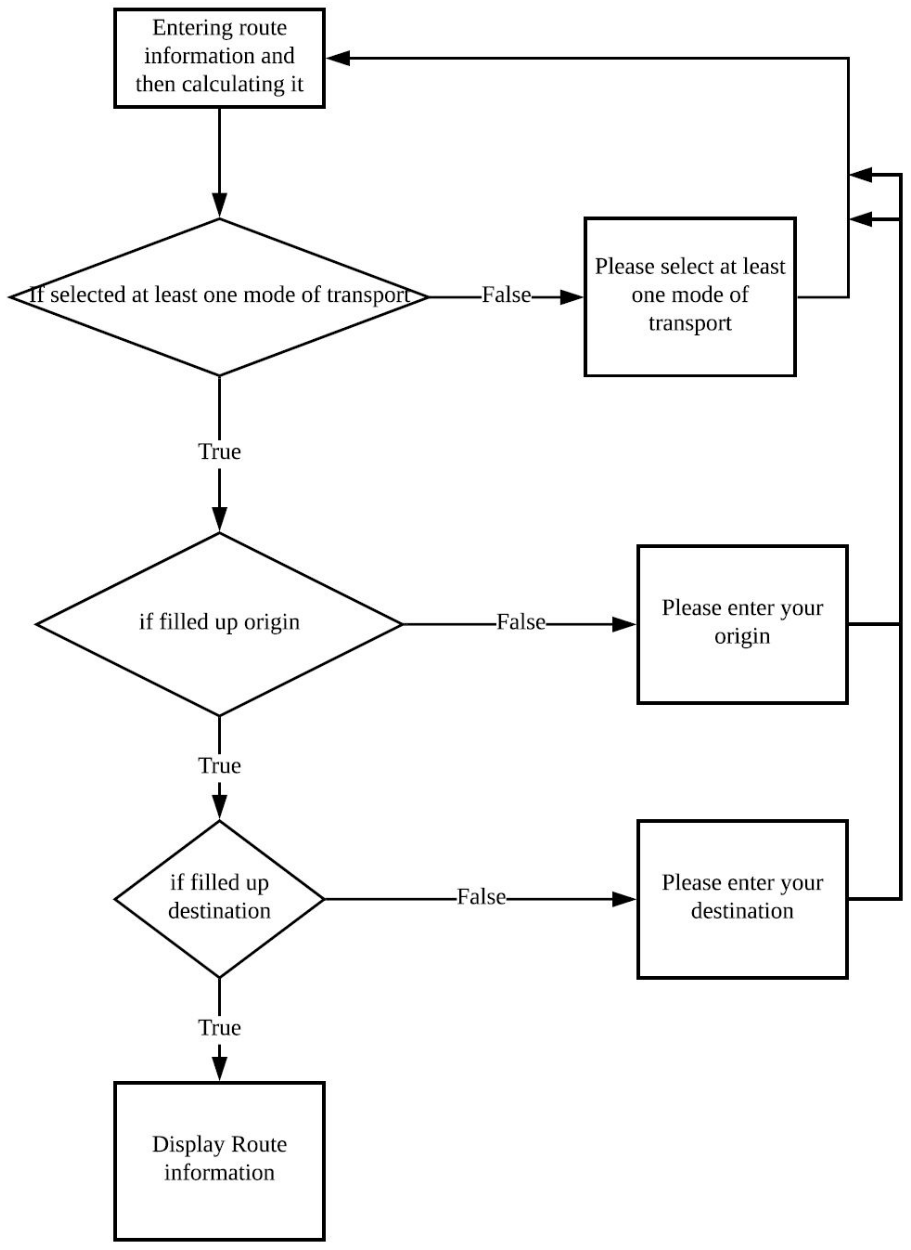
It would only happen if the system have already prompted the user to alight but the user GPS went out of the vicinity of the bus stop very quickly

It only happens when the user location went near the bus stop (ie reach the stop) then slowly goes further away from the stop (and the user did not press next)

Test input (user GPS) Latitude Longitude		**Expected Results**	**Actual Results**
1.30925	103.906	You have missed your stop	You have missed your stop

White Box Testing

1) **entering route information and then calculating routes**


Entering route information and then calculating it
If selected at least one mode of transport
False
Please select at least one mode of transport
True
if filled up origin
False
Please enter your origin
True
if filled up destination
False
Please enter your destination
True
Display Route information

Source: http://www.frontiernet.net/~kwiegers/process_assets/srs_template.doc