BD Travellers

Software Requirements Specifications

BD Travellers
(Software Requirements Specifications)

Submitted by,

Sharafat Ahmed
BSSE 0617
Institute of Information Technology,
University of Dhaka

Supervised by,

Dr. Kazi Muheymin-Us-Sakib
Director and Associate Professor
Institute of Information Technology
University of Dhaka

Submitted to,

Asif Imran
Lecturer
Institute of Information Technology,
University of Dhaka

Sheikh Muhammad Sarwar
Assistant Professor
Institute of Information Technology,
University of Dhaka

Date: 13th April, 2016

LETTER OF TRANSMITTAL

5th April, 2015

Asif Imran
Lecturer
Institute of Information Technology (IIT)
University of Dhaka

Sir
I have prepared the report on Software Requirements Specification of **“BD Travellers”** for your approval. This report details the requirements I gathered for the project.

The primary purpose of this report is to summarize my findings from the work that I completed as my Software Requirements Specification and Analysis course project. This report includes the details of each step I followed to collect the requirements.

Sincerely Yours
Sharafat Ahmed
BSSE 0617

Enclosure: SRS Report

Table of Contents

INTRODUCTION

Purpose

This document is the Software Requirements Specification (SRS) for the BD Travellers. It contains detailed functional, non-functional, and support requirements and establishes a requirements baseline for development of the system. The requirements contained in the SRS are independent, uniquely numbered, and organized by topic. The SRS serves as the official means of communicating user requirements to the developer and provides a common reference point for both the developer team and stakeholder community. The SRS will evolve over time as users and developers work together to validate, clarify and expand its contents.

Intended Audience

This SRS is intended for several audiences, including the customer, as well as the project managers, designers, developers, and testers.

- The customer will use this SRS to verify that the developer team has created a product that is acceptable to the customer.
- The project managers of the developer team will use this SRS to plan milestones and a delivery date, and ensure that the developing team is on track during development of the system.
- The designers will use this SRS as a basis for creating the system's design. The designers will continually refer back to this SRS to ensure that the system they are designing will fulfill the customer's needs.
- The developers will use this SRS as a basis for developing the system's functionality. The developers will link the requirements defined in this SRS to the software they create to ensure that they have created software that will fulfill all of the customer's documented requirements.
- The testers will use this SRS to derive test plans and test cases for each documented requirement. When portions of the software are complete, the testers will run their tests on that software to ensure that the software fulfills the requirements documented in this SRS. The testers will again run their

tests on the entire system when it is complete and ensure that all requirements documented in this SRS have been fulfilled.

INCEPTION

Introduction

Inception is the beginning phase of requirements engineering. It defines how does a software project get started and what is the scope and nature of the problem to be solved. The goal of the inception phase is to identify concurrence needs and conflict requirements among the stakeholders of a software project. To establish the groundwork I have worked with the following factors related to the inception phases:

- Identifying Stakeholders
- First question to the Stakeholders
- Recognizing multiple viewpoints
- Working towards collaboration

Identifying the Stakeholders:

Stakeholder refers to any person or group who can influence or can be influenced by the system directly or indirectly. Stakeholders include end-users who interact with the system and everyone else in an organization that may be influenced by its installation**.** Stakeholder identification is the process used to identify all stakeholders for a project. It is important to understand that not all stakeholders have the same influence or effect on a project, nor will they be influenced in the same manner. It should be done in a methodical and logical way to ensure that stakeholders are not easily omitted. The following questions help us to identify stakeholders:

- Who uses the system?
- Who is effected by the outputs of the project?
- Who evaluates/approves system?
- Who maintains the system?
- Who has knowledge (specialist) about the system?

 Whose work will influence my project? (During the project and also once the project is completed).

I identified the stakeholders for BD Travellers are:

- **Admin:** The admin will manage and handle the whole system after developed. The profile or group creation, information storing, manipulation, verification and correction, system updating etc. will be the responsibility of the admin.

- **Travellers:** Travellers are persons who interact with BD Travellers mostly. They give lots of information and also they seek lots of information from it.

My questions to the stakeholders

I set my questions for the stakeholders in a way so that they could give their opinion and requirements for the system. The questions are mentioned in the last part of the SRS. My questions also focus on the measurable benefits and successful implementation of the project.

Recognizing Multiple Viewpoints:

I have collected the following view points by discussing with Dr. Kazi Muheymin-Us-Sakib (Associate Professor of IIT, DU.) and several travellers who have travelled many places across Bangladesh.

1. Admin:
  a) Maintain a database for storing information.
  b) Database must be secured.
  c) Report generating system

2. Traveller:
  a) Easy Access.
  b) Easy maintenance such as inserting, updating and deleting.

c) Easy to operate.

d) Easy expenditure management system.

Working towards Collaboration

I have asked my stakeholders for their requirements and found out that each of them has their own requirements. Some of the requirements are common as well as conflicting. So I need to follow the steps given below to merge the requirements:

- ➢ Find out the common and conflicting requirements.
- ➢ Divide the requirements into different categories.
- ➢ Identify the special requirements that the stakeholders have.
- ➢ Identify the all the requirements according to the stakeholder’s priority points and prioritize them through voting.
- ➢ Take final decision about the requirements.

Conclusion

The inception phase of this project helped us to identify the stakeholders as well as their different requirements. I have recognized multiple viewpoints of stakeholders by communicating with them. In the next chapter, I have discussed about finding requirements and scenario.

ELICITATION

Introduction

Elicitation is a task that helps the customer to define what is required. To complete the elicitation step I faced many problems like problems of scope, problems of volatility and problems of understanding. However, this is not an easy task. To help overcome these problems, I have worked with the Eliciting requirements activity in an organized and systematic manner.

Eliciting Requirements

Unlike inception where Q&A (Question and Answer) approach is used, elicitation makes use of a requirements elicitation format that combines the elements of problem solving, elaboration, negotiation, and specification. It requires the cooperation of a group of end-users and developers to elicit requirements .To elicit requirements I completed following four works.

1. Collaborative Requirements Gathering

2. Quality Function Deployment

3. Usage Scenarios

4. Elicitation work products

Collaborative Requirements Gathering

Many different approaches to collaborative requirements gathering have been proposed. Each makes use of a slightly different scenario .I completed following steps to do it.

- The meetings were conducted with the stakeholders I mentioned earlier. They were questioned about their requirements and expectations from BD Travellers.

- At last I selected my final requirement list from the meetings.

Quality Function Deployment

Quality Function Deployment (QFD) is a technique that translates the needs of the customer into technical requirements for software. It concentrates on maximizing customer satisfaction from the Software engineering process .With respect to my project the following requirements are identified by a QFD.

Normal Requirements

Normal requirements consist of objectives and goals that are stated during the meeting with the customers. Normal requirements of my project are:

- Android based system

- Allow access from any Android phone using internet connection.

- The registered user will go through the log in and log out process to use their account. They have access to almost every features of the system.

- The admin will input some static information about various places.

- Allow users to give ratings and reviews about some place in the map. They can also report any inappropriate or abusive review.

- Allow admin to delete any inappropriate or abusive review. He/she can also delete any user or any group.

- Allow travellers to manage expenditure and tour plan.

Expected Requirements

These requirements are implicit to the system and may be so fundamental that the customer does not explicitly state them. Their absence will be a cause for dissatisfaction.

- Maintain a database to store the information.

- A search option to find any place.

Exciting Requirements:

These requirements are for features that go beyond the customer's expectations and prove to be very satisfying when present

- Allow travellers to tour as a group.

- Allow travellers to send notification to his/her group mates when he/she is in danger.

Usage Scenarios

Bangladesh is a land of beauties with many beautiful places like The Sundarbans, Cox's Bazaar, or Saint Martin etc. Travellers from Bangladesh and abroad visit this places all the year around to observe the beauty or to relax. They face various problems for lack of information, lack of plan and even lack of tour management, like expenditure management etc. So a mobile based application can be developed named as "BD Travellers", which will provide information to the users about their desired places, help them to make plan for tours and also will keep track of expenditures made.

A user will have to be authenticated by his/her Google ID (Gmail) first. If he wants to login, a Gmail API will be shown. Once logged in, a user will not be logged out unless he does it manually.

A user can select his/her desired places in the application. (We will provide information about Saint Martin only, in this project.) There will be three parts - those are Information, Plan a trip and Update information.

In the Information part there will be transportation information, must do list and an interactive map. In the first two parts, there will be some static textual information. The information will be maintained solely by the administrator. The information may contain following things –

a. How to go there from Dhaka?
b. List of at most 10 things that should be done by the traveler.

A user can see different types of transport news and contact numbers (Bus, Launch, Train, Plane) in this app. There will be information about source to destination, time, vehicle type and ticket price.

A user can see his current position in the map which will be calculated by GPS. If a user clicks a point of the map, he can see last 5 reviews with rating about that place. Reviews are categorized as about hotels, beautiful places, food info, danger zones and others, which will be differentiated by colors.

There are two parts of plan a trip. One is create a group another is join a group. If a user wants to create a group, then he/she will have the option to give the tour a name. Otherwise, the group name will be "[Place name] tour" by default. He/she can generate an invitation code and share it with his friends via facebook, Bluetooth, messenger etc. The one who created the group will be termed as admin. He/she can add/remove members as well as admin in the group. Another option will be about joining an existing group. A user can join a group by providing the invitation code of a group. Admin must approve his/her join request.

The final part will be Update information. If a user clicks a point of the map, he will get an option about the category of that place. He must select a category. Then he will get an option about giving a text review and then he will also have to give a rating from 1 to 10. If anyone thinks a review is offensive or inappropriate, he/she can report it to the admin of the application “BD Travellers”. He can delete a review if he also thinks it is offensive or inappropriate.

After the “Select desired places” part, there will be an option of “Tour Management”, this has two parts. One is expenditure management another is a To do list.

Expenditure will be categorized in two parts. One is Personal expenditure and other is Group expenditure. Personal expenditure will be available for only the user who spent it. He can see the total money that he/she spent in the tour. Group expenditure will be available for everyone in the group. There will be a fund for group expenditure. Admin can collect money from group members for this fund. The amount of money given by a member in this fund will be available for every group member as a list. A person can enlist expenditure as Personal or as Group expenditure. He/she has to input expense amount, time and item (On which he spent). A group expenditure must be approved by the admin of the group. Every expenditure will be shown in a list. If admin pays member money, there will be a tick mark beside his/her expenditure and the amount will also be deducted from group fund.

A user can also plan a trip for his/her group. Anyone in the group can suggest adding plans. Only admin of the group can add those plans. When a plan is executed, admin can mark the plan as done.

When a user faces any emergency situation, he can send message to his group mates with his current location. His/her tour group mates will be in a “Emergency Help Group” by default.

The admin of this app can delete any account if a user constantly posts inappropriate or abusive reviews. He can also disband any group if the group remains inactive for a long time.

Elicitation work product

The output of the elicitation task can vary depending on size of the system or product to be built. My elicitation work product includes:

- Make a statement of my requirements for BD Travellers.
- Make a bounded statement of scope for my system.

- Make a list of customer, user and other stakeholder who participated in requirements elicitation.

- Set of usage scenarios.

- Description of the system’s technical environment.

Scenario Based Modeling

Use Case: A use case diagram is a graphic depiction of the interactions among the elements of a system. This diagram can portray the different types of users of a system and how they interact with the case.

Activity Diagram: Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. Here the overall flow of control is shown.

Swim lane diagram: Swim lane diagrams are the process flow model where the processes are grouped visually by placing them in lanes and these lanes are for the actors of the system.

In this section use case scenarios, activity diagram and swim lane diagram are described elaborately:

Level 0:

Use Case:

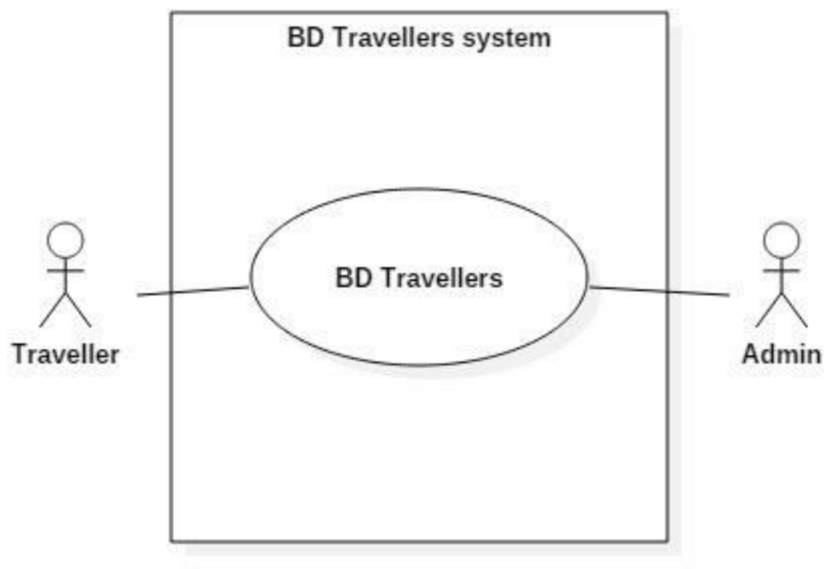


Figure 1: Use Case Level 1

Use Case Id	**UC_0**
Use Case Name	BD Travellers
Primary actor	Travellers, Admin
Preconditions	System exists
Action in reply	This level is very primitive. I will illustrate in Level 1.1
Exceptions	No exceptions.
Secondary Actors	No secondary actor.

Activity Diagram:

This level is very primitive. I will illustrate it in the level 1.1.

Swim Lane Diagram:

This level is very primitive. I will illustrate it in the level 1.1.

Level 1:

Use Case:

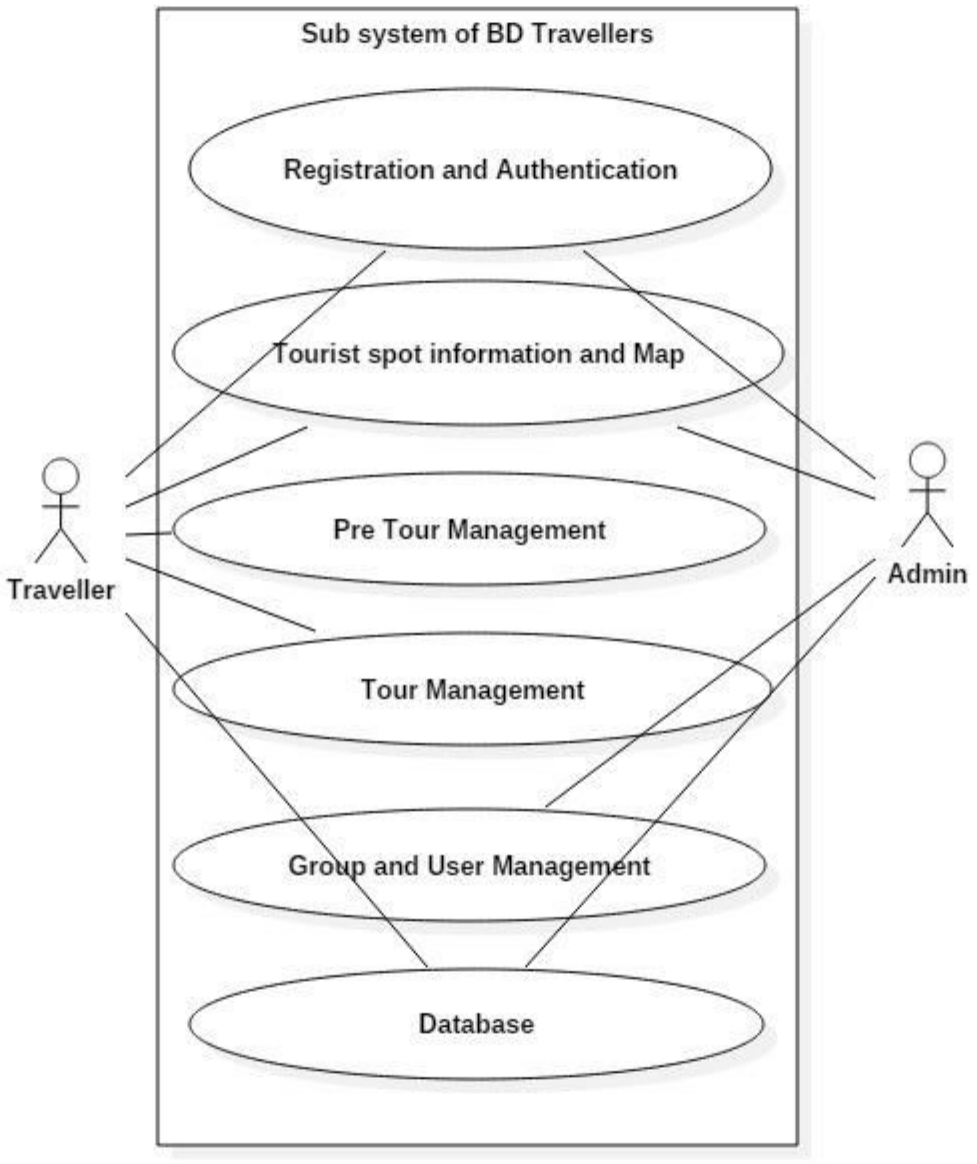

Sub system of BD Travellers
Registration and Authentication
Tourist spot information and Map
Pre Tour Management
Tour Management
Group and User Management
Database
Traveller
Admin

Figure 2: Use Case Level 1

Use Case Id	UC_1
Use Case Name	Sub Systems of BD Travellers
Primary actor	Traveller, Admin
Preconditions	System exists
Trigger	A user is logging in.
Action in reply	This level is very primitive. I will illustrate in Level 1.1
Exceptions	No exception
Secondary Actors	No secondary actor.

Activity Diagram:

This level is very primitive. I will illustrate it in the level 1.1.

Swim Lane Diagram:

This level is very primitive. I will illustrate it in the level 1.1.

Level 1.1:

Use Case:

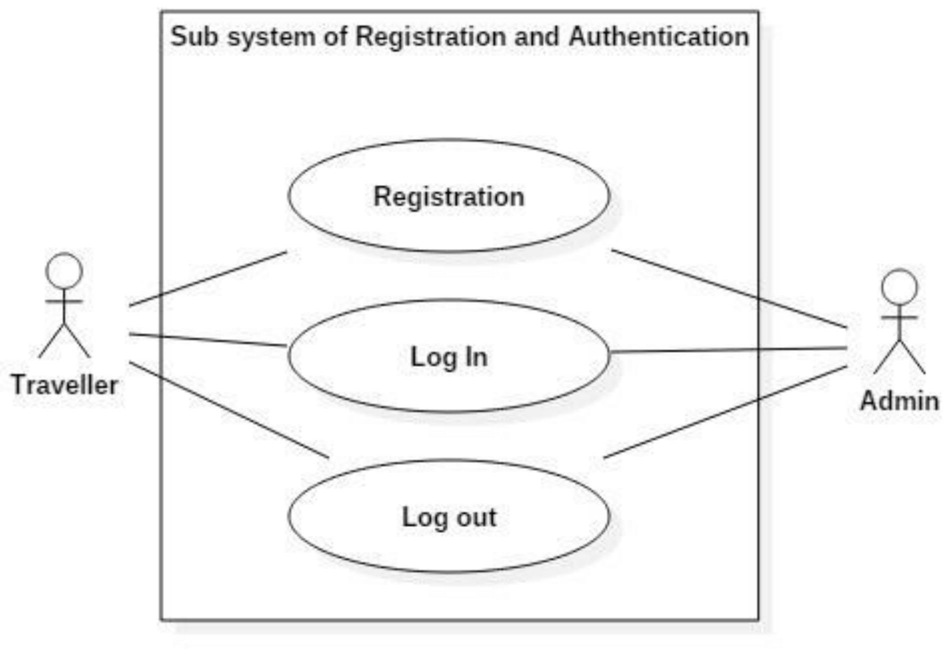


Figure 3: Use Case Level 1.1

Use Case Id	UC_1.1
Use Case Name	Registration and Authentication
Primary actor	Traveller, Admin
Goals in context	To identify users
Preconditions	System is programmed for authentications
Trigger	A user is logging in.
Action in reply	Travellers authenticate by Google ID. Admin authenticates by username and password.
Exceptions	If the username or password is wrong: try again.
Secondary Actors	No secondary actors.

Activity Diagram:

Activity diagram of level 1.1 is

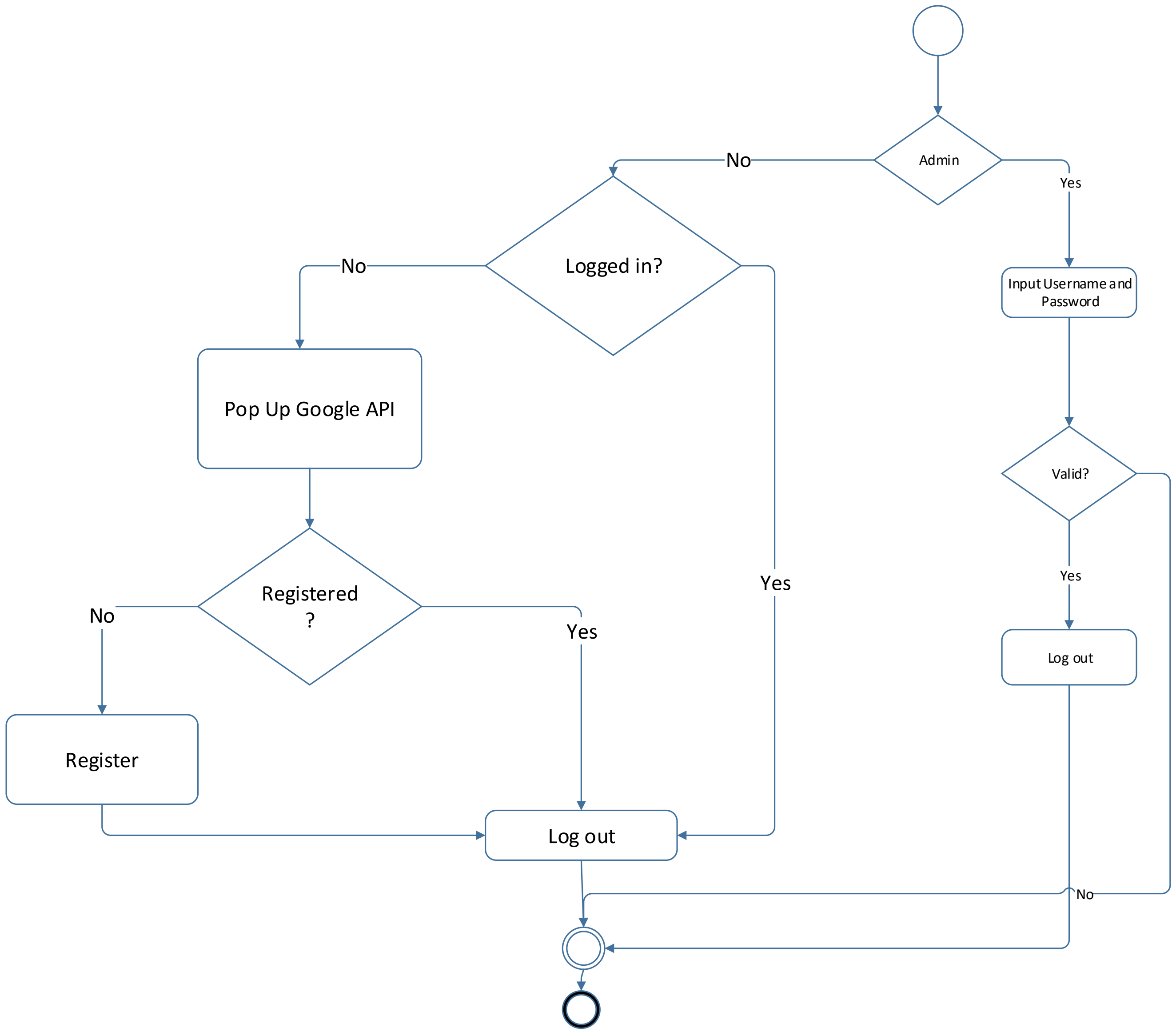


Figure 4: Activity Diagram of Authentication

Swim Lane Diagram:

Swim lane diagram of level 1.1 is

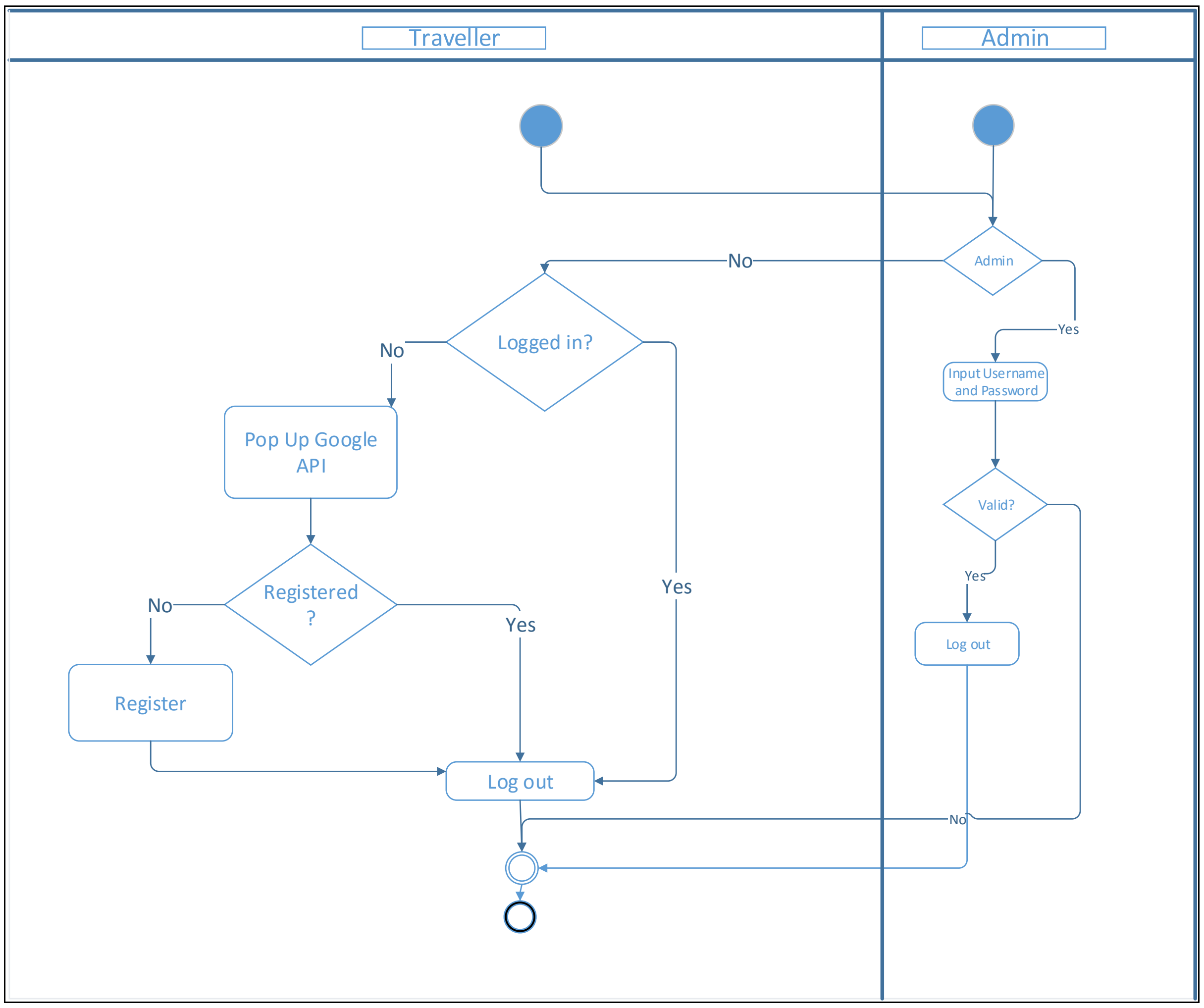


Figure 5: Swim Lane Diagram of Authentication

Level 1.2:

Use Case:

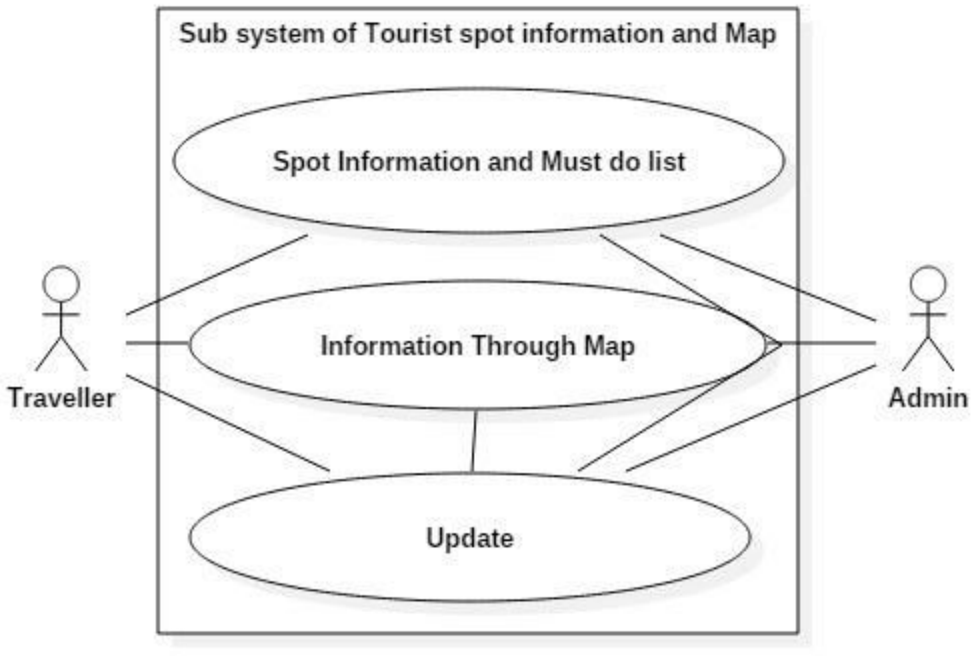

Sub system of Tourist spot information and Map
Spot Information and Must do list
Information Through Map
Update
Traveller
Admin

Figure 6: Use Case Level 1.2

Use Case Id	**UC_1.2**
Use Case Name	Tourist spot information and Map
Primary actor	Traveller, Admin
Goals in context	To give information
Preconditions	User is authenticated
Trigger	User prompts for information or update information.
Action in reply	User can view or update desired information.
Exceptions	No exceptions.
Secondary Actors	No secondary actor.

Activity Diagram:

Activity diagram of level 1.2 is

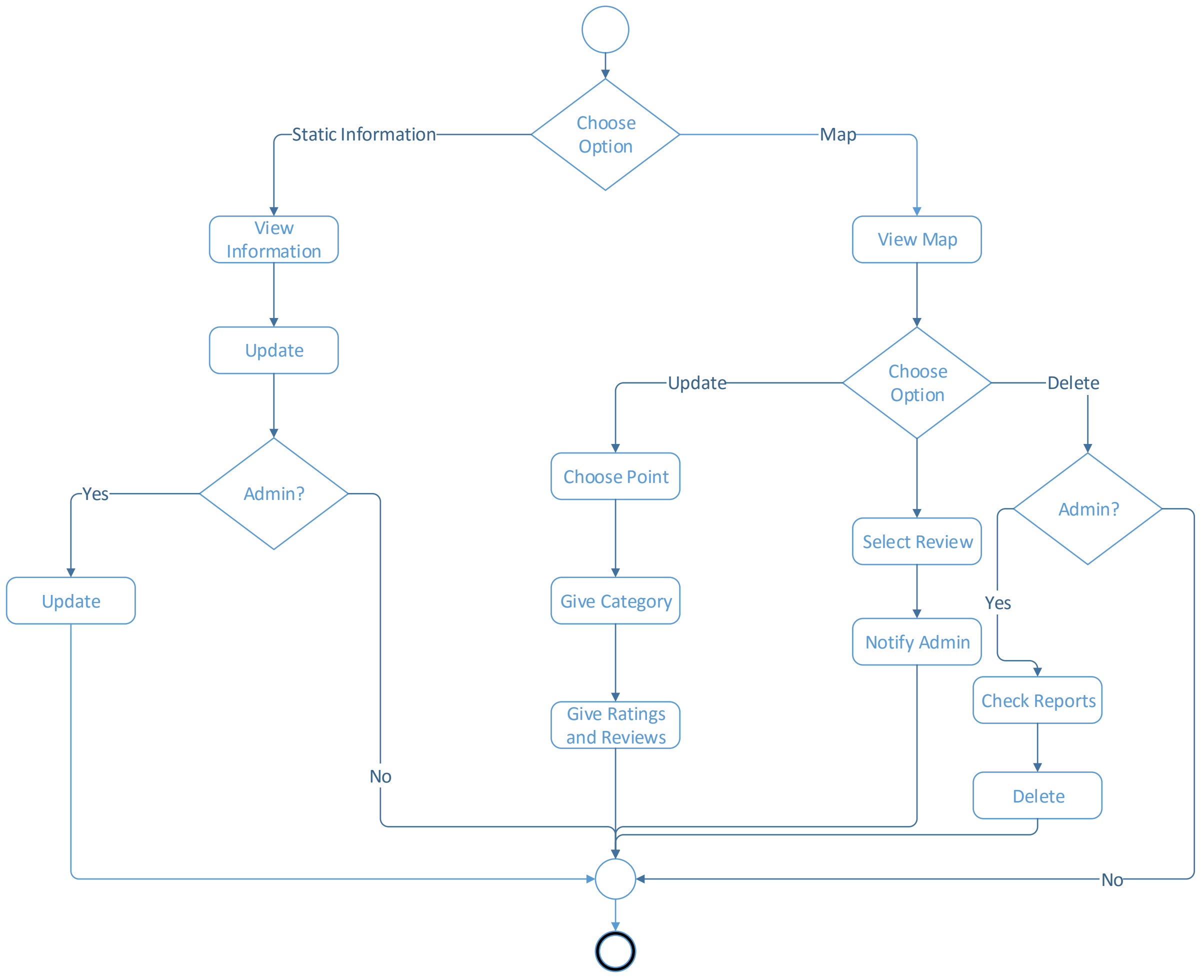


Figure 7: Activity Diagram of Tour Information

Swim Lane Diagram:

Swim lane diagram of level 1.2 is

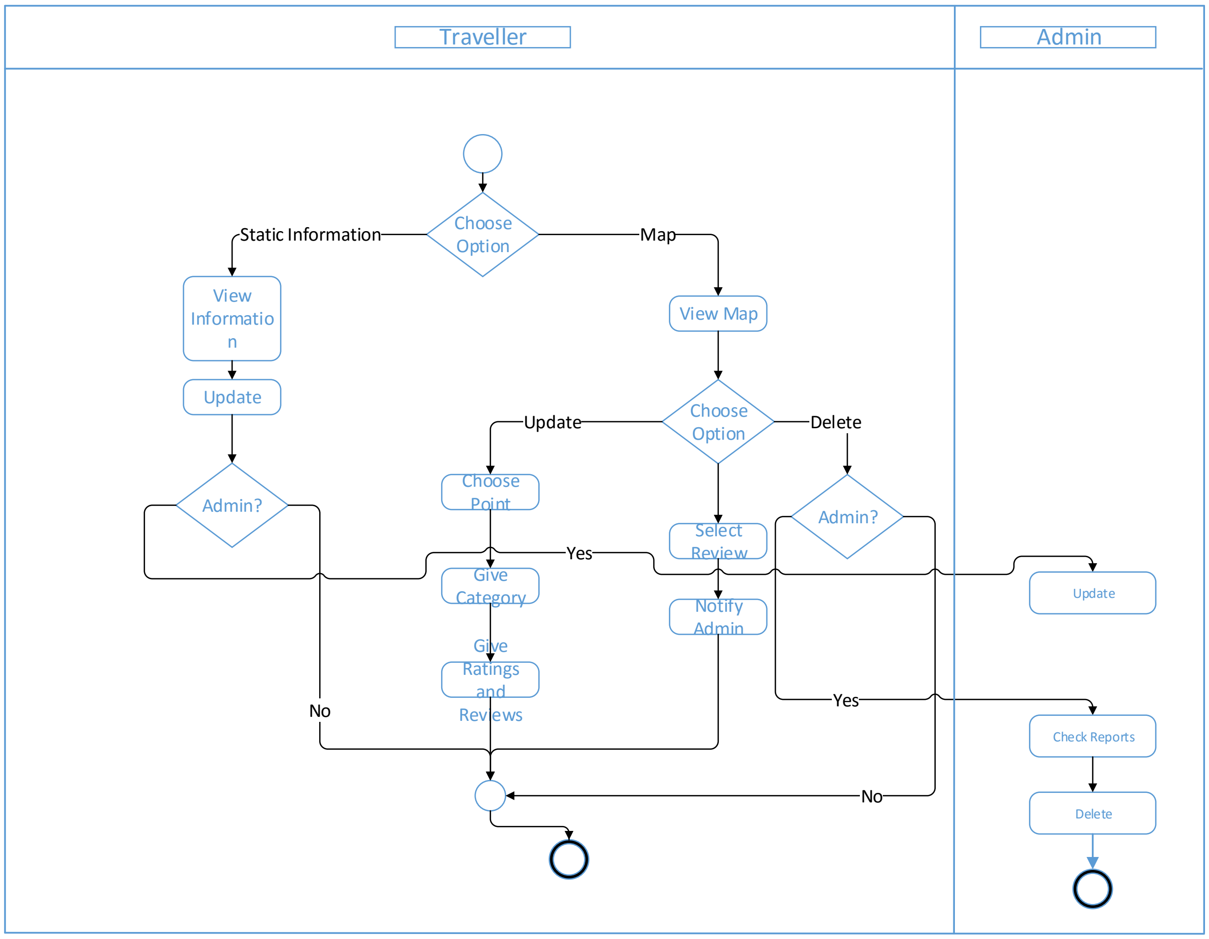


Figure 8: Swim Lane Diagram of Tour Information

Level 1.3:

Use Case:

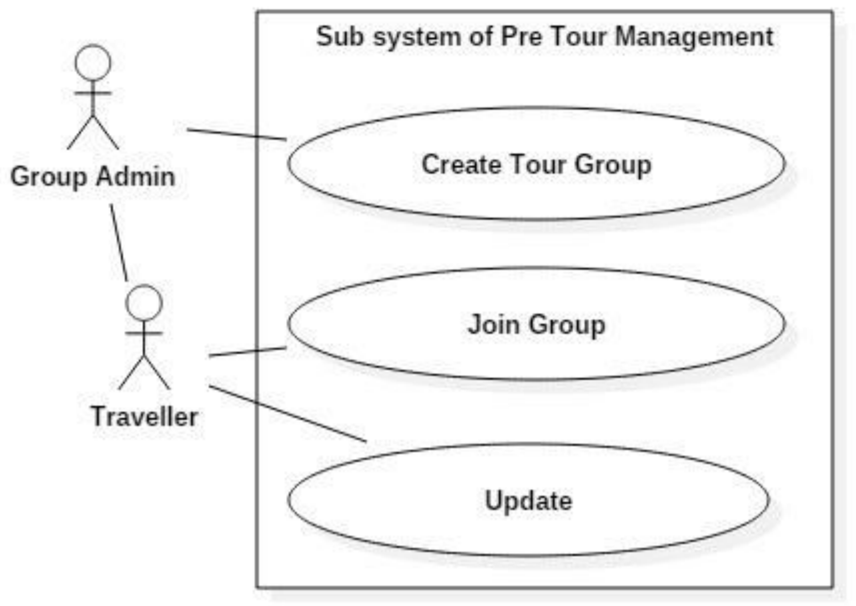


Figure 9: Use Case Level 1.3

Use Case Id	**UC_1.3**
Use Case Name	Pre Tour Management
Primary actor	Traveller
Goals in context	Form group
Preconditions	User is authenticated
Trigger	User prompts to create, share or join group
Action in reply	User can create, share or join group
Exceptions	No exceptions.
Secondary Actors	No secondary actor.

Activity Diagram:

Activity diagram of level 1.3 is

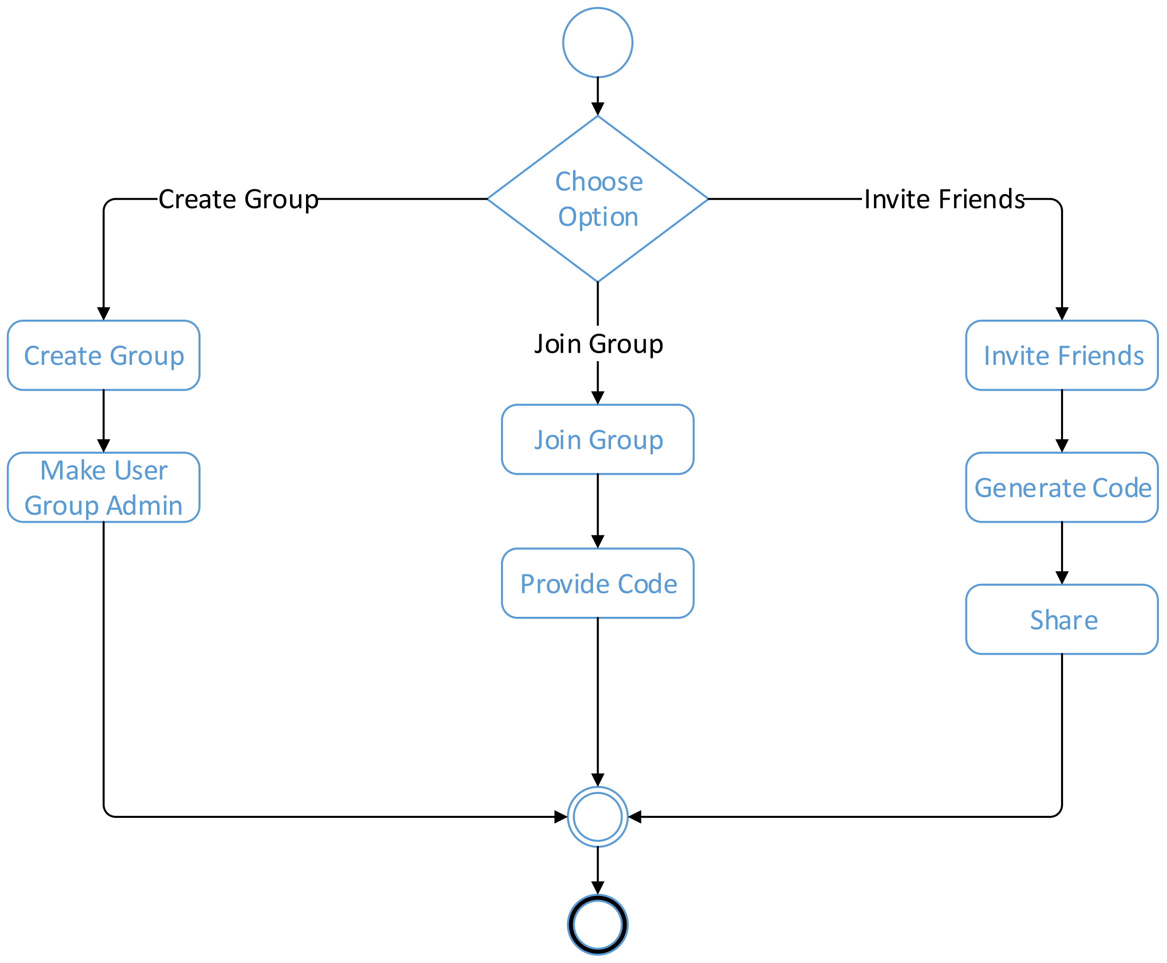


Figure 10: Activity Diagram of Pre Tour Management

Swim Lane Diagram:

Swim lane diagram of level 1.3 is

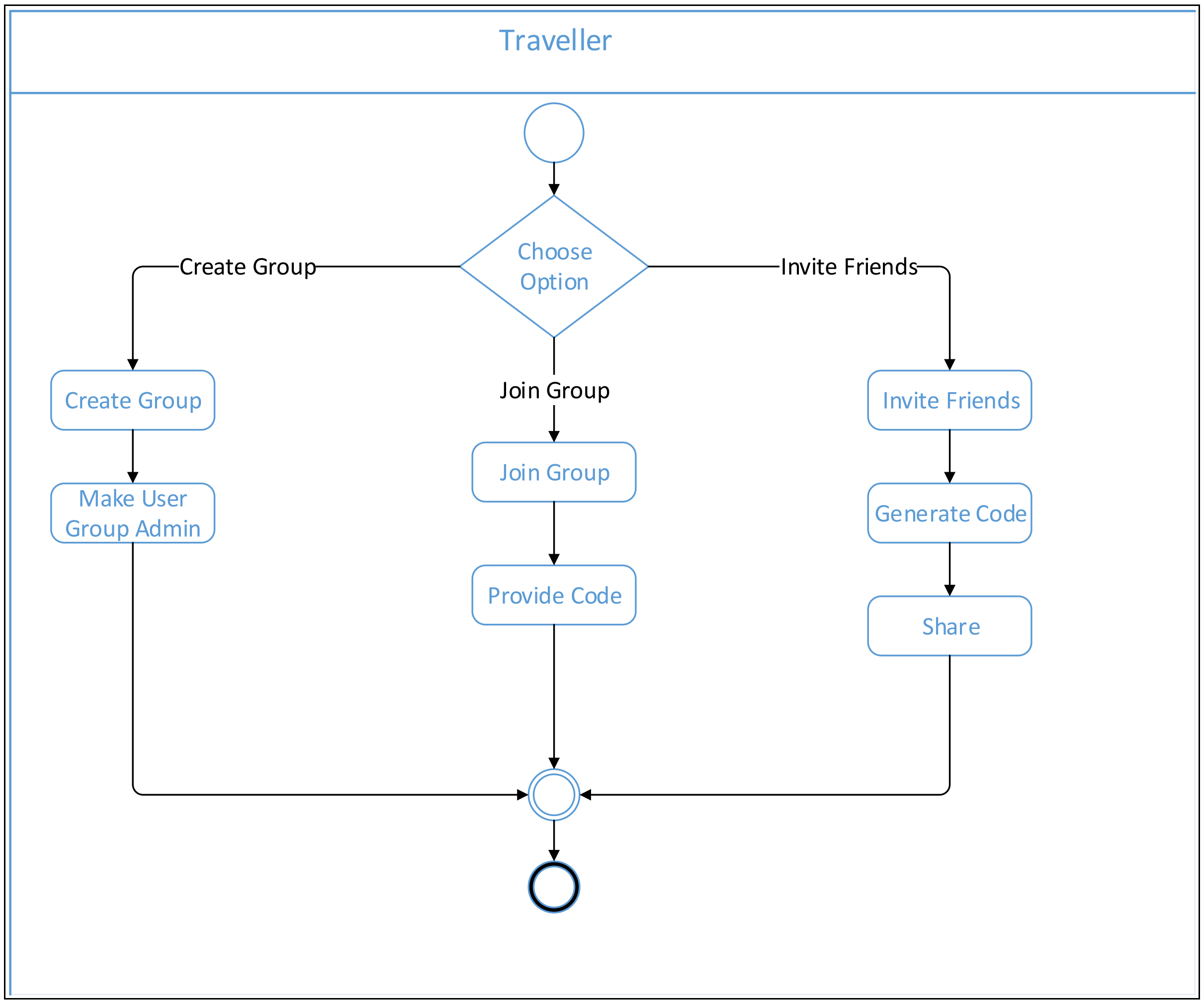


Figure 11: Swim Lane Diagram of Pre Tour Management

Level 1.4:

Use Case:

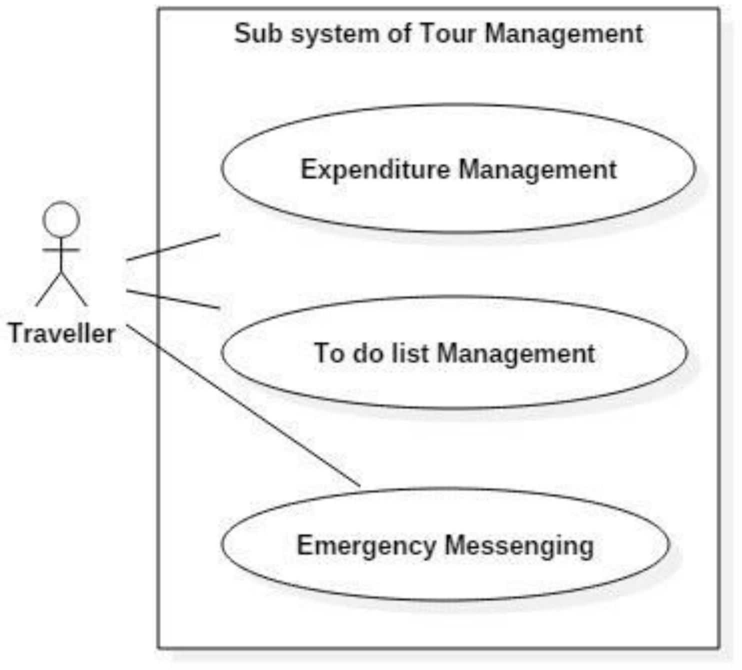


Figure 12: Use Case Level 1.4

Use Case Id	**UC_1.4**
Use Case Name	Tour Management
Primary actor	Traveller
Goals in context	Manage expenditure and tour plans
Preconditions	User is authenticated
Trigger	User inserts expenditure info or tour plans
Action in reply	User can manage expenditure info and tour plans
Exceptions	No exceptions.
Secondary Actors	No secondary actor.

Activity Diagram:

This level is very primitive. I will illustrate it in the level 1.4.1 and 1.4.2

Swim Lane Diagram:

This level is very primitive. I will illustrate it in the level 1.4.1 and 1.4.2

Level 1.4.1:

Use Case:

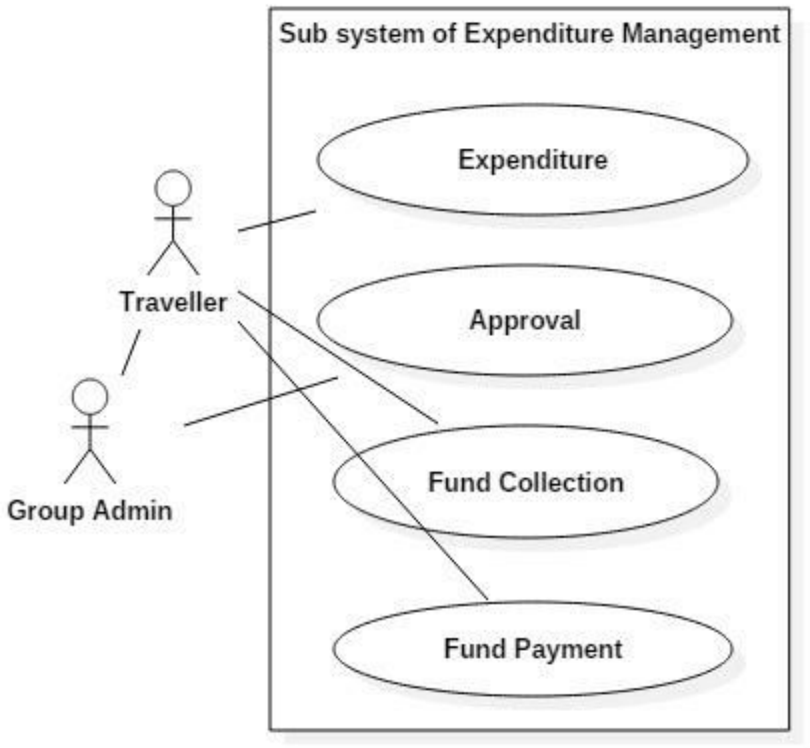


Figure 13: Use Case Level 1.4.1

Use Case Id	**UC_1.4.1**
Use Case Name	Expenditure Management
Primary actor	Traveller
Goals in context	Manage expenditure
Preconditions	User is authenticated
Trigger	User inserts personal or group expenditure info
Action in reply	User can manage expenditure
Exceptions	No exceptions.
Secondary Actors	No secondary actor.

Activity Diagram:

Activity diagram of level 1.4.1 is

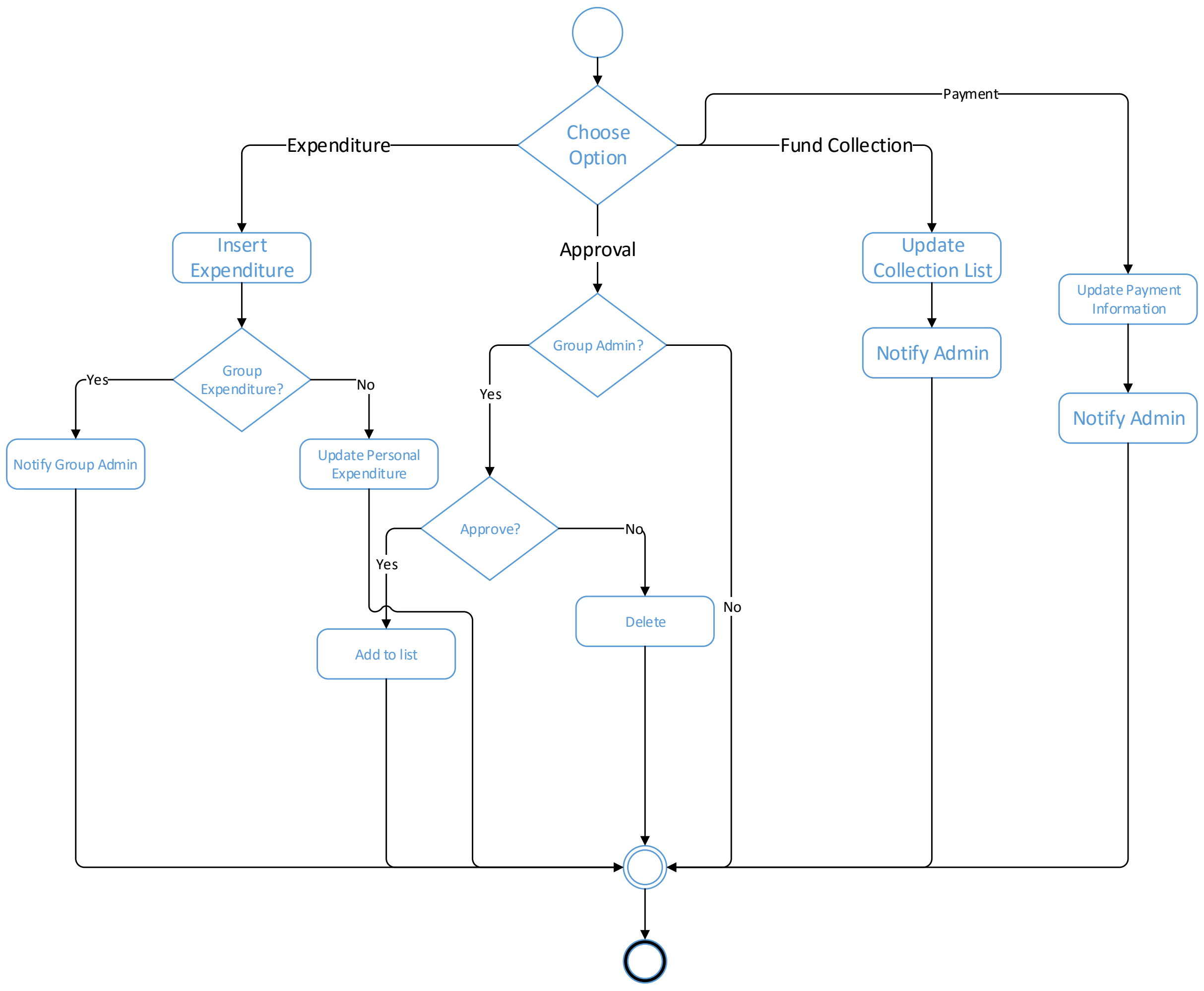


Figure 14: Activity Diagram of Expenditure Management

Swim Lane Diagram:

Swim lane diagram of level 1.4.1 is

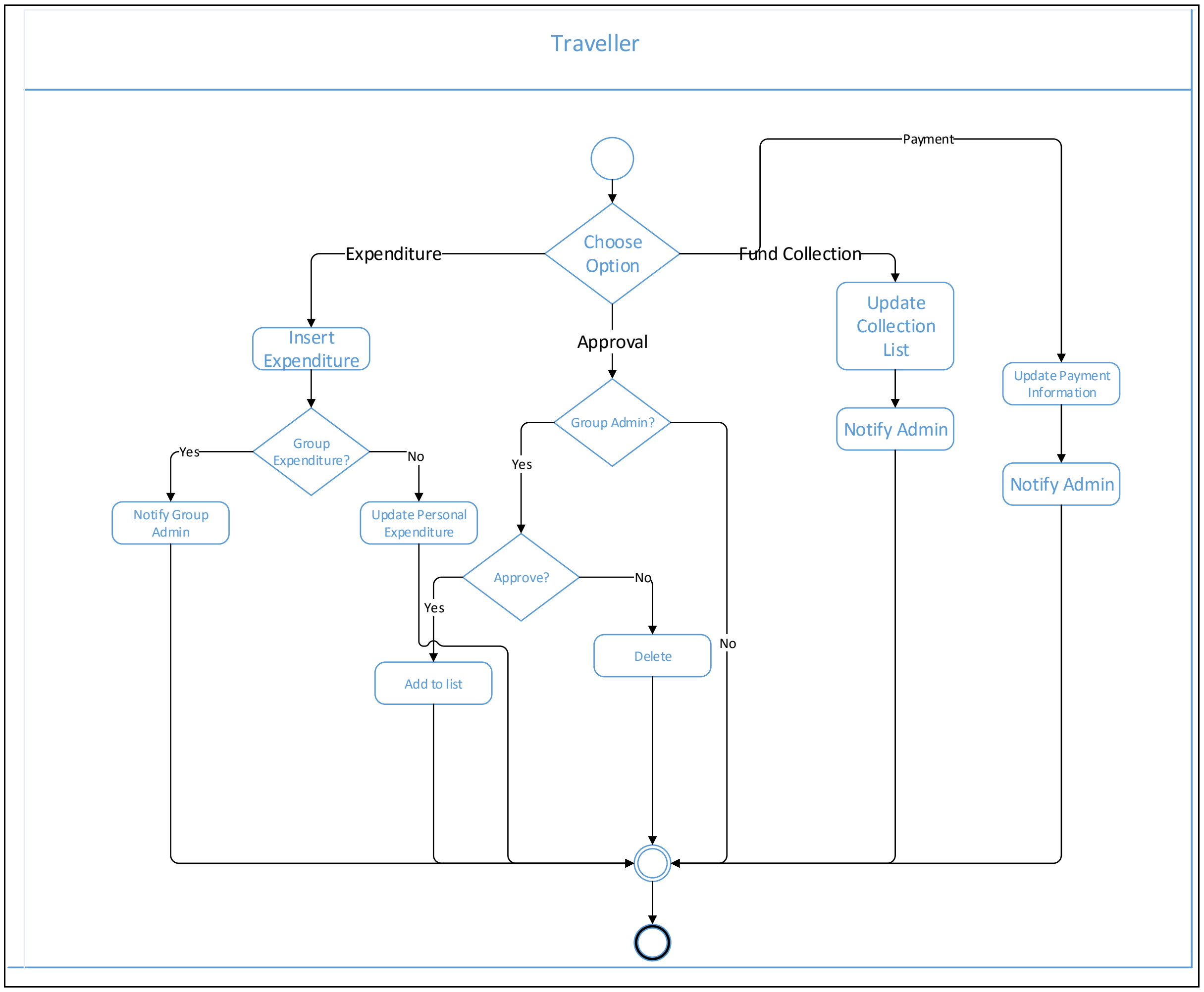


Figure 15: Swim Lane Diagram of Expenditure Management

Level 1.4.2:

Use Case:

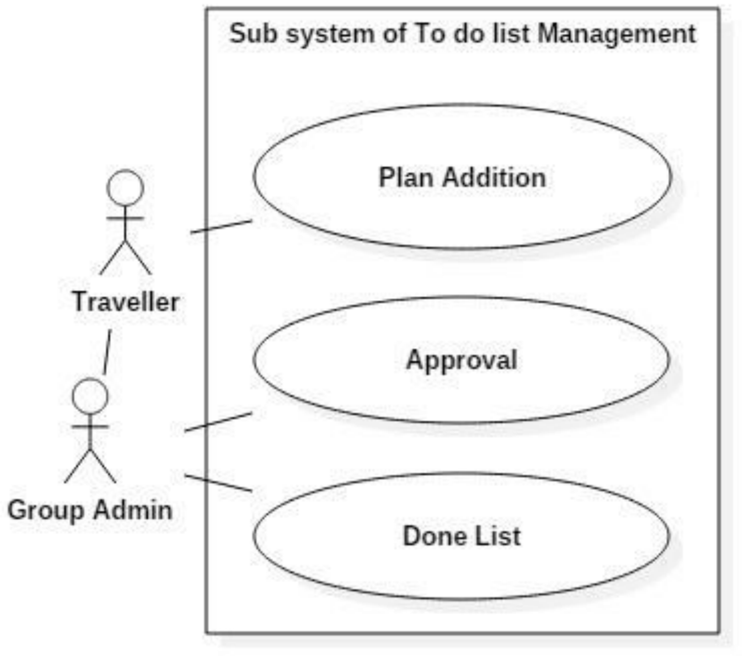


Figure 16: Use Case Level 1.4.2

Use Case Id	**UC_1.4.2**
Use Case Name	Tour Plan Management
Primary actor	Traveller
Goals in context	Manage tour plans
Preconditions	User is authenticated
Trigger	User inserts tour plans
Action in reply	User can manage tour plans
Exceptions	No exceptions.
Secondary Actors	No secondary actor.

Activity Diagram:

Activity diagram of level 1.4.2 is

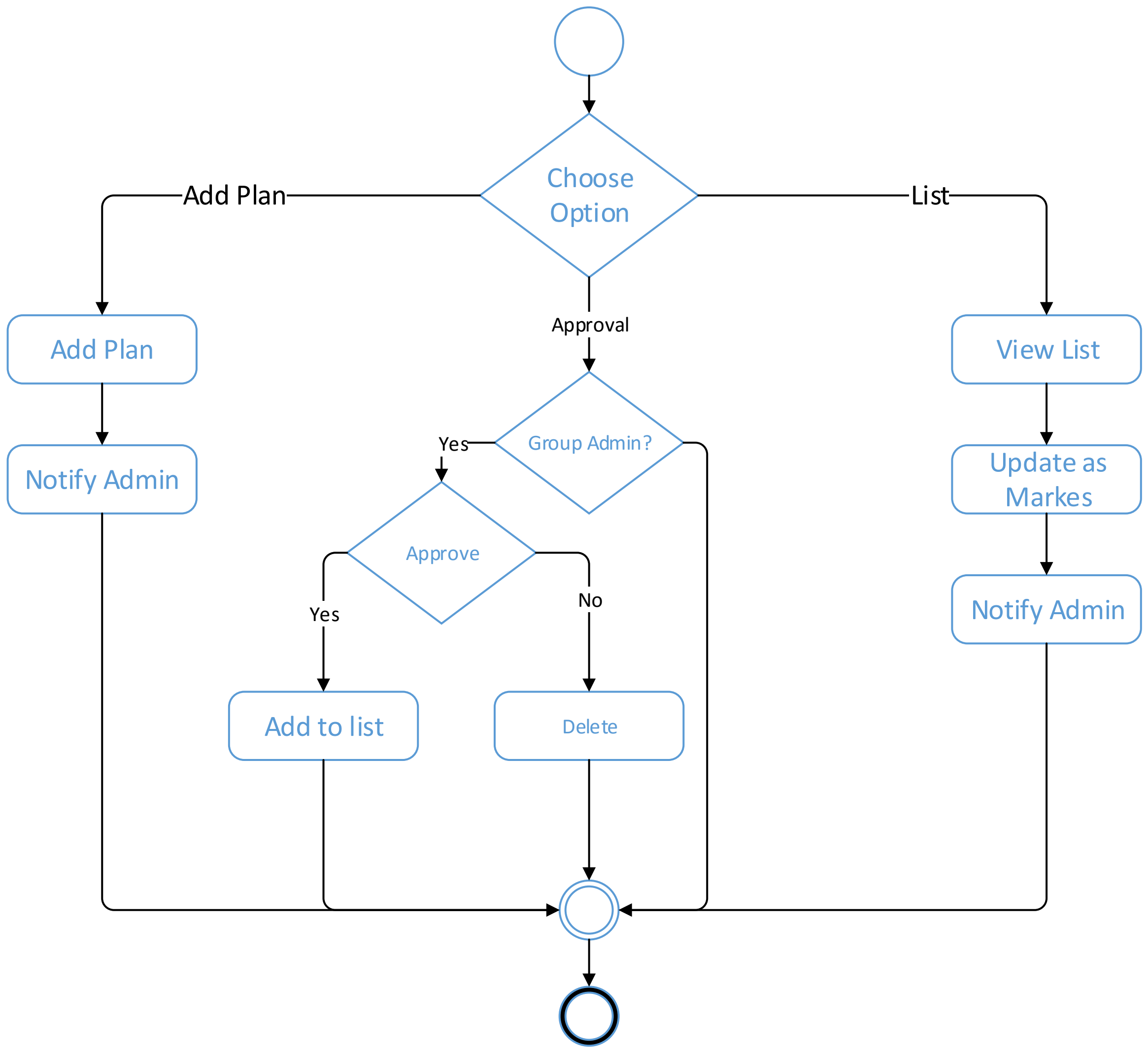


Figure 17: Activity Diagram of To Do List Management

Swim Lane Diagram:

Swim lane diagram of level 1.4.2 is

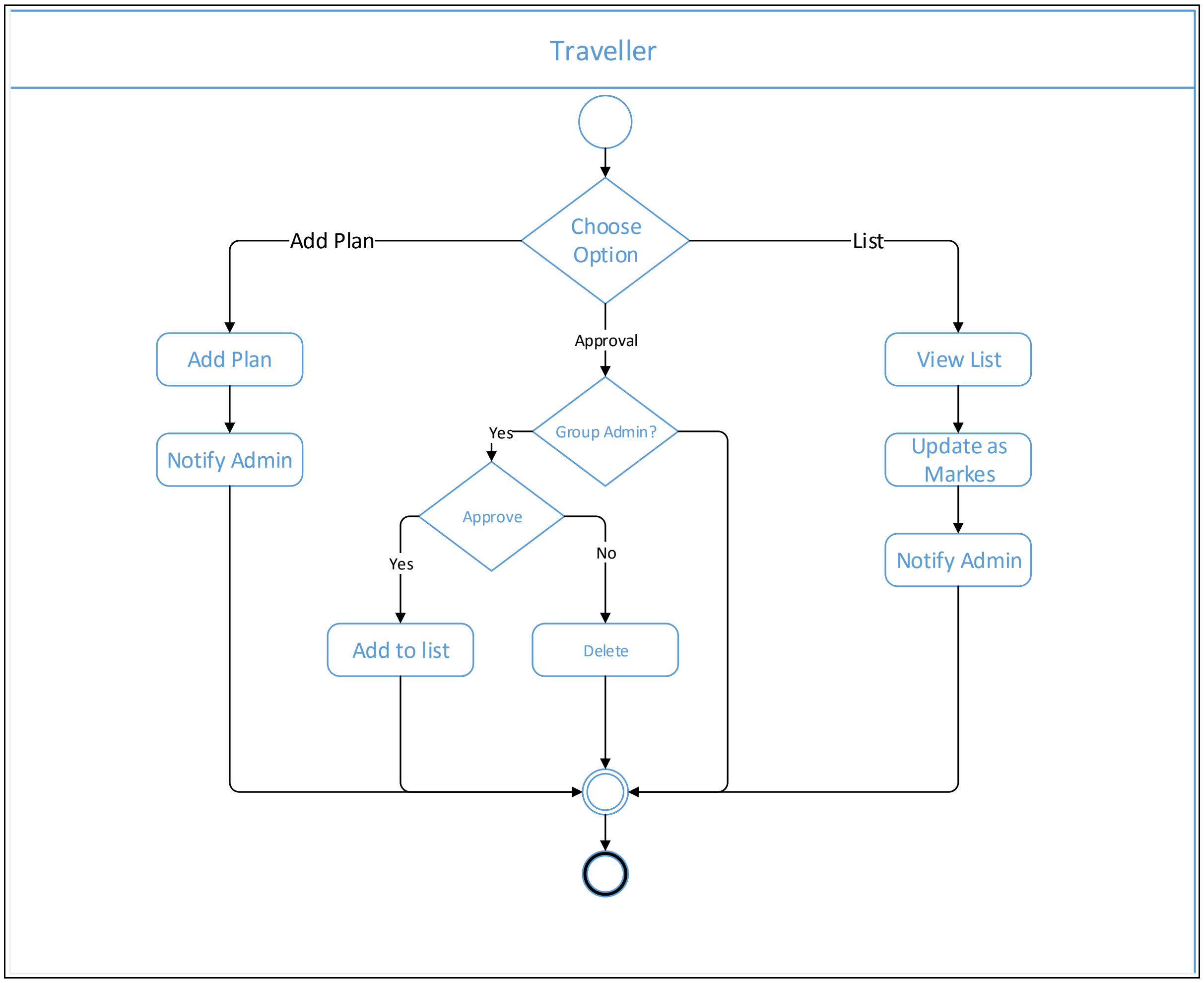


Figure 18: Swim Lane Diagram of To Do List Management

Level 1.4.3:

Activity Diagram:

Activity diagram of level 1.4.3 is

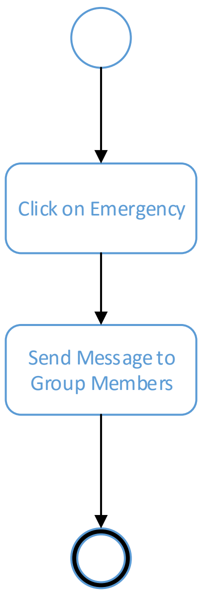


Figure 19: Activity Diagram of Emergency Message

Swim Lane Diagram:

Swim lane diagram of level 1.4.3 is

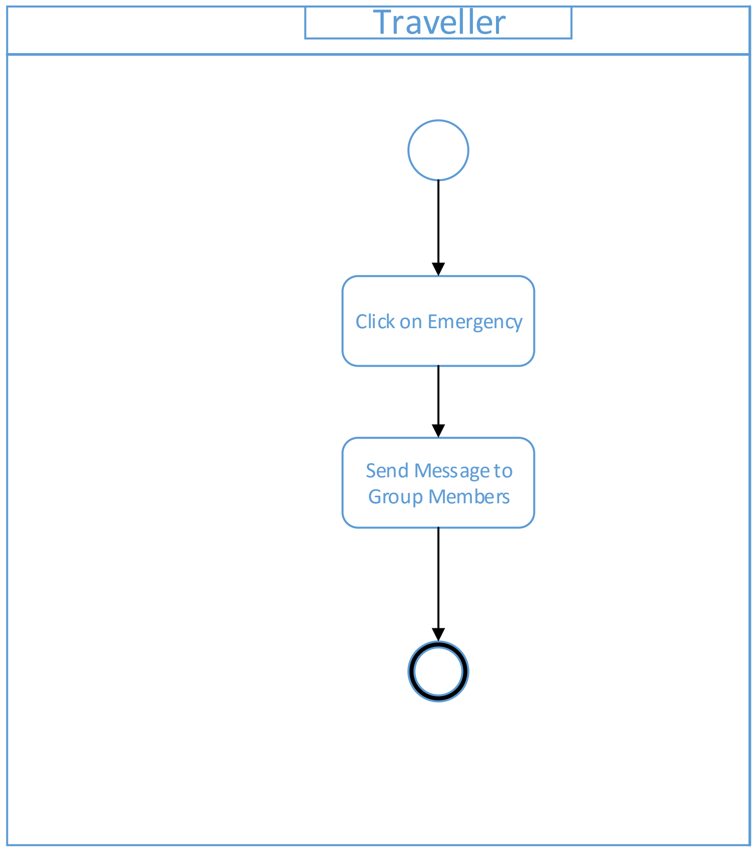


Figure 20: Swim Lane Diagram of Emergency Message

Level 1.5:

Use Case:

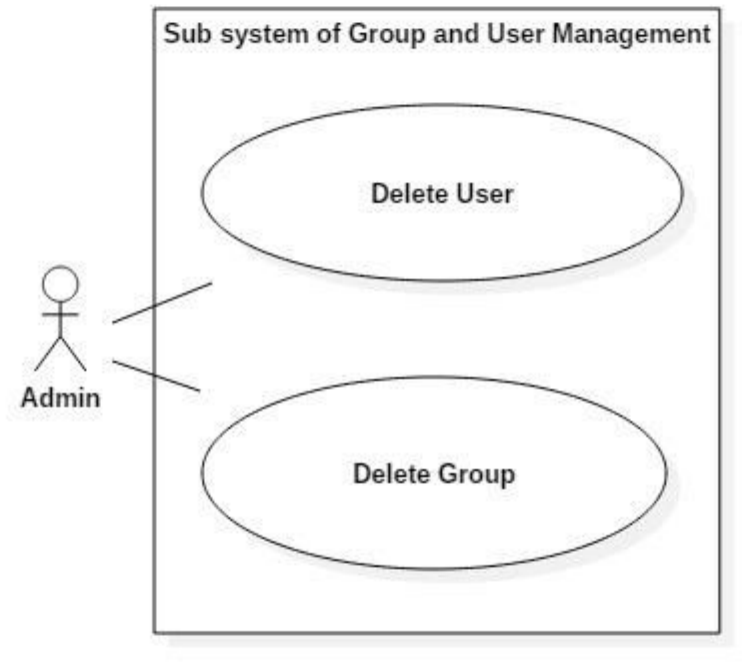


Figure 21: Use Case Level 1.5

Use Case Id	UC_1.5
Use Case Name	User and Group Management
Primary actor	Admin
Goals in context	Manage users and groups
Preconditions	Admin is authenticated
Trigger	Admin deletes user or group
Action in reply	User or groups get deleted
Exceptions	No exceptions.
Secondary Actors	No secondary actor.

Activity Diagram:

Activity diagram of level 1.5 is

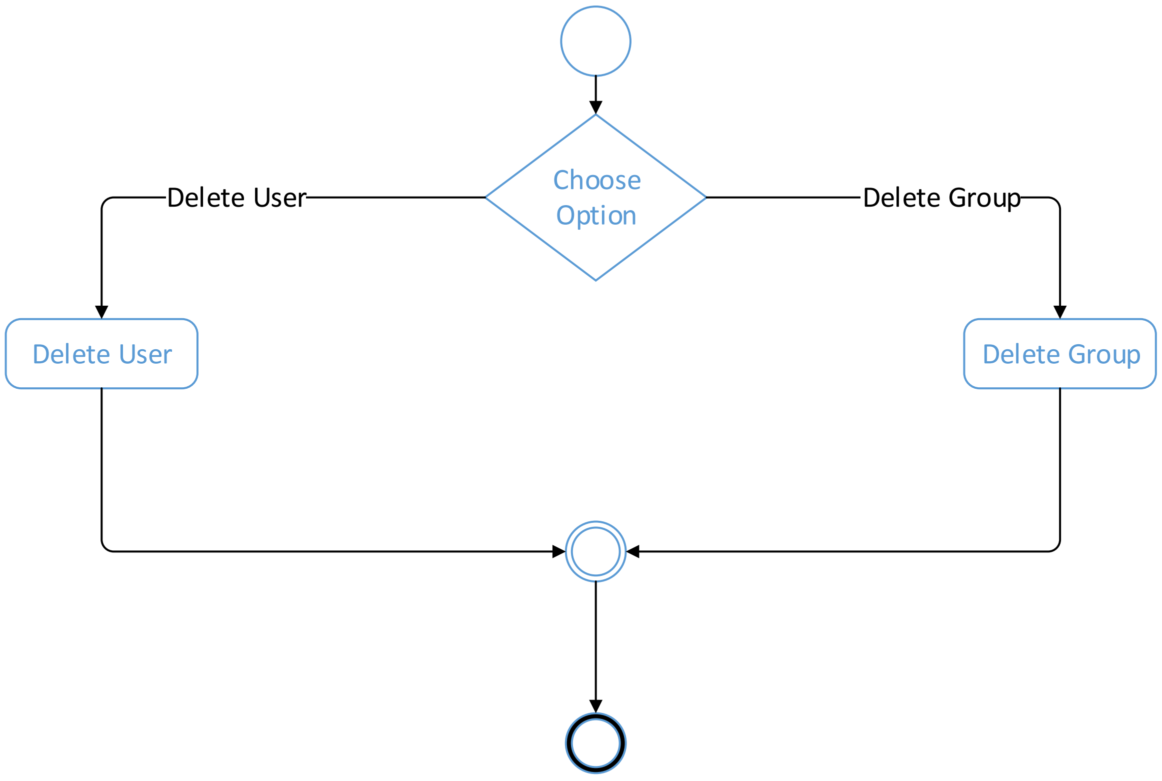


Figure 22: Activity Diagram of User Management

Swim Lane Diagram:

Swim lane diagram of level 1.5 is

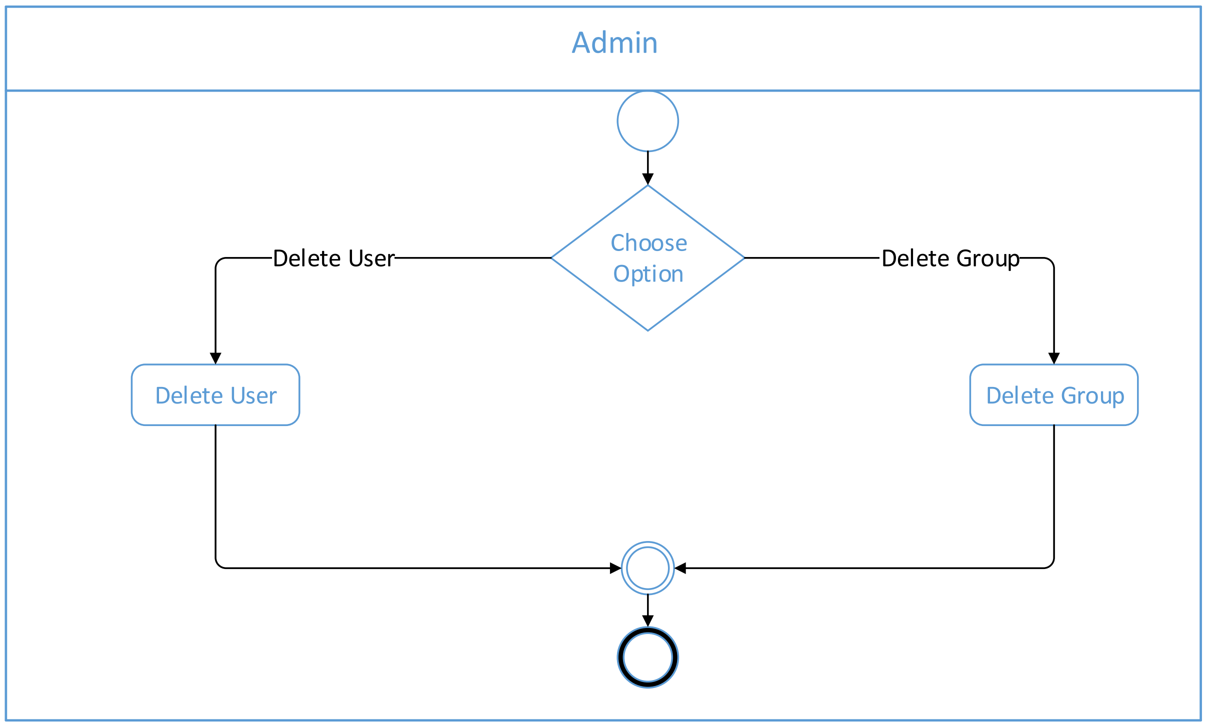


Figure 23: Swim Lane Diagram of User Management

Level 1.6:

Use Case:

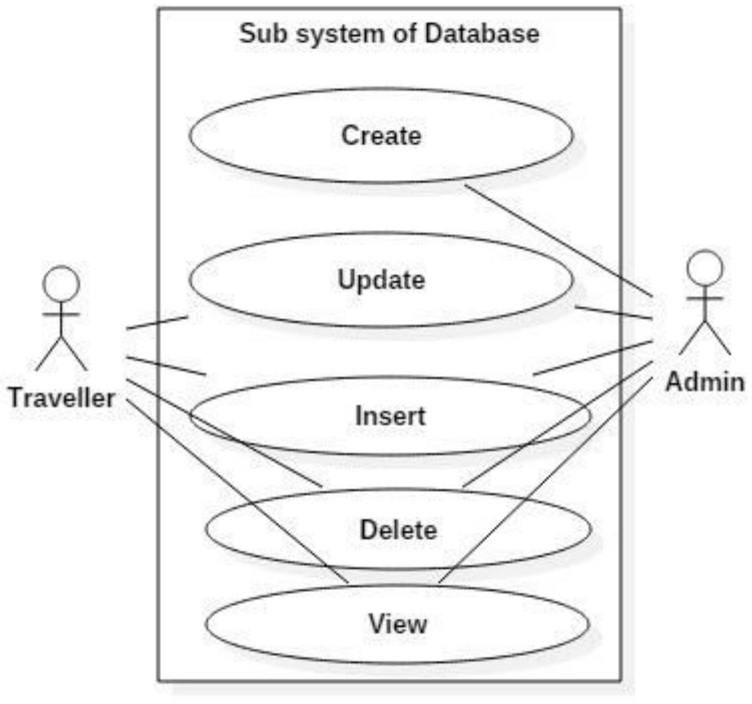


Figure 24: Use Case Level 1.6

Use Case Id	UC_1.6
Use Case Name	Database Management
Primary actor	Admin, Travellers
Goals in context	Manage database
Preconditions	Admin and user is authenticated
Trigger	Admin or user update information
Action in reply	Information gets updated
Exceptions	No exceptions.
Secondary Actors	No secondary actor.

Activity Diagram:

Activity diagram of level 1.6 is

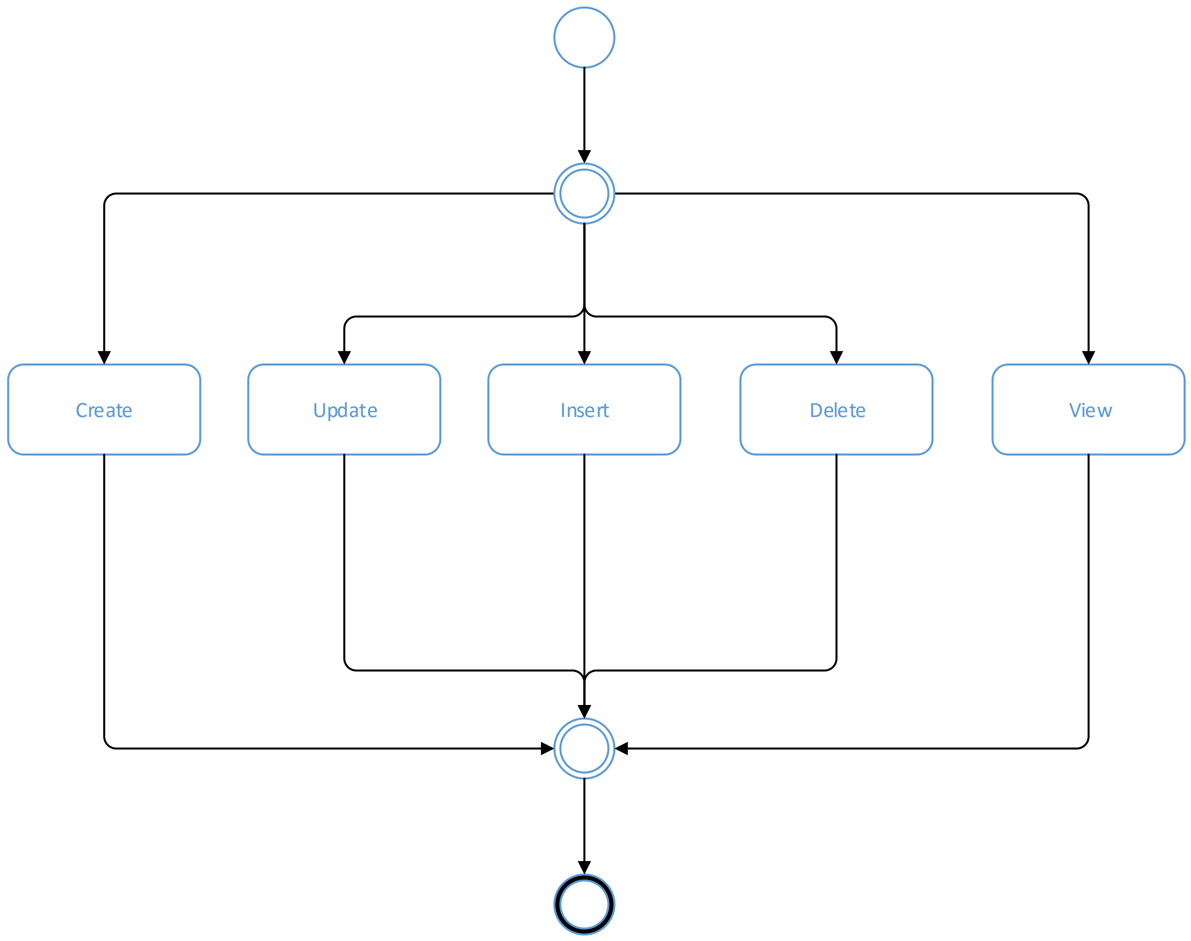


Figure 25: Activity Diagram of Database Management

Swim Lane Diagram:

Swim lane diagram of level 1.6 is

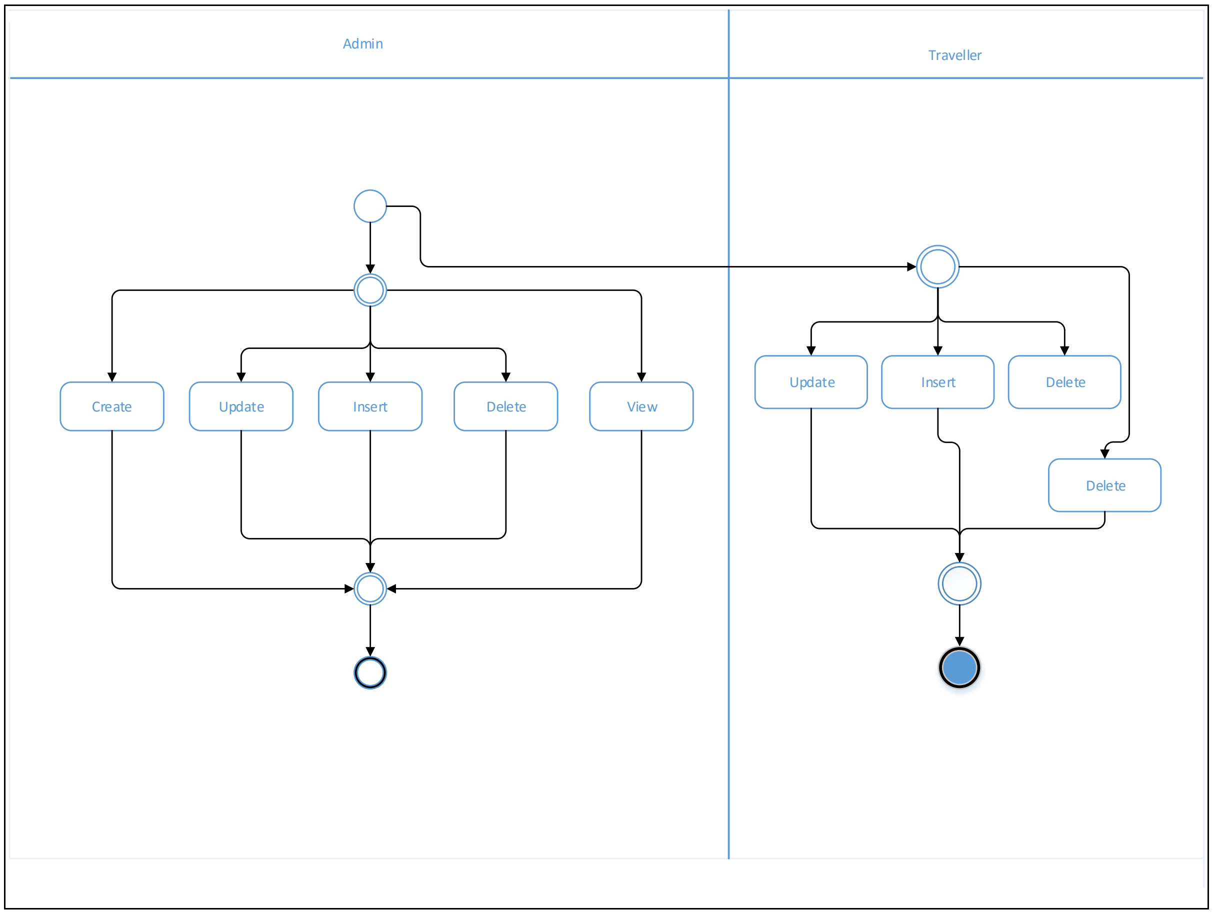


Figure 26: Swim Lane Diagram of Database Management

Data Based Modeling

Data based modeling determines the logical structures of a database and fundamentally determines in which manner data can be stored, organized and manipulated.

Finding Data Object:

Table: 1

No	Noun	Problem/Solution Space (P/S)	Attributes
1	Bangladesh	P	-
2	Land	P	-
3	Places	P	-
4	Sundarbans	P	-
5	Cox's Bazar	P	-
6	Saint Martin	P	-
7	Travellers	P	-
8	Year	P	-
9	Information	S	-
10	Plan	S	-
11	Management	P	-
12	Tour	P	-
13	Expenditure	P	-
14	Mobile	P	-
15	Application	P	-
16	BD Travellers	P	-
17	User	S	18, 28, 50, 59, 60, 66
18	Google ID	S	-
19	Project	P	-
20	Transportation	P	-
21	Interactive	S	9, 61, 68

	Map		
22	**Static Information**	**S**	**9, 10, 23, 62**
23	**Transportation News**	**S**	**24, 25, 26, 27, 63**
24	**Contact Number**	**S**	**-**
25	**Time**	**S**	**-**
26	**Vehicle type**	**S**	**-**
27	**Ticket price**	**S**	**-**
28	**Position**	**S**	**-**
29	**Review**	**S**	**30, 31, 32, 33, 34, 35, 36, 47, 48, 61**
30	**Ratings**	**S**	**-**
31	**Hotel**	**S**	**-**
32	**Places**	**S**	**-**
33	**Food**	**S**	**-**
34	**Danger zone**	**S**	**-**
35	**Others**	**S**	**-**
36	**Color**	**S**	**-**
37	**Part**	**P**	**-**
38	**Group**	**S**	**40, 41, 45, 46, 49, 51, 58, 60, 66, 67**
39	**Option**	**P**	**-**
40	**Tour name**	**S**	**-**
41	**Invitation code**	**S**	**-**
42	**Facebook**	**P**	**-**
43	**Bluetooth**	**P**	**-**
44	**Messenger**	**P**	**-**
45	**Group Admin**	**S**	**-**
46	**Request**	**S**	**-**
47	**Inappropriate**	**S**	**-**
48	**Abusive**	**S**	**-**
49	**To do list**	**S**	**-**

50	**Personal Expenditure**	**S**	**52, 54, 55, 66**
51	**Group Expenditure**	**S**	**52, 53, 54, 55**
52	**Total**	**S**	**-**
53	**Fund**	**S**	**-**
54	**Amount**	**S**	**56, 57**
55	**List**	**S**	**-**
56	**Time**	**S**	**-**
57	**Item**	**S**	**-**
58	**Payment List**	**S**	**55**
59	**User ID**	**S**	**-**
60	**Group ID**	**S**	**-**
61	**Review ID**	**S**	**-**
62	**Static Info ID**	**S**	**-**
63	**Transport News ID**	**S**	**-**
64	**Admin**	**S**	**-**
65	**Admin ID**	**S**	**-**
66	**Personal Expenditure ID**	**S**	**-**
67	**Group Admin ID**	**S**	**-**
68	**Map ID**	**S**	**-**

Data objects and their attributes:

- User
  1. User ID
  2. Current Position
  3. Google ID

- Map:
    1. Map ID
    2. Review ID

- Static Information:
    1. Static Information ID
    2. Plan
    3. Information

- Transportation news:
    1. Transport News ID
    2. Contact Number
    3. Time
    4. Vehicle Type
    5. Price

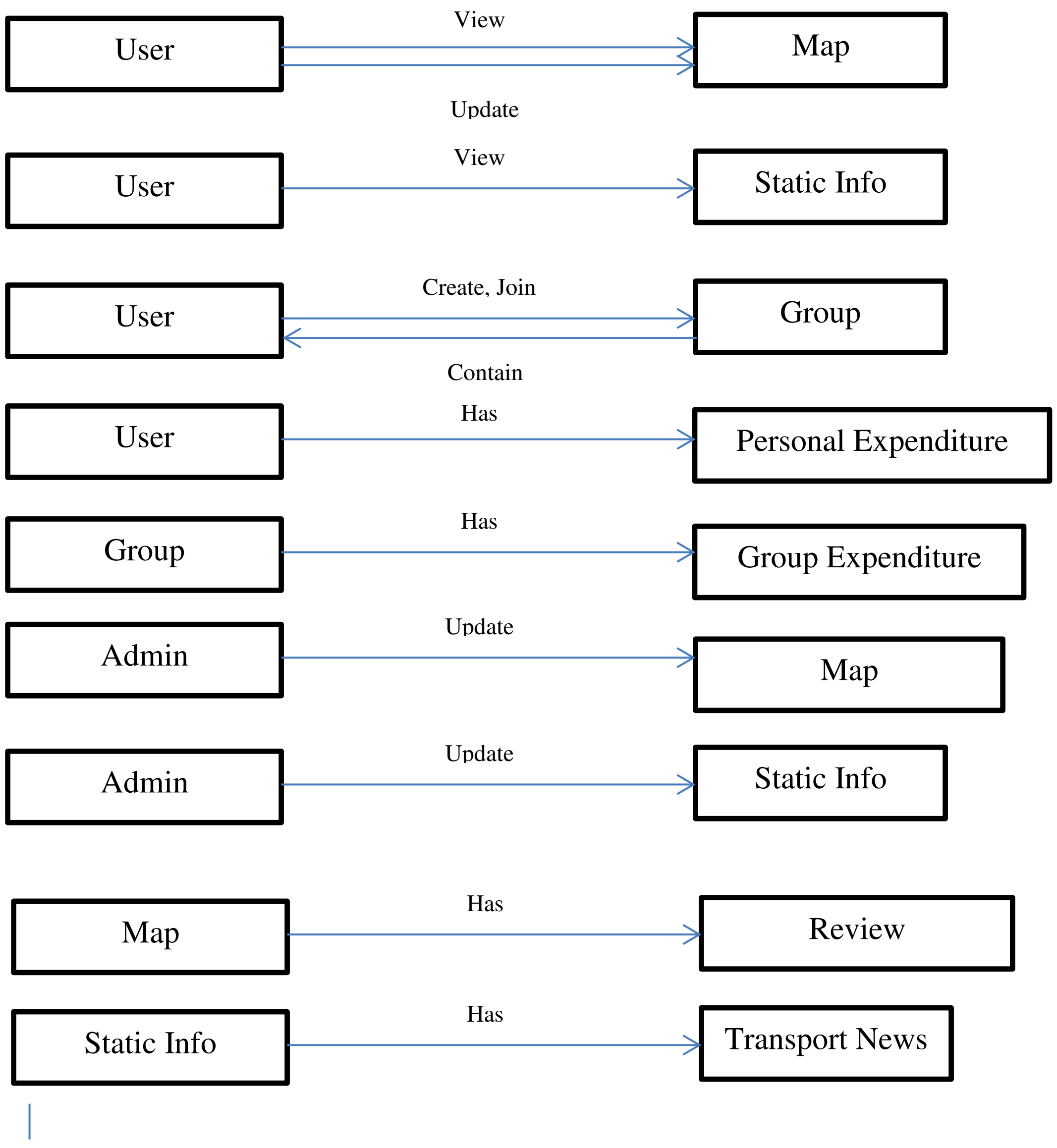
- Review:
    1. Review ID
    2. Ratings
    3. Hotel
    4. Places
    5. Food
    6. Danger Zone
    7. Inappropriate
    8. Abusive

- Group:
    1. Group ID
    2. Tour Name
    3. Invitation Code
    4. Group Admin ID
    5. Payment list

- Personal Expenditure:
    1. Personal Expenditure ID
    2. Total
    3. Amount
    4. List

- Group expenditure:
    1. Total
    2. Fund
    3. Amount
    4. List

Relations between data objects:



ER-Diagram

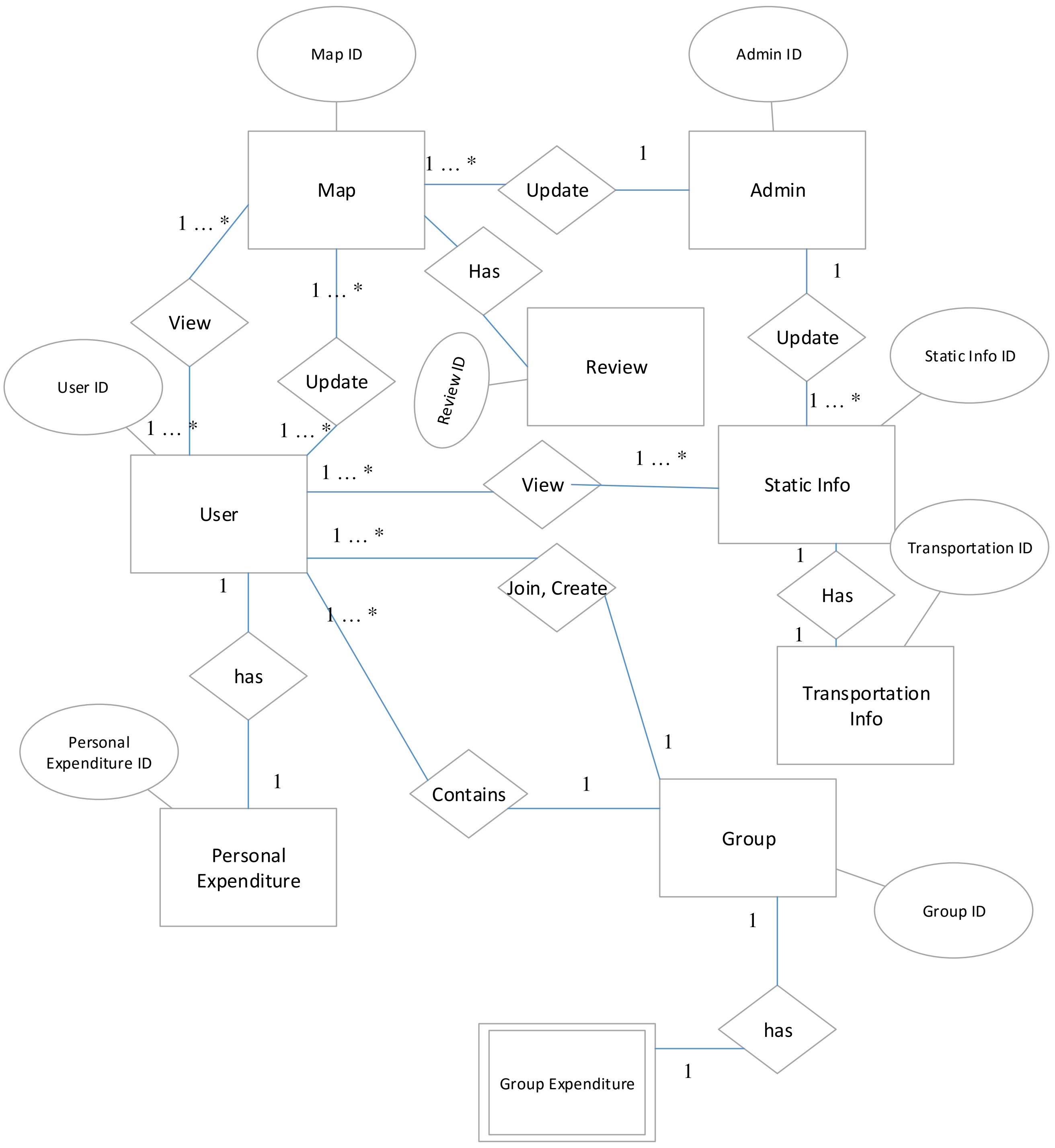


Figure 26: ER-Diagram

Data Tables:

Table: 2

User

Attribute	**Type**
<u>User ID</u>	**Integer**
Google ID	**Varchar2(50)**
Map ID (Foreign Key)	**Integer**
Group ID (Foreign Key)	**Integer**
Personal Expenditure ID (Foreign Key)	**Integer**
Current Position	**Varchar2(40)**

Table: 3

Map

Attribute	**Type**
<u>Map ID</u>	**Integer**
Static Info ID (Foreign Key)	**Integer**
Review ID (Foreign Key)	**Integer**
Admin ID (Foreign Key)	**Integer**

Table: 4

Static Information

Attribute	**Type**
<u>Static Info ID</u>	**Integer**
Transportation News ID (Foreign Key)	**Integer**
Plan	**Varchar2(200)**
Information	**Varchar2(200)**

Table: 5

Review

Attribute	**Type**
<u>Review ID</u>	**Integer**
Ratings	**Number**
Hotel	**Varchar2(50)**
Places	**Varchar2(50)**
Food	**Varchar2(50)**
Danger Zone	**Varchar2(50)**
Inappropriate	**Integer**
Abusive	**Integer**

Table: 6

Group

Attribute	**Type**
<u>Group ID</u>	**Integer**
Tour Name	**Varchar2(50)**
Invitation Code	**Varchar2(50)**
Group Admin ID	**Integer**
Payment list	**Varchar2(500)**

Table: 7

Personal Expenditure

Attribute	**Type**
Personal Expenditure ID	**Integer**
Total	**Integer**
Amount	**Integer**
List	**Varchar2(50)**

Table: 8

Group Expenditure

Attribute	**Type**
Group ID (Foreign Key)	**Integer**
Total	**Integer**
Fund	**Integer**
Amount	**Integer**
List	**Varchar2 (200)**

Class Based Modeling

Class Based Modeling

Class based modeling is designed to demonstrate the whole software on the view or perspective of object oriented concept. In this model what the objects are and what their responsibilities will be, how they will interact with each other is defined very clearly.

1.1. General Classifications

After selecting the nouns by grammatical parsing from the solution space of the story, these are characterized in seven general classifications. The seven general characteristics are as follow:

1. External entities
2. Things
3. Occurrences
4. Roles
5. Organizational units
6. Places
7. Structures

Here the 'passed' nouns are the potential classes and the 'failed' nouns become the attributes of the classes.

Table: 9

No	Noun	General Characteristics
1	**Information**	**-**
2	**Plan**	**-**
3	**User**	**4, 5**
4	**Google ID**	**2**
5	**Interactive Map**	**2**

6	**Static Information**	**-**
7	**Transportation News**	**-**
8	**Contact Number**	**2**
9	**Time**	**-**
10	**Vehicle type**	**-**
11	**Ticket price**	**-**
12	**Position**	**-**
13	**Review**	**2**
14	**Ratings**	**2**
15	**Hotel**	**2**
16	**Places**	**2**
17	**Food**	**2**
18	**Danger zone**	**-**
19	**Others**	**-**
20	**Color**	**2**
21	**Group**	**5**
22	**Tour name**	**2**
23	**Invitation code**	**2**
24	**Group Admin**	**4**
25	**Request**	**2**
26	**Inappropriate**	**-**
27	**Abusive**	**-**
28	**To do list**	**2**
29	**Personal Expenditure**	**2**
30	**Group Expenditure**	**2**
31	**Total**	**-**
32	**Fund**	**2**
33	**Amount**	**-**
34	**List**	**2**

35	**Time**	**2**
36	**Item**	**2**
37	**Payment List**	**2**
38	**User ID**	**2**
39	**Group ID**	**2**
40	**Review ID**	**2**
41	**Static Info ID**	**2**
42	**Transport News ID**	**2**
43	**Admin**	**4, 5**
44	**Admin ID**	**2**
45	**Personal Expenditure ID**	**2**
46	**Group Admin ID**	**2**
47	**Map ID**	**2**
48	**Database**	**1**

1.2. Selection characteristics

The potential classes are then selected as class by six ‘selection characteristics’. A potential class becomes a class when it fulfills majority of all six characteristics.

1. Retained Information
2. Needed Services
3. Multiple Attributes
4. Common attributes
5. Common operations
6. Essential requirements

Table: 10

No	Noun	Selection Characteristics
1	User	1, 2, 3, 4, 5
2	Google ID	1
3	Interactive Map	2
4	Contact Number	1
5	Review	1
6	Ratings	1
7	Hotel	1
8	Places	1
9	Food	1
10	Color	1
11	Group	1, 2, 3, 4, 5
12	Tour name	1
13	Invitation code	1
14	Group Admin	1
15	Request	1
16	To do list	1
17	Personal Expenditure	1, 4
18	Group Expenditure	1, 4
19	Fund	1
20	List	1
21	Time	1
22	Item	1
23	Payment List	1
24	User ID	1
25	Group ID	1
26	Review ID	1
27	Static Info ID	1

28	**Transport News ID**	**1**
29	**Admin**	**3, 5**
30	**Admin ID**	**1**
31	**Personal Expenditure ID**	**1**
32	**Group Admin ID**	**1**
33	**Map ID**	**1**
34	**Database**	**1, 2, 3, 4, 5, 6**
35	**Static Information**	**2, 3**

So, the selected classes are:

1. User
2. Group
3. Admin
4. Database

1.3. Specifying Attributes

Every selected class‘s attributes are here specified here. Attributes are generally selected nouns. Sometimes we introduce new attributes.

Table: 11

Class Name	**Attributes**
User	UserId, GoogleId, GroupId
Group	GroupId, GroupAdminId

Admin	AdminId, Password
Database	All Attributes

1.4. Defining Operations

Here verbs are selected from the solution space of the story. Every class's methods are selected.

Table: 12

Class Name	**Attributes**	**Methods**
User	UserId, GoogleId, GroupId	showApi(), logIn(), select(), updateInfo(), view(), calculatePosition(), approve(), report(), suggest(), sendMessage(), logout()
Group	GroupId, GroupAdminId	view(), join(), create(), giveTourName(), generate(), share()
Admin	AdminId, Password	update(), view(), delete(), logIn(), logout()
Database	All Attributes	create(), update(), view(), insert(), delete()

1.5. Classes

Here the selected classes are shown in UML (Unified Modeling Language) form.

Class: User
UserId, GoogleId, GroupId
showApi(), logIn(), select(), updateInfo(), view(), calculatePosition(), approve(), report(), suggest(), sendMessage(), logout()

Class: Group
GroupId, GroupAdminId
view(), join(), create(), giveTourName(), generate(), share()

Class: Admin
AdminId, Password
update(), view(), delete(), logIn(), logout()

Class: Database
All Attributes
create(), update(), delete(), view(), insert()

CRC Diagram

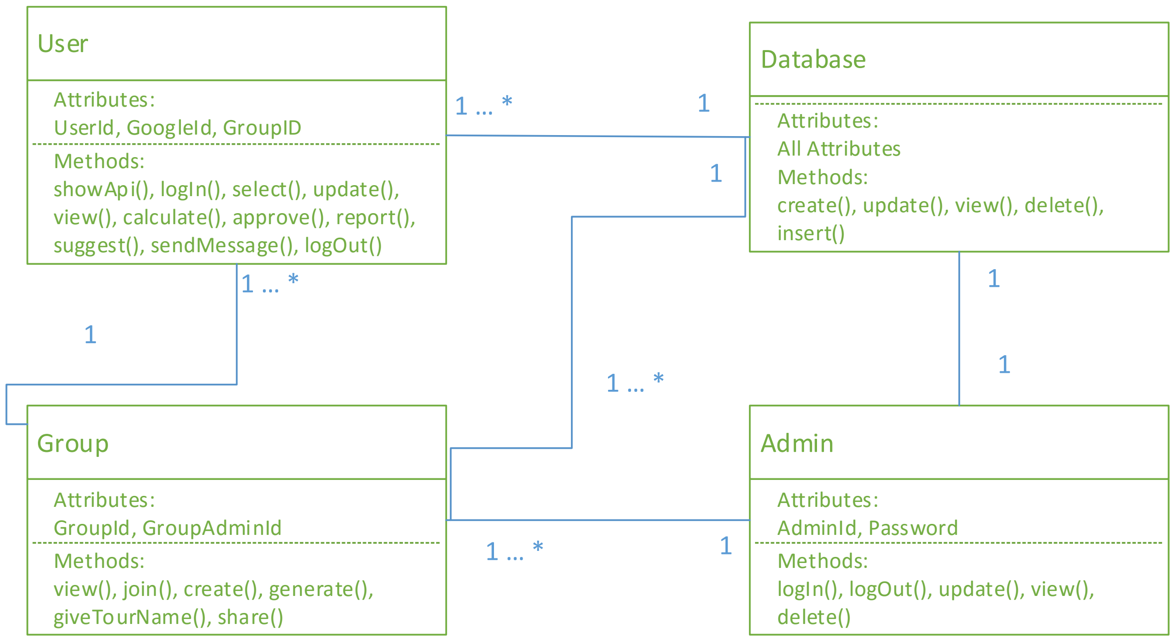

User
Attributes:
UserId, GoogleId, GroupID
Methods:
showApi(), logIn(), select(), update(),
view(), calculate(), approve(), report(),
suggest(), sendMessage(), logOut()
Database
Attributes:
All Attributes
Methods:
create(), update(), view(), delete(),
insert()
Group
Attributes:
GroupId, GroupAdminId
Methods:
view(), join(), create(), generate(),
giveTourName(), share()
Admin
Attributes:
AdminId, Password
Methods:
logIn(), logOut(), update(), view(),
delete()
1 ... *
1
1
1 ... *
1
1
1
1 ... *
1 ... *
1

Figure 27: CRC Diagram

Flow Oriented Model

Although data flow-oriented modeling is perceived as an outdated technique by some software engineers, it continues to be one of the most widely used requirements analysis notations in use today. It provides additional insight into system requirements and flow.

Data Flow Diagram:

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. In the figures, data objects are represented by labeled arrows and transformations are represented by circles.

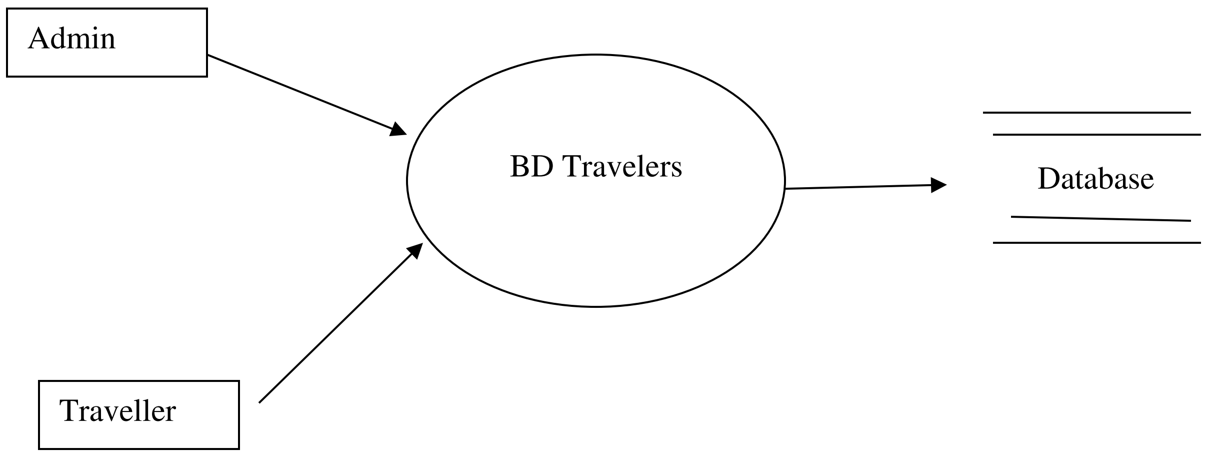


Figure 28: Data Flow Diagram of level 0

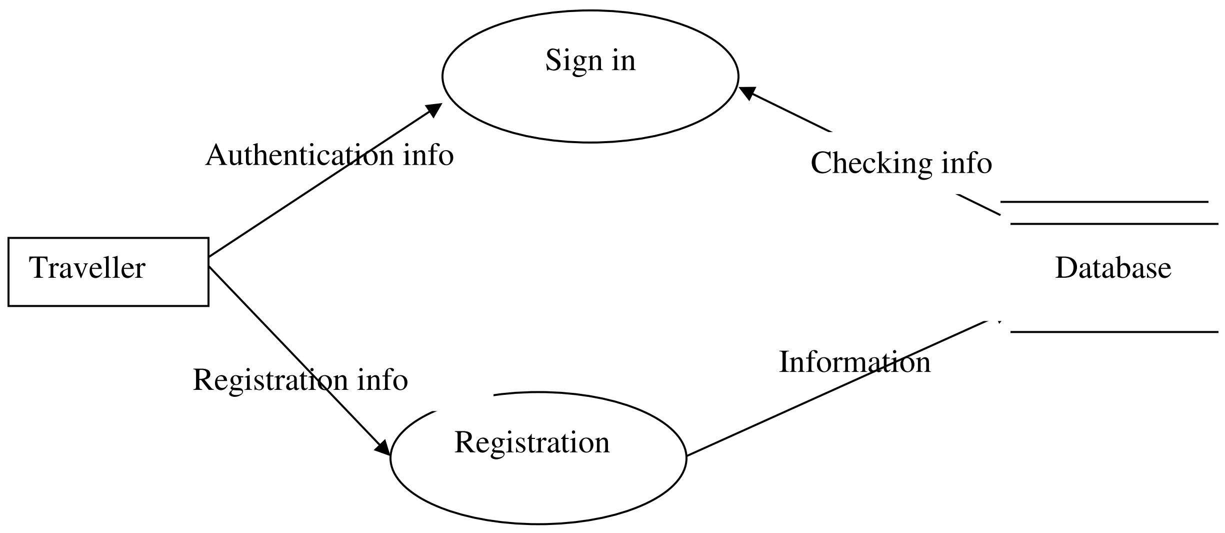


Figure 29: Data Flow Diagram of level 1.1

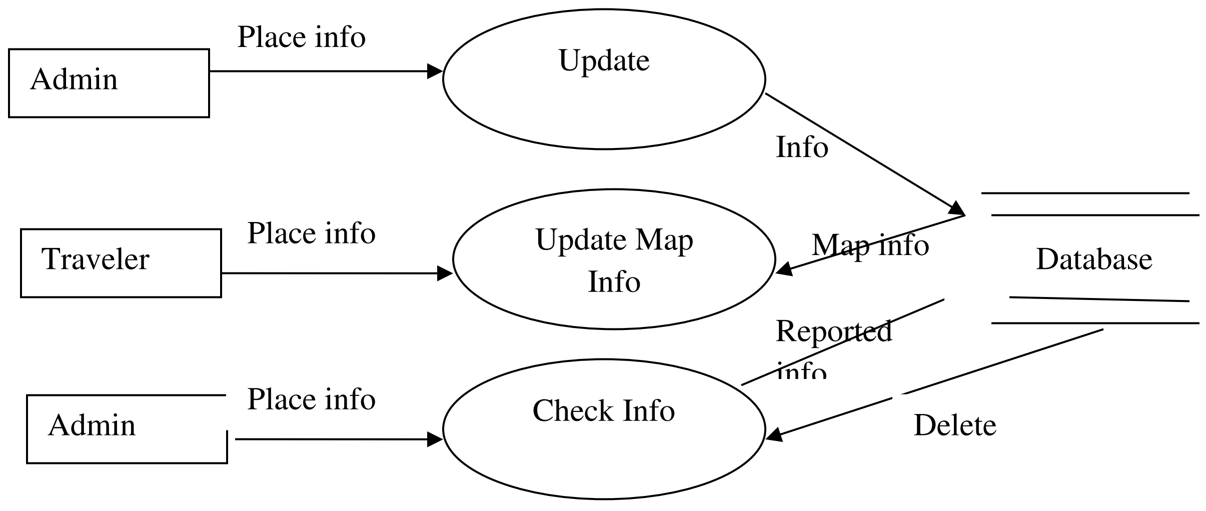


Figure 30: Data Flow Diagram of Level 1.2

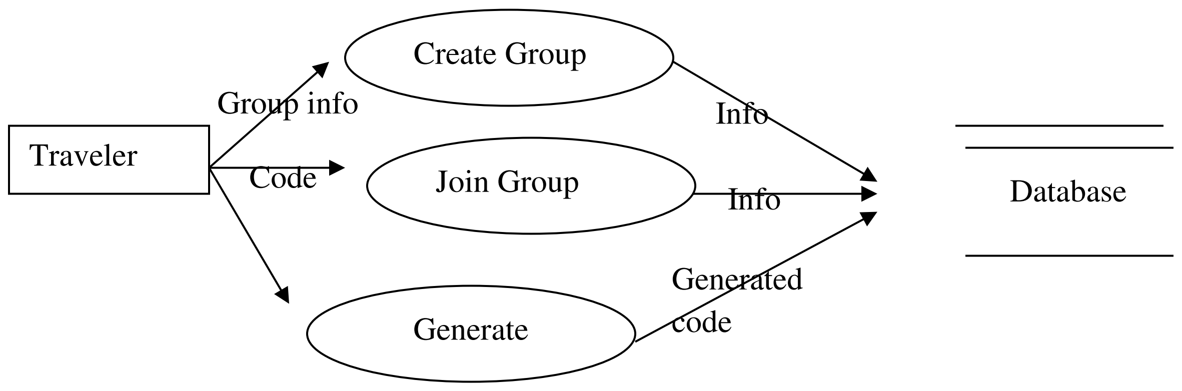


Figure 31: Data Flow Diagram of Level 1.3

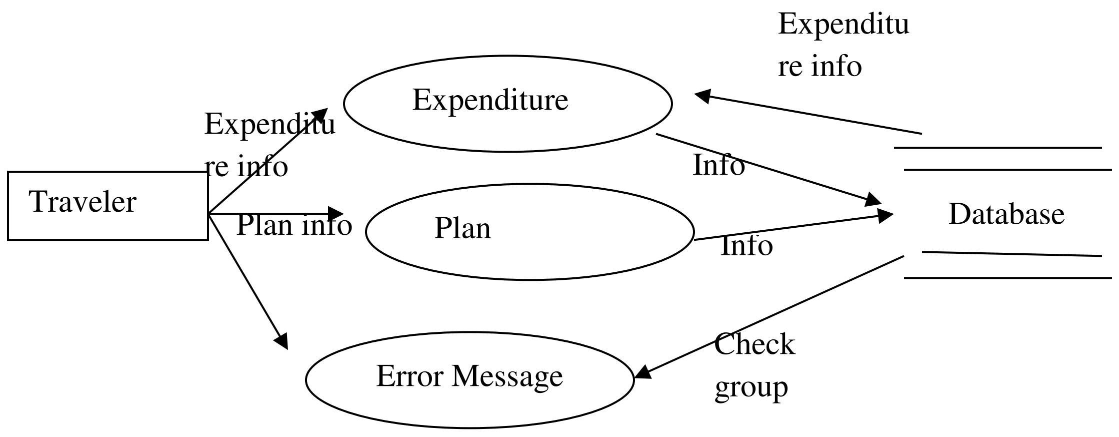


Figure 32: Data Flow Diagram of Level 1.4

Behavioral Model

The behavioral model indicates how software will respond to external events.

State Transition Diagram

State transition diagram represents active states for each class the events (triggers).

State Transition Diagram (User Class):

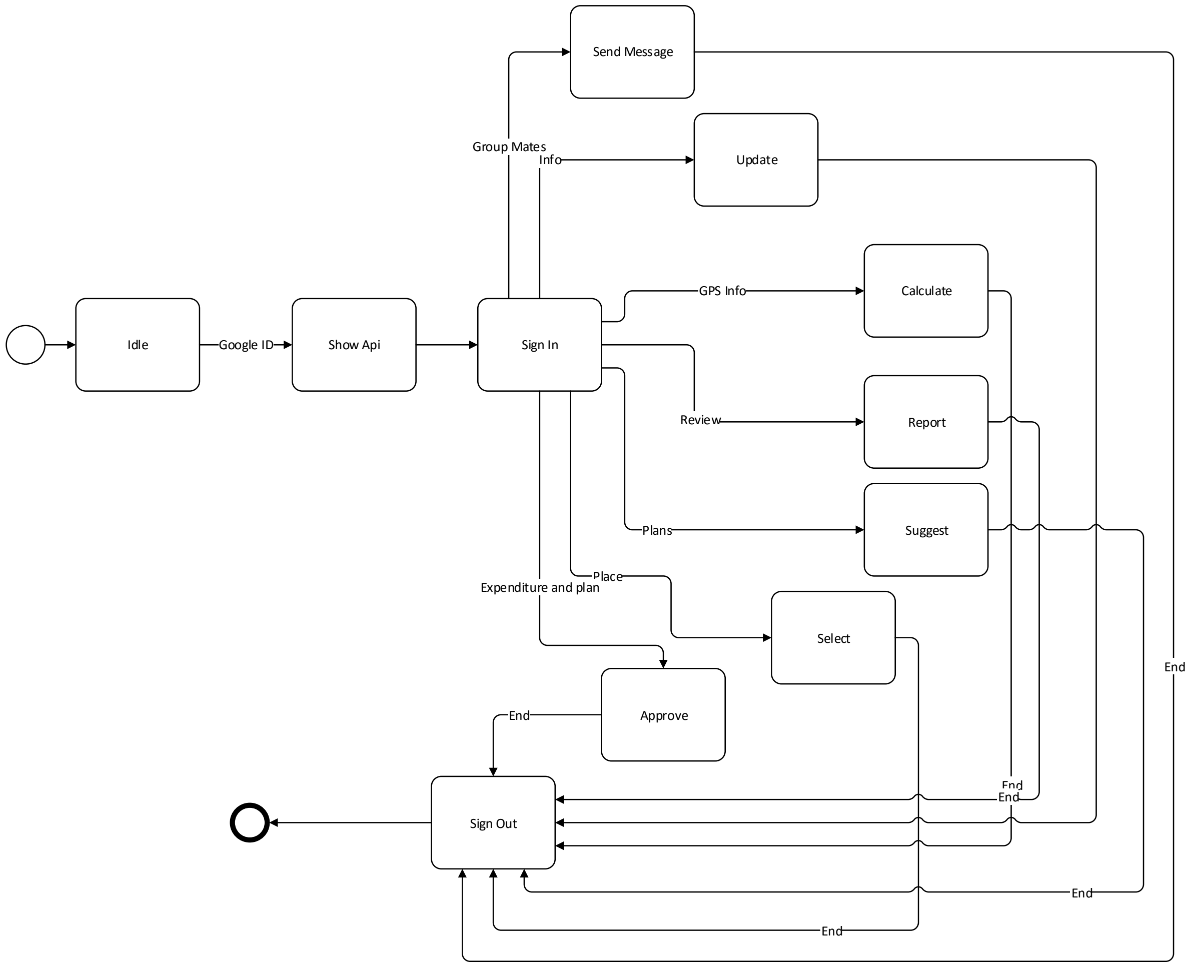


Figure 33: State Transition Diagram (User class)

State Transition Diagram (Group Class):

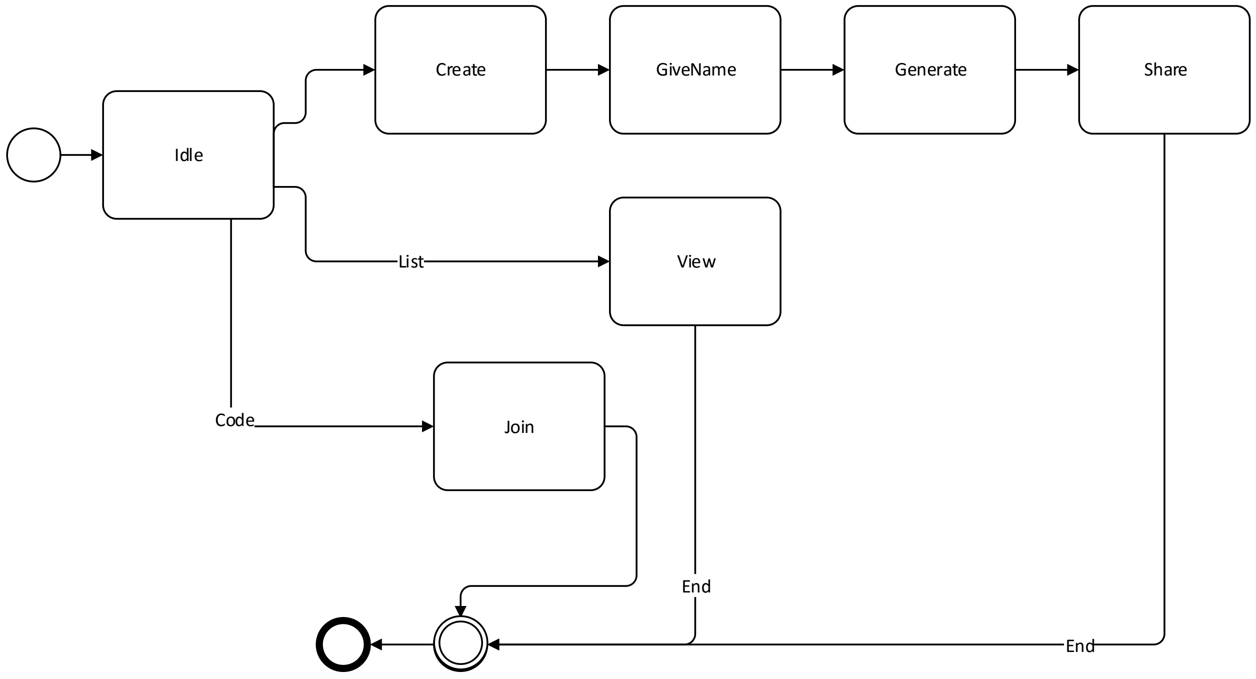


Figure 34: State Transition Diagram (Group class)

State Transition Diagram (Admin Class):

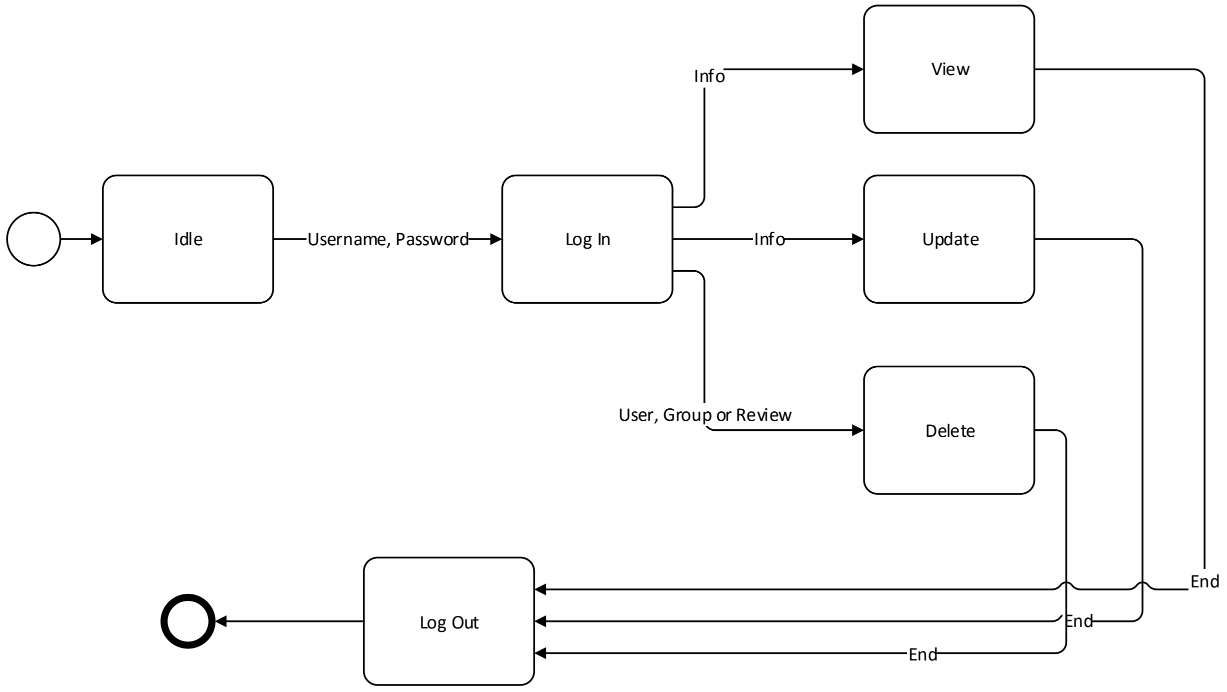


Figure 35: State Transition Diagram (Admin class)

State Transition Diagram (Database Class):

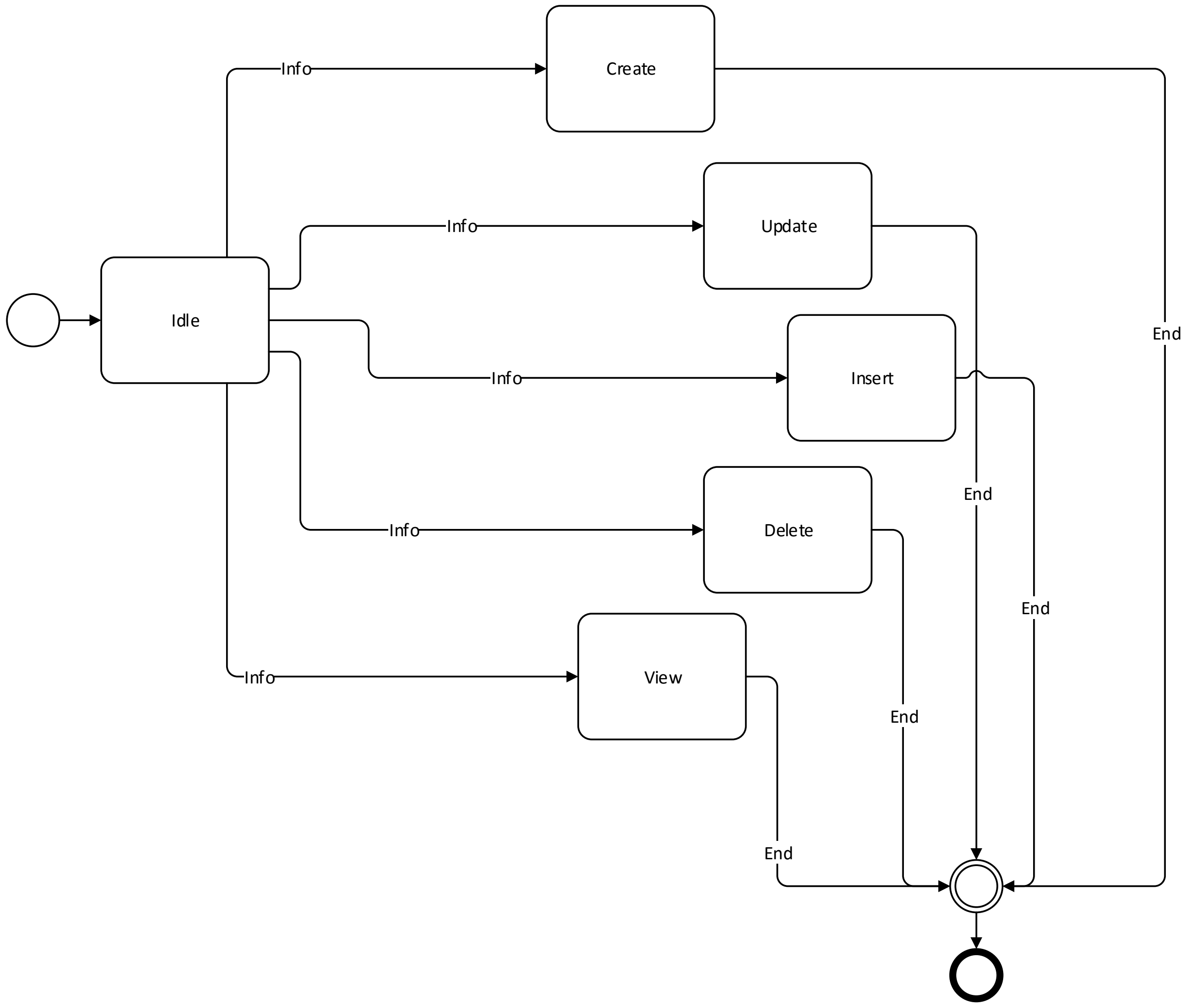
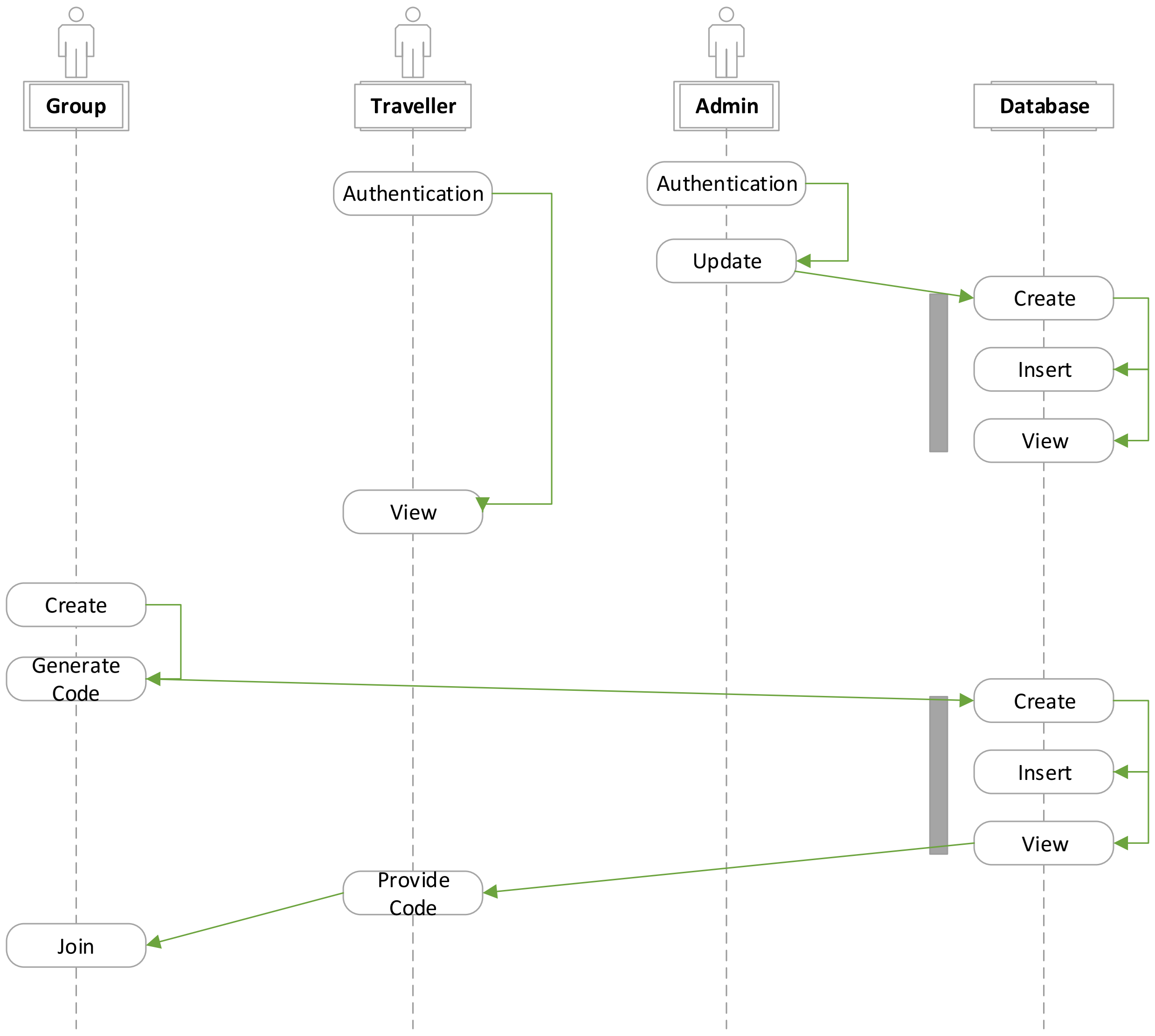
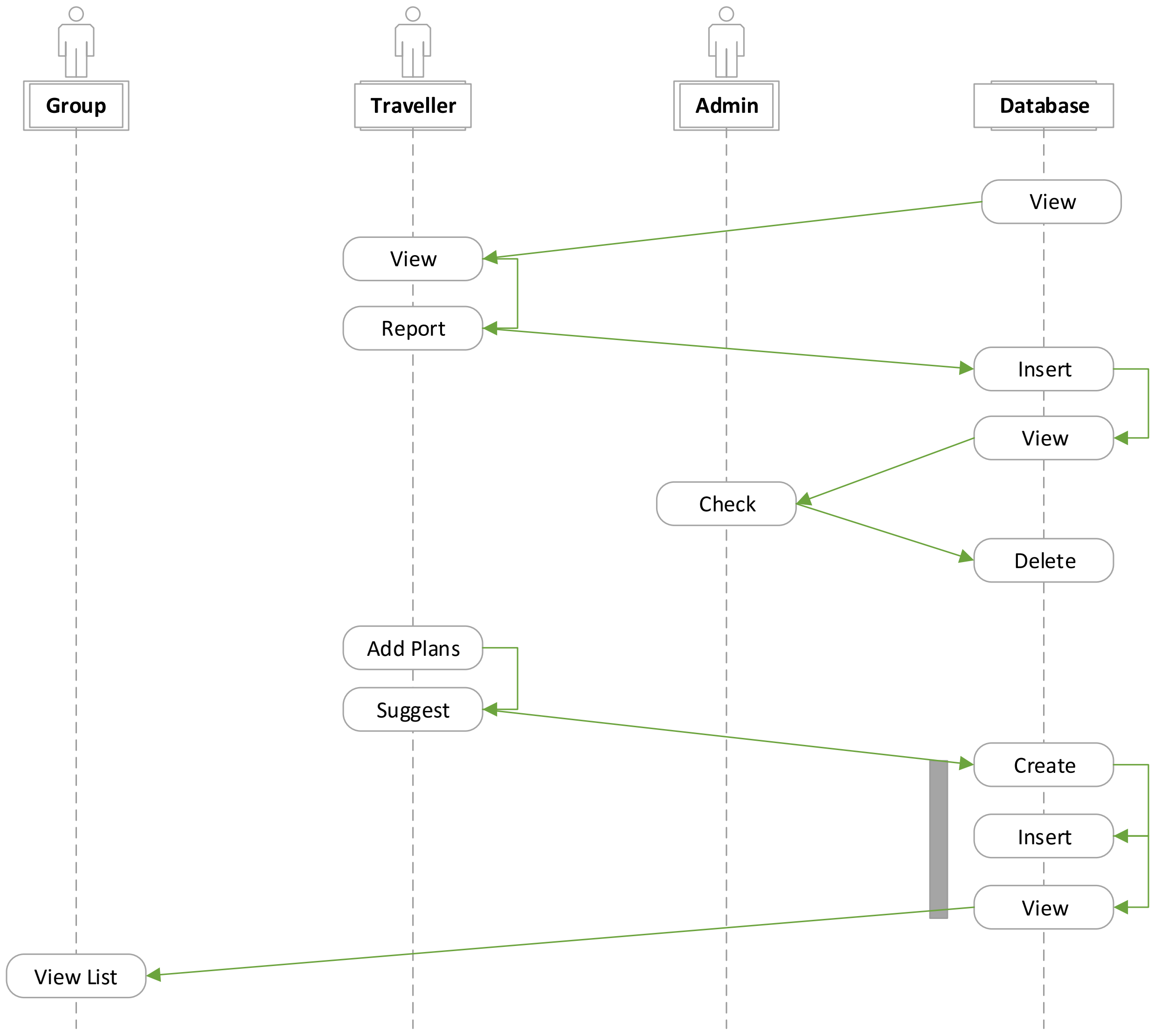


Figure 36: State Transition Diagram (Database class)

Sequence Diagram

Sequence diagram indicates how events cause transitions from object to object.




Group
Traveller
Admin
Database
View
View
Report
Insert
View
Check
Delete
Add Plans
Suggest
Create
Insert
View
View List

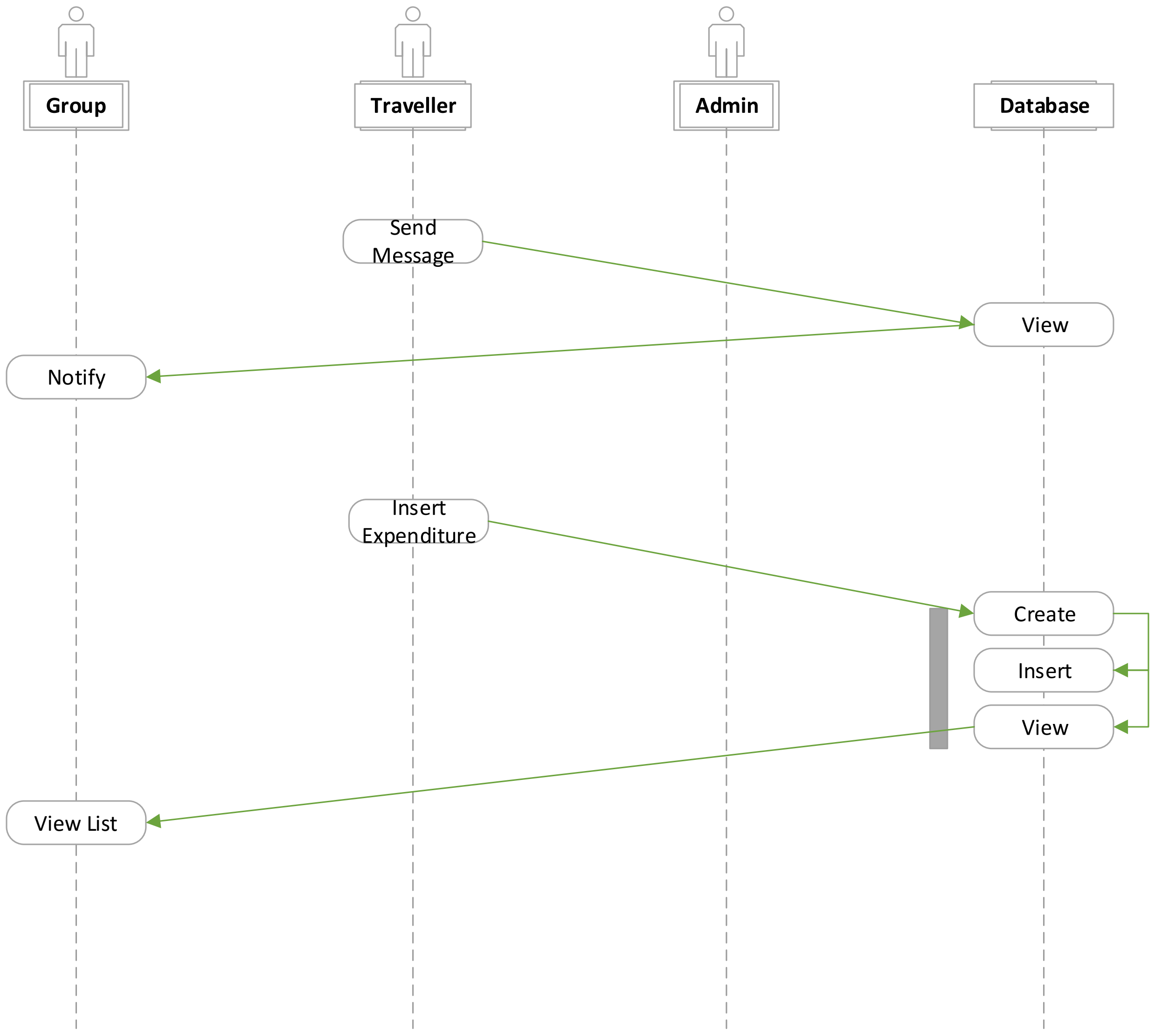


Figure 37: Sequence Diagram

Conclusion

I am pleased to work with software requirements specifications. I have learnt a lot about Software Requirements Specifications. I also have learnt how to do software requirements analysis, how to conduct with the stakeholders. I have illustrated different models with diagrams which will help the developers, software designers and other people associated with it to understand about the system and to do their tasks in a better way. Students as well as teachers may use this document as academic learning resource. I hope that the readers will get benefit from the document.

References

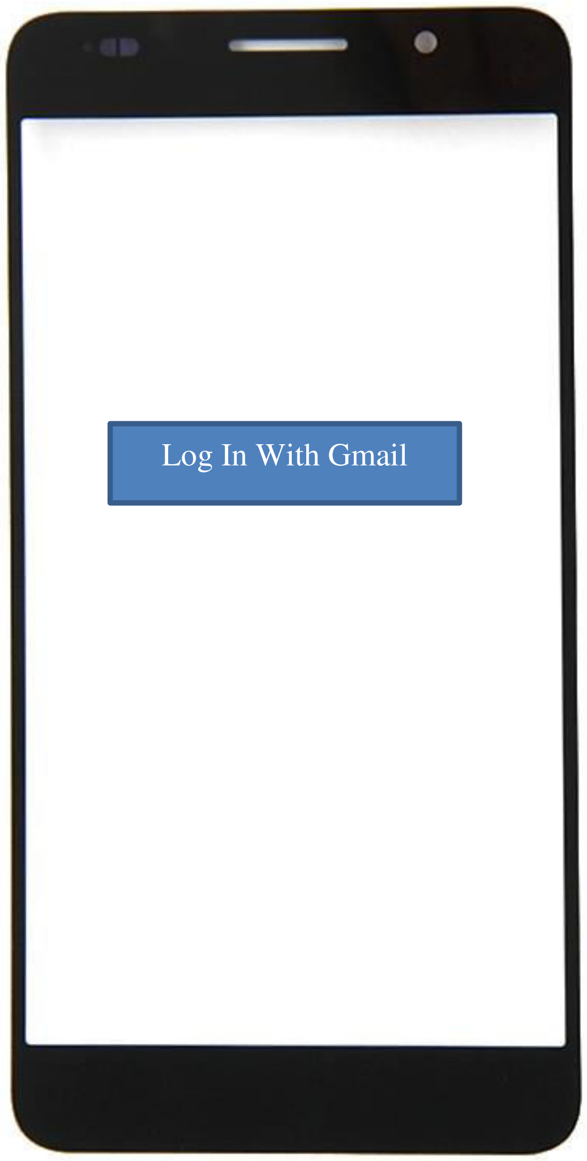
Software Engineering: A Practitioner's Approach

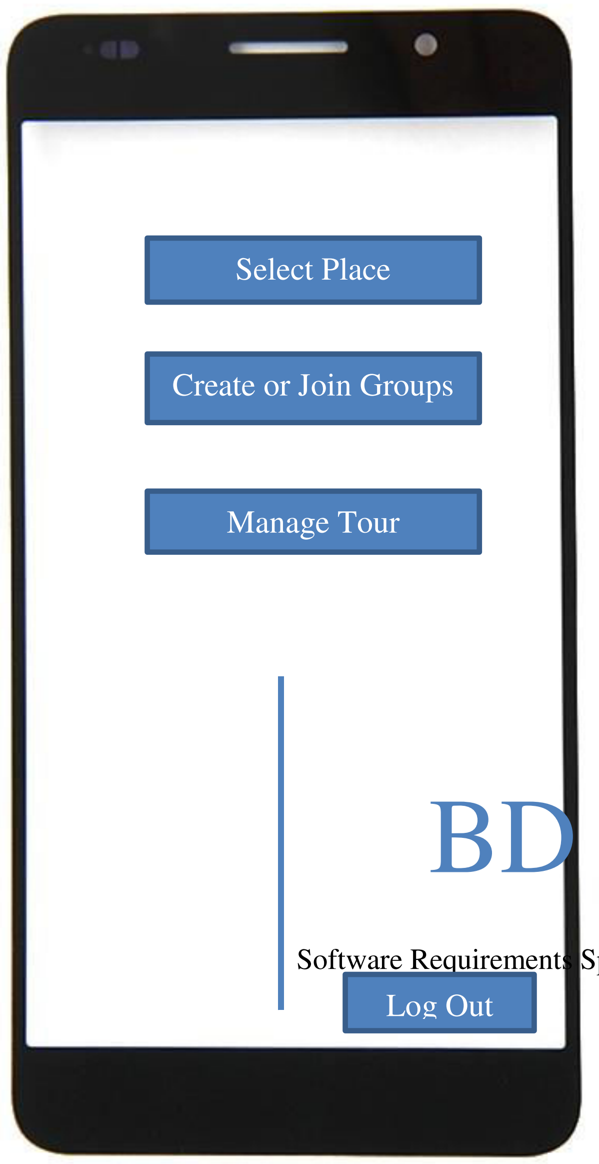
By Roger Pressman

www.draw.io

http://www.tutorialspoint.com/uml/index.htm

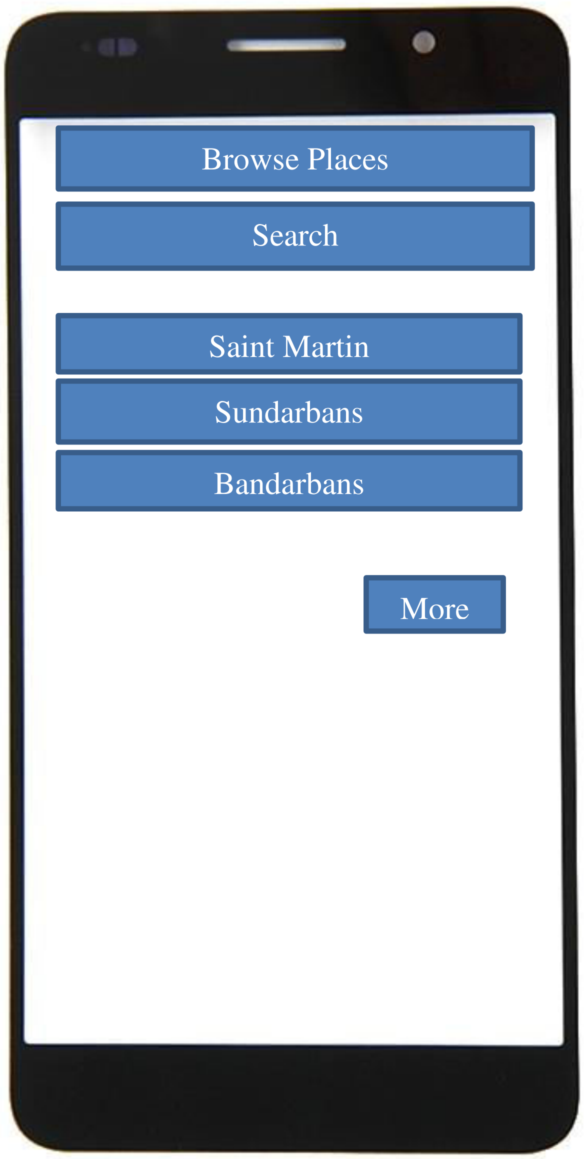
Screenshots:

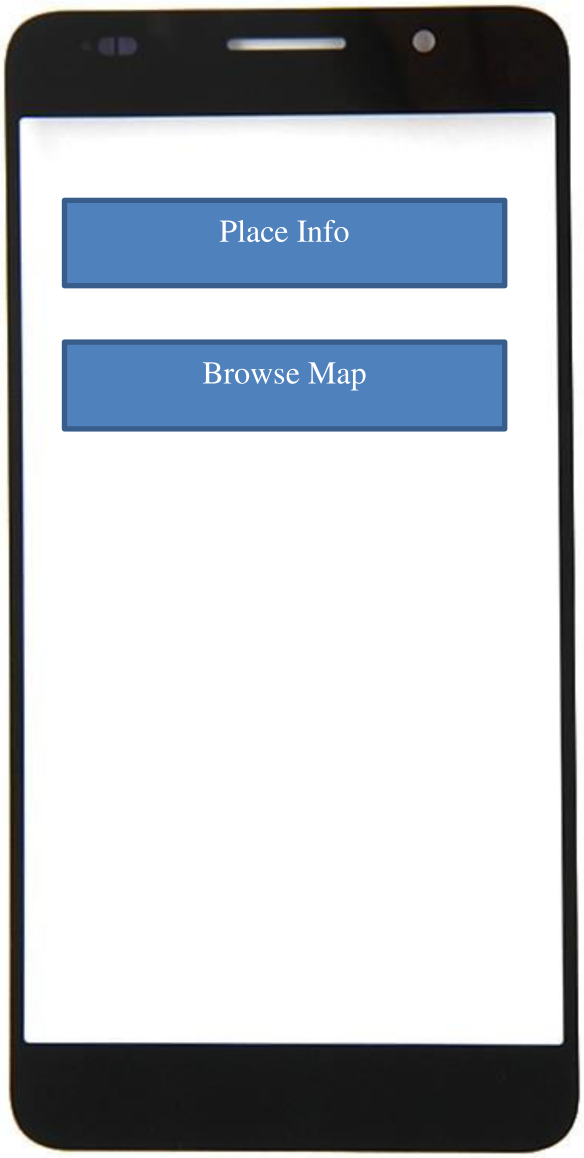


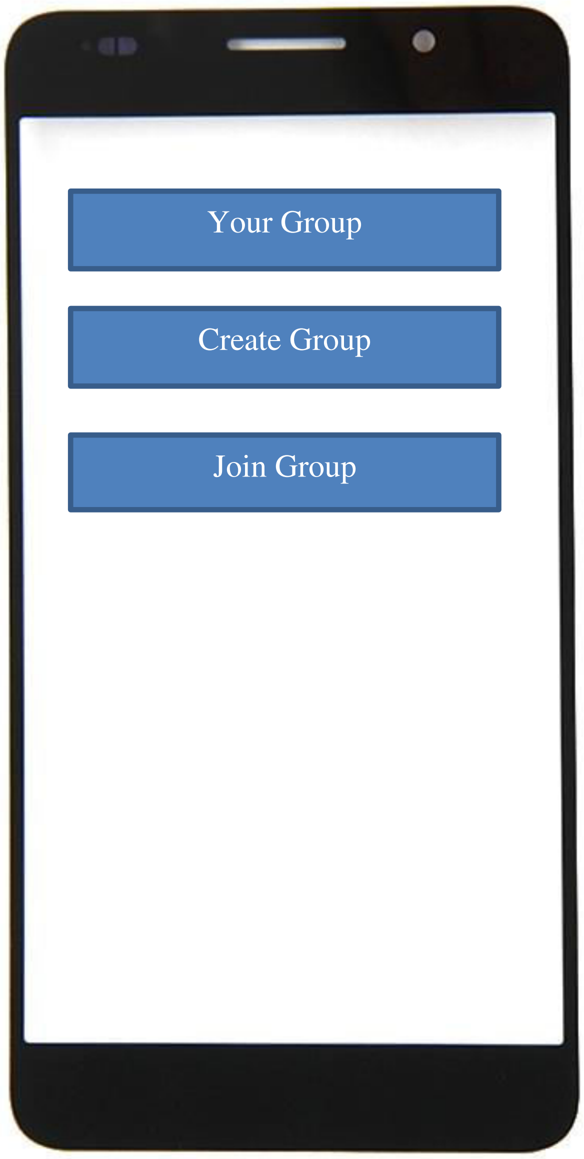


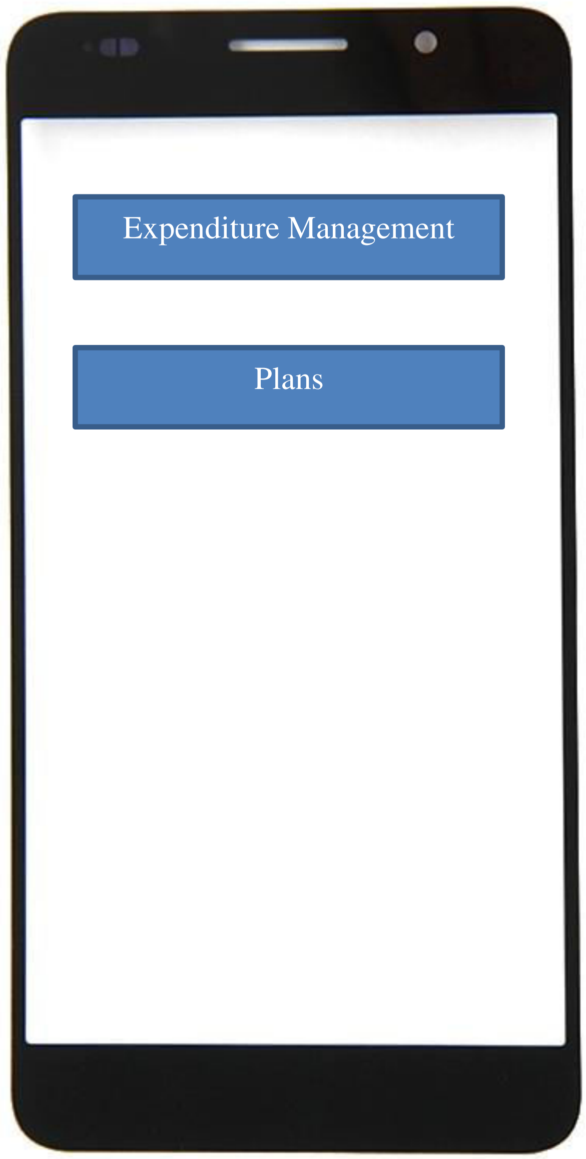
BD Travellers

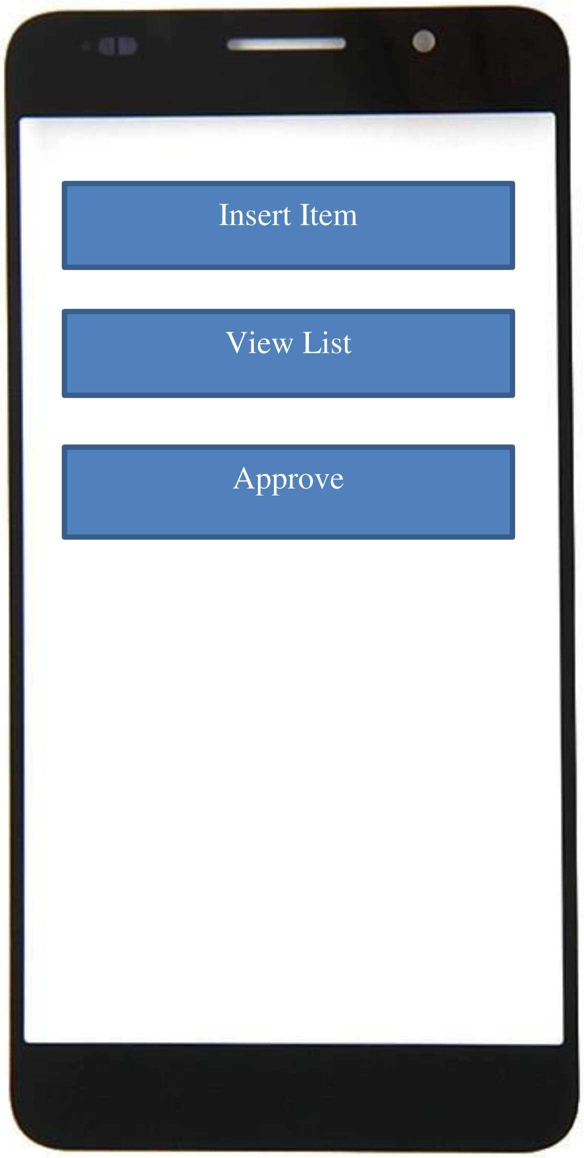
Software Requirements Specifications


Browse Places
Search
Saint Martin
Sundarbans
Bandarbans
More


Place Info
Browse Map


Your Group
Create Group
Join Group


Expenditure Management
Plans


Insert Item
View List
Approve