Software Requirements Specification

for

MyTripDiary

Version Alpha approved

Prepared by
Tai Chen An
Tan Ming Rui, Ezra
Goel Armaan
Sim Guanyu
Xu Yinfeng
Nepal Aaradh

Nanyang Technological University, Team No Idea

1 Feb 2023

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This document specifies the software requirements for the application MyTripDiary Version Alpha.

MyTripDiary is a Mobile app that enables its users to keep track of their daily commutes allowing them to budget better and save time by informing them about alternative faster, cheaper routes and modes of transport. Users can visualize their commute routes, how much they spent on them and how long their trip took. MyTripDiary gives an estimate of the cost of each trip based on factors such as start location, destination, mode of transport (including Private Car, Taxi, Bike, Public Transport and Walking), fuel prices, parking availability, and transport fares.

1.2 Document Conventions

Priority of Requirements: The priority of higher-level requirements is assumed to be inherited by detailed requirements unless explicitly stated otherwise. The priority of all higher-level requirements is assumed to be equal unless explicitly stated otherwise.

Level 1 Heading: Font Family: Times, Font Size: 18, Font Weight: Bold
Level 2 Heading: Font Family: Times, Font Size: 14, Font Weight: Bold
Level 3 Heading: Font Family: Times, Font Size: 12
Content: Font Family: Arial, Font Size: 11

Throughout this document, 'the app' or simply 'app' refers to MyTripDiary unless explicitly specified otherwise.

Other conventions and definitions of terms used in this document can be found in **Append A: Data Dictionary**.

1.3 Intended Audience and Reading Suggestions

This document is intended for users of the app, the software developers of the app, the documentation writers, the project managers, the marketing staff, and testers.

This document details the description, use cases, functional and non-functional requirements, interfaces, constraints of the app. It is intended to be read in sequence by all stakeholders involved in this app.

1.4 Product Scope

MyTripDiary is an app that aims to enable people to organize and track their daily commutes. With MyTripDiary, daily commutes become faster, cheaper, more reliable, and more personalized. The simple to use interface, allows its users to take control of how they reach their destination making commuting more than just going from point A to point B.

1.5 References

I. Source Code (GitHub): https://github.com/ardnep/SC2006-Project-No-Idea
II. React Native: https://reactnative.dev/
III. Firebase: https://firebase.google.com/
IV. SQLite: https://sqlite.org/index.html

2. Overall Description

2.1 Product Perspective

MyTripDiary is a new app and is not part of a larger product family or product line.

2.2 Product Functions

MyTripDiary has the following main functions:

I. The app allows users to save, edit and delete trips
II. The app shows the most optimal route to take from a source location to a destination location
III. The app gives an estimate of how much a trip will cost
IV. The app allows the user to visualize their daily commutes showing them information including but not limited to how much the user spent on public transport in a month, on average, how long they spend on taxis, etc.
V. The app displays a history of each individual trip that the user has completed
VI. The app allows a user to star a trip. A starred trip is pinned to the top of the list of saved trips
VII. The app gives its user the flexibility to edit the price of a previously executed trip enabling them to adjust for any discrepancies existent in the price of the trip estimated by the app

2.3 User Classes and Characteristics

2.4 Operating Environment

The app operates in Android Version 4.4 and above and iOS Version 10 and above.

The development environment for the app is as follows:

Development Environment
Front-end: React Native for Mobile Application

Back-end: Firebase, SQLite

2.5 Design and Implementation Constraints

I. The app uses SQLite as a database
II. The app relies on Firebase for user login and registration. Hence, it is subject to Firebase services being available. In cases where Firebase services as unreachable or down, the app may not function as expected. The solution to this is to wait for the Firebase services to be available.

2.6 User Documentation

A demonstration of how the app works along with a guide to install and run the app is provided in the README.md file on the GitHub repository which contains the source code for this app (see Section 1.5 References).

2.7 Assumptions and Dependencies

I. The app relies on external APIs to get information such as the route of a trip and the estimated cost of the trip.
II. It is assumed that the information obtained from these APIs are accurate
III. In cases where the API services become unavailable, the app may not meet some requirements specified in this document.

3. External Interface Requirements

3.1 User Interfaces

Please refer to the deliverables/lab1/ui_mockups folder for User Interface Mockups.

3.2 Hardware Interfaces

TBD

3.3 Software Interfaces

TBD

3.4 Communications Interfaces

TBD

4. System Features

4.1 Add Trip

Use Case ID:	AT01		
Use Case Name:	Add Trip		
Created By:	Tan Ming Rui, Ezra	Last Updated By:	Tan Ming Rui, Ezra
Date Created:	31/01/23	Date Last Updated:	31/01/23

Actor:	User
Description:	Adds a trip, which is defined by the trip's start point and end point. Each trip also has a name.
Preconditions:	1. The user must be logged in.
Postconditions:	1. The user is able to Execute trip
Priority:	High

Frequency of Use:	Low
Flow of Events:	1. User fills in the name, start point, and end point of the trip they wish to create. 2. User clicks “Save”.
Alternative Flows:	
Exceptions:	EX-S2: User clicks “Cancel” 1. System returns to homepage
Includes:	GP01
Special Requirements:	
Assumptions:	1. The connection of the app and the Database is established
Notes and Issues:	

Functional Requirements:

1. The user must be able to add a trip.
   1.1. The system must display 3 text fields with autocomplete for the user to enter trip information.
      1.1.1. The system must display one text field with autocomplete for Start Point.
      1.1.2. The system must display one text field with autocomplete for End Point.
      1.1.3. The system must display one text field with autocomplete for Trip Name.
   1.2. The system must display an “Add Trip” button.
   1.3. The system must display a “Cancel” button.
   1.4. When the user clicks “Add Trip”, the system must save the trip information to the database.

4.2 Register

Use Case ID:	RR01		
Use Case Name:	Register		
Created By:	Tan Ming Rui, Ezra	Last Updated By:	Tan Ming Rui, Ezra
Date Created:	31/01/23	Date Last Updated:	31/01/23

Actor:	User

Description:	The user registers an account with their email address and password.
Preconditions:	1. The User's device must be connected to WiFi/Cellular Data. 2. The email address given is not registered in the System.
Postconditions:	1. The registered user account is stored in the database. 2. The User can login to the System using the registered account.
Priority:	High
Frequency of Use:	Once per User
Flow of Events:	1. The System requests the User to input the following information fields: • Email address • Password • Confirm Password 2. The System validates the required fields. 3. User clicks on the Register button 4. The System sends an email with a confirmation link to the registered email address. 5. If the User clicks on the confirmation link, the user account is successfully created. 6. Return to User Login menu after confirmation of account creation.
Alternative Flows:	AF-S4: Text fields are not filled in appropriately 1. The System will prompt the User to fill in the required fields. 2. The System returns to Step 1. AF-S2: Email already exists in the database 1. The System will prompt an error message "Email address is already in use" AF-S2: Username already exists in the database 1. The System will prompt an error message "Username is already in use"
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Functional Requirements:

1. The user must be able to register for a new account via our registration system.
   1.1. The system must display 3 text fields for the user to enter registration information.

1.1.1. The system must display one text field for Email Address
1.1.2. The system must display one text field for Password
1.1.3. The system must display one text field for Confirm Password
1.2. The system must display a “Register” button.
1.3. When the user clicks “Register”, the system must validate that all required fields are not left empty.
1.4. When the user clicks “Register”, the system must validate that the Email Address entered is a valid email
1.5. When the user clicks “Register”, the system must validate that the password entered must meet the following requirements:
1.5.1. A minimum of 8 characters in length.
1.5.2. Contain at least one (1) character from three (3) of the following categories: Uppercase letter (A-Z) Lowercase letter (a-z) Digit (0-9) Special character (~`!@#$%^&*()+=_-{}[]\|:;”"?/<>,.).

4.3 Login

Use Case ID:	LL01		
Use Case Name:	Login		
Created By:	Tan Ming Rui, Ezra	Last Updated By:	Tan Ming Rui, Ezra
Date Created:	31/01/23	Date Last Updated:	31/01/23

Actor:	User
Description:	Login to System through verification of User's account email and password.
Preconditions:	1. The User's device must be connected to WiFi/Cellular Data. 2. The System currently does not have a User logged in. 3. User has an existing and verified account associated with the email stored in the database. 4. The System must be able to communicate with the Database. 5. The hosted Database must be online.
Postconditions:	-
Priority:	High
Frequency of Use:	Low
Flow of Events:	1. The User inputs the email address and password 2. The User clicks the "Login" button. 3. The System verifies the login credentials. 4. If the login credentials are verified, the User successfully logs in. 5. The User can proceed to use the System.
Alternative Flows:	AF-S4: If the login credentials are invalid 1. The System informs the User the login has failed. 2. The System returns to Step 1.
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	

Notes and Issues:	

Functional Requirements:

1. The user must be able to log into the system.
   1.1. The system must display 2 text fields for the user to enter his login information.
      1.1.1. The system must display one text field for Email Address.
      1.1.2. The system must display one text field for Password.
   1.2. The system must display a "Login" button.
   1.3. When the user clicks "Login", the system must validate that all fields are not left empty.
   1.4. When the user clicks "Login", the system must verify the information obtained from the text fields.
      1.4.1. The email must be found in the database of the system.
      1.4.2. The password entered must match the password of the user.
   1.5. When the user clicks "Login" and information is verified to be a valid account, the system must log the user into the home screen of the system.

4.4 Delete Trip

Use Case ID:	DT01		
Use Case Name:	Delete Trip		
Created By:	Nepal Aaradh	Last Updated By:	Nepal Aaradh
Date Created:	01/02/2023	Date Last Updated:	01/02/2023

Actor:	User
Description:	Delete a trip that has been saved
Preconditions:	1. The user must be logged in 2. The trip being requested to be deleted must exit
Postconditions:	The trip is no longer existent
Priority:	
Frequency of Use:	
Flow of Events:	1. The user selects a trip to delete from a list of previously saved trips 2. The user confirms that the deletion should take place
Alternative Flows:	1. The user does not confirm the deletion and instead decides to cancel the delete request
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	1. The connection of the app and the Database is established
Notes and Issues:	

<u>Functional Requirements:</u>

1. The user must be able to delete a previously saved trip from the system

1.1. The system must display a list of previously saved trips
1.2. The user must be given the option to select one or multiple trips from this list to delete
1.3. The system must display a confirmation message asking the user to confirm if they want to go forward with deleting the selected trips
    1.3.1. The message must clearly state the implications of deleting a previously saved trip
1.4. The user must be given a choice to either confirm the deletion or cancel it
1.5. After the confirmation from the user, the deletion must be reflected in the system

4.5 Edit Trip

Use Case ID:	ET01		
Use Case Name:	Edit Trip		
Created By:	Nepal Aaradh	Last Updated By:	Nepal Aaradh
Date Created:	01/02/2023	Date Last Updated:	01/02/2023

Actor:	User
Description:	Edit the name of a previously saved trip
Preconditions:	1. The user must be logged in 2. The trip being requested to be edited must exit
Postconditions:	The trip name has been updated
Priority:	
Frequency of Use:	
Flow of Events:	1. The user selects a trip to edit from a list of previously saved trips 2. The user enters the new name for the trip with which to replace the current trip name 3. The user confirms the change
Alternative Flows:	
Exceptions:	EX1: If user cancels while editing 1. System returns to “Manage trips” screen
Includes:	
Special Requirements:	
Assumptions:	1. The connection of the app and the Database is established
Notes and Issues:	

Functional Requirements:

1. The user must be able to edit the name of a previously saved trip

1.1. The system must display a list of previously saved trips
1.2. The user must be given the option to select one trip from this list to edit
1.3. The system must display a text field for the user to enter the new name of the trip
    1.3.1. The text field must allow the user to input 64 characters
1.4. The system must give the user a choice to confirm the change or cancel editing the trip
    1.4.1. The system must not allow the user to confirm the change if the text field is empty or if the new name entered is the same as the current name of the trip
1.5. After the confirmation from the user, the change must be reflected in the system

4.6 Star Trip

Use Case ID:	ST01		
Use Case Name:	Star Trip		
Created By:	Nepal Aaradh	Last Updated By:	Nepal Aaradh
Date Created:	01/02/2023	Date Last Updated:	01/02/2023

Actor:	User
Description:	'Star' a trip (indicate a trip as a frequently used trip)
Preconditions:	1. The user must be logged in 2. The trip being requested to be starred must exit
Postconditions:	The trip is pinned to top of the list of previously saved trips making it easily accessible to the user
Priority:	
Frequency of Use:	
Flow of Events:	1. The user selects a trip to star from a list of previously saved trips
Alternative Flows:	
Exceptions:	
Includes:	
Special Requirements:	
Assumptions:	1. The connection of the app and the Database is established
Notes and Issues:	

Functional Requirements:

1. The user must be able to 'Star' a previously saved trip on the system
   1.1. The system must display a list of previously saved trips
   1.2. The user must be given the option to select one trip from this list to star
   1.3. The system must move the starred trip to the top of the list of previously saved trips
   1.4. The system must show an indication (either via a Star icon or a Pin icon beside the trip) that the trip has been starred.

4.7 Execute Trip

Use Case ID:	ET02		
Use Case Name:	Execute Trip		
Created By:	Goel Armaan	Last Updated By:	Goel Armaan
Date Created:	31/01/23	Date Last Updated:	31/01/23

Actor:	User
Description:	Records an instance of a saved trip with the user's selected mode of transport.
Preconditions:	1. User is logged in 2. User has the trip saved
Postconditions:	1. An instance of the saved trip is added to the user’s trip history
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User clicks a saved trip 2. User chooses their preferred mode of transport 3. User reviews the estimated price 4. User clicks the execute trip button 5. An instance of the saved trip is recorded in the system
Alternative Flows:	
Exceptions:	EX1: User cancels execution 1. User can choose not to execute a trip, at which point the use case exits at either step 2 or step 3 EX2: Trip cannot be executed 1. If a chosen mode of transport is unavailable, the user cannot move on to review estimated prices. 2. The user is informed of this limitation and is asked to choose another mode of transportation.
Includes:	GR01, GP01
Special Requirements:	
Assumptions:	The user executes a trip right before they embark on it

Notes and Issues:	

Functional Requirements:

1. The user must be able to run an instance of a saved trip.
    1.1. The user must be able to select a mode of transport
        1.1.1. There should be a button for Private Car/Motorbike
        1.1.2. There should be a button for Public Transport
        1.1.3. There should be a button for Private Hire Taxi
        1.1.4. There should be a button for Bicycle
        1.1.5. There should be a button for Walking
    1.2. The system must display the route of the trip with the selected mode of transport
        1.2.1. The system must retrieve the route from Use Case GR1 (Get Route)
        1.2.2. The system must display the interactive map retrieved from the Google Maps API
        1.2.3. The system must display the route on top of the interactive map
        1.2.4. The user must be able to zoom in and out using the pan and pinch gestures on screen
    1.3. The system must display the price for the selected mode of transport
        1.3.1. The system must retrieve the price from Use Case GP1 (Get Price)
        1.3.2. The system must display the estimated price beside the selected mode of transport
        1.3.3. The price must be displayed in S$
        1.3.4. The price must be rounded to 2 decimal places
    1.4. The system must save an instance of the trip to the database
        1.4.1. The system must save the saved trip ID
        1.4.2. The system must save the mode of transport
        1.4.3. The system must save the time of execution
        1.4.4. The system must save the estimated price

4.8 Get Route

Use Case ID:	GR01		
Use Case Name:	Get Route		
Created By:	Goel Armaan	Last Updated By:	Goel Armaan
Date Created:	31/01/23	Date Last Updated:	31/01/23

Actor:	Google Maps API
Description:	Retrieves the route between a trip's starting point and destination point for the chosen mode of transport
Preconditions:	1. User is logged in 2A. User is adding a new trip or 2B. User is executing a saved trip 3. User device has a working WiFi or Cellular connection
Postconditions:	1. The correct route from the trip's starting point to the trip's destination point with the chosen mode of transport is displayed
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User has entered the starting and destination point for a new trip 2. The API is queried and the route for a private car is displayed by default.
Alternative Flows:	AF-S1: User is executing a saved trip 1. If the user is executing a saved trip, they do not have to enter the starting and destination point as required in step 1. AF-S2: User chooses another mode of transport 1. If the user chooses another mode of transport, the API is queried again and a route with the new mode of transport is displayed.
Exceptions:	EX1: No route can be found 1. If there is no route between the start and destination point for the selected mode of transport, a route cannot be displayed. 2. The user is informed of this limitation and is asked to try another mode of transportation EX2: API is unavailable 1. If the Google Maps API is unavailable or cannot be queried, a route cannot be displayed.

	2. The user is informed of this limitation and is asked to try again later.
Includes:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Functional Requirements:

1. The system must get retrieve the route for a given starting point and destination point
   1.1. The system must query the Google Maps API to retrieve the route for a given starting point and destination point

4.9 Get Price

Use Case ID:	GP01		
Use Case Name:	Get Price		
Created By:	Goel Armaan	Last Updated By:	Goel Armaan
Date Created:	31/01/23	Date Last Updated:	31/01/23

Actor:	Car Park API, Public Transport API, Taxi API
Description:	Calculates an estimated price of a route with the chosen mode of transportation
Preconditions:	1. User is logged in 2A. User is adding a new trip or 2B. User is executing a saved trip 3. The trip's route is valid 4. User device has a working WiFi or Cellular connection
Postconditions:	1. A well-estimated price of a route with the chosen mode of transportation is displayed
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User selects a valid mode of transportation for a trip 2. The respective API is queried 3. The price for the trip's route with the chosen mode of transportation is displayed
Alternative Flows:	AF-S2: 1. If the mode of transportation is biking or walking, no API is queried. The price is 'free'.
Exceptions:	EX1: API is unavailable 1. If the API used in calculating the price for a chosen mode of transportation is unavailable, then the estimated price cannot be calculated 2. Instead, a message is shown to the user stating the estimated pricing for the chosen mode of transportation is currently unavailable
Includes:	GR01
Special Requirements:	

Assumptions:	
Notes and Issues:	

Functional Requirements:

1. The system must retrieve the price for a given route and mode of transportation
    1.1. The system must make an API call to the appropriate API to get the price of a route
        1.1.1. The system must contact the Public Transport API when using Public Transport
        1.1.2. The system must contact the Private Hire API when using private hire taxi

4.10 View Stats

Use Case ID:	VS01		
Use Case Name:	View Stats		
Created By:	Xu Yinfeng	Last Updated By:	Xu Yinfeng
Date Created:	01-02-2023	Date Last Updated:	01-02-2023

Actor:	User, Database
Description:	System get analytics data from database and display on the analytics dashboard
Preconditions:	1. User is connected to WiFi/Cellular Data 2. User has logged into their user account 3. User has completed at least 1 trip
Postconditions:	-
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User completes a trip, trip recorded in trip history 2. System functions calculate/store trip information statistics 3. System get trip statistics and display in the analytics dashboard
Alternative Flows:	AF1: 1. If user has not completed any trips, display “No Stats Available”
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Functional Requirements:

1. The system must query the database for all trip statistics
   1.1. The system must have calculated all the statistics based on trip history
   1.2. The system must have saved all calculated statistics into database

2. The system must render the statistics in the form of percentages/charts
   2.1. The rendered analytics dashboard must be updated on hourly basis

4.11 View Trip History

Use Case ID:	VH01		
Use Case Name:	View Trip History		
Created By:	Xu Yinfeng	Last Updated By:	Xu Yinfeng
Date Created:	01-02-2023	Date Last Updated:	01-02-2023

Actor:	Database, User
Description:	Get user trip history, display user trip history
Preconditions:	1. User is connected to WiFi/Cellular Data 2. User has logged into their user account
Postconditions:	-
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User login to their user account 2. User clicked on "Trip History" side bar 3. System retrieves user trip history from database 4. System display trip history by user-chosen order
Alternative Flows:	AF1: 1. If user has not completed any trips, display “No trip history”
Exceptions:	-
Includes:	EP01
Special Requirements:	-
Assumptions:	User has logged into their account
Notes and Issues:	-

Functional Requirements:

1. The system must query historical trip information from the database
   1.1. The data retrieved need to be sorted by time
2. The system need to render the historical trip information on Trip History page

2.1. Trip History page render must be refreshed as per access

4.12 Edit Past Trip Price

Use Case ID:	EP01		
Use Case Name:	Edit Past Trip Price		
Created By:	Xu Yinfeng	Last Updated By:	Xu Yinfeng
Date Created:	01-02-2023	Date Last Updated:	01-02-2023

Actor:	User, Database
Description:	User enter/edit trip price information upon completion of a trip
Preconditions:	1. User is connected to WiFi/Cellular Data 2. User has logged into their user account 3. User has completed at least 1 trip
Postconditions:	-
Priority:	High
Frequency of Use:	High
Flow of Events:	1. User completes a trip, trip recorded in trip history 2. User goes into trip history and click “edit” button 3. User changes the value of “trip price” for trip 4. User click “confirm edit” 5. System send post request to update database trip price
Alternative Flows:	AF1: 1. If user has not completed any trips, display “No trip available”
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Functional Requirements:

1. The system must store edited information as session storage
2. The system must query the database according to the index of trip
3. The system must update the index using session storage data
4. The system must dynamically render the updated price on the trip information page after confirmation of edit by the user

4.13 View Saved Trips

Use Case ID:	VT01		
Use Case Name:	View Saved Trips		
Created By:	Nepal Aaradh	Last Updated By:	Nepal Aaradh
Date Created:	02/02/2023	Date Last Updated:	02/02/2023

Actor:	User
Description:	Displays the previously saved trips to the user
Preconditions:	1. The user must be logged in 2. The user must have saved from trips previously
Postconditions:	1. The user can view the previously saved trips
Priority:	
Frequency of Use:	
Flow of Events:	
Alternative Flows:	
Exceptions:	
Includes:	DT01, ST01, ET01
Special Requirements:	
Assumptions:	1. The connection of the app and the Database is established
Notes and Issues:	

Functional Requirements:

1. The system allows the users to view previously saved trips
   1.1. The system must display a list of all the previously saved trips by the user
   1.2. The system must allow the user to click on one of the listed trips to Execute Trip (Use Case ID: ET01)

4.14 Get Carpark Availability

Use Case ID:	GA01		
Use Case Name:	Get Carpark Availability		
Created By:	Xu Yinfeng	Last Updated By:	Xu Yinfeng
Date Created:	28/01/2023	Date Last Updated:	28/01/2023

Actor:	Carpark API, System, User
Description:	System gets carpark availability information based on user destination
Preconditions:	1. Users need to be connected to WiFi/Cellular data 2. Users need to enter a valid destination 3. Users need to tick “yes” for travelling by car
Postconditions:	-
Priority:	High
Frequency of Use:	High
Flow of Events:	1. Users select destination and preference ranking 2. Users tick “yes” for travelling by car 3. Upon route search, System should send get request to Carpark API to get carpark availability information
Alternative Flows:	-
Exceptions:	EX1: Invalid destination 1. Display error message and prompt user to reenter destination EX2: Carpark API Query failure 1. System prompts error message 2. “Retry” button enabled for retrying API query
Includes:	-
Special Requirements:	-
Assumptions:	1. Carpark API is online and querable
Notes and Issues:	-

Functional Requirements:

1. The system must retrieve car park availability information based on user destination
   1.1. The system must make API call to get car park availability in proximity of the user's entered destination
2. The system must render the imagery location of car parks nearby and their available parking slots
   2.1. The Imagery location point must be clickable to display car park name and postal code

4.15 Get Parking Price

Use Case ID:	GP01		
Use Case Name:	Get Parking Price		
Created By:	Xu Yinfeng	Last Updated By:	Xu Yinfeng
Date Created:	28/01/2023	Date Last Updated:	28/01/2023

Actor:	Carpark API, System, User
Description:	Get parking rate information of carparks and calculate parking fees based on parking duration and time of the day
Preconditions:	1. Users need to be connected to Wi-Fi/Cellular data 2. Users need to enter a valid destination 3. Users need to tick “yes” for travelling by car
Postconditions:	-
Priority:	High
Frequency of Use:	High
Flow of Events:	1. Users select destination and preference ranking 2. Users tick “yes” for travelling by car 3. Users enter duration of stay and time of arrival 4. System gets carpark rate from Carpark API 5. System calculates total parking fee based on carpark rate, time of day and length of stay
Alternative Flows:	-
Exceptions:	EX1: Invalid destination 1. Display error message and prompt user to renter destination EX2: Carpark API Query failure 1. System prompts error message 2. “Retry” button enabled for retrying API query
Includes:	-
Special Requirements:	-
Assumptions:	1. Carpark API is online and query-able
Notes and Issues:	-

Functional Requirements:

1. The system must retrieve car park price information based on user destination
   1.1. The system must make API call to get car park parking price in proximity of the user's entered destination
2. The system must render the imagery location of car parks nearby and their price
   2.1. Details of parking rate must be viewable upon clicking into the imagery location

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 Each page must load within 3 seconds.

5.1.2 Car park availability in an area must be loaded within 3 seconds.

5.1.3 The system must be able to calculate the fare cost of a public transport route within 3 seconds.

5.1.4 The system must be able to synchronize data to the online database within 3 seconds.

5.2 Usability Requirements

5.2.1 **The system must display error messages that allow the user to k**now what went wrong when an error occurs. 75% of the users must be able to understand the error and not repeat the same flow of events.

5.2.2 80% of first-time users must be able to add their first trip to the app within 2 minutes.

5.3 Security Requirements

5.3.1 The system must send authentication data and CSRF tokens with encryption rather than plain-text form.

5.3.2 The system must be able to send a verification link to the user's email for confirmation.

5.3.3 The system must use encryption when saving sensitive details such as a user's password.

5.4 Reliability Requirements

5.4.1 The system must follow data visualisation standards when displaying user analytics. (Example: pie chart adding up to 100% or have the correct 'slices' corresponding to the ratios of the values it presents)

5.4.2 The system must be able to show estimated prices 95% of the time.

5.5 Business Rules

5.5.1 Users are required to login and authenticate themselves with their own account before accessing the website services.

5.5.2 Users must be verified before they are allowed to change their login credentials

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Data Dictionary

Term	Definition
System	The app MyTripDiary
Front-end	The User Interface (UI) elements of the mobile application
Bank-end	The system that allows for storing and manipulating data. Usually hidden from the end-user.
API	Application Programming Interface
User	The entity that uses the application.
Mode of Transport	The medium via which one is traveling from point A to point B. Includes private cars, taxis, private hired cars, buses, MRTs, walking, biking
Trip	A short distance travel from point A to point B.
Trip Stats/Journey Analytics	The analytics of the history of the trips a user has completed so far
Instance of a Trip	The record of when a user goes on a saved trip once. This record includes time of execution of the trip, mode of transport and the price of the trip.
Saved Trip	A Trip that is saved on the system. The information of the trip saved includes the start and end points of the trip and the name assigned to the trip.
Starring a Trip	Indicating that a trip is frequently used and hence should be on top of the list of saved trips for ease of access.

Executing a Trip	Starting an instance of a Trip

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

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