EASY BUS TRACKER

Software Project Lab II

Easy Bus Tracker

Software Project Lab II

SE 505

Submitted by:

Fazle Mohammed Tawsif

BSS 0628

Supervised by:

Dr. B. M. Mainul Hossain

Assistant Professor

Submitted to:

Sheikh Muhammad Sarwar

Assistant Professor,

Asif Imran

Lecturer

Institute of Information Technology,

University of Dhaka

Submission Date:

13th April, 2016

LETTER OF TRANSMITTAL

April 13, 2016
Asif Imran
Lecturer
Institute of Information Technology
University of Dhaka

Dear Sir,
I have prepared the enclosed report on Software Requirements Specifications of "Easy Bus Tracker" application 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 Project Lab II. This report includes the details of each steps I followed to collect the requirements.

Sincerely Yours,
Fazle Mohammed Tawsif
BSSE 0628

Enclosure: SRS Report

EXECUTIVE SUMMARY

The purpose of Easy Bus Tracker application is to provide a convenient, easy-to-use, Internet-based application for both bus service provider and regular public transport user. Easy Bus Tracker Application will provide tracking of a bus according to the user's choice. Besides tracking, it will also provide the route of a bus and stoppages on that route. However, this application will also show the arrival time to the stoppages from the current location of bus. On the other hand, Bus service providers will use this application to provide updates of their bus to the customers. They will use this application for tracking the current location and current status of the bus according to the traffic condition. This report provides the Software Requirements Specifications (SRS) to develop the system.

ACKNOWLEDGEMENT

By the Grace of ALMIGHTY ALLAH, I have completed my Report on Software Requirements Specification of Automated Student Profile System as a part of Software Project Lab II.

I am very grateful to my project supervisor Dr. B. M Mainul Hossain Sir for his supervision throughout in working time. He helped me a lot by sharing his valuable knowledge throughout the project.

Contents

Index of Figures

Index of Tables

Chapter 1
Introduction

This chapter describes the objectives of this report as well as the audiences who should have to go through this report for individual purposes.

1.1 Purpose

This document is based on the Software Requirement Analysis (SRS) for Easy Bus Tracker, an android application. It includes all necessary requirements to develop this application no matter whether they are functional or non-functional. The information about the requirements here have been organized systematically so that everyone can easily figure out a summarized concept about Easy Bus Tracker. This SRS serves as the official means of communicating user requirements to the developer and provides a common reference point for both the developer and stakeholder community. It will evolve over time as users and developers work together to validate, clarify and expand its contents.

1.2 Intended Audience

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

- The customer will use this document to ensure that whatever he requires has been fulfilled by the project teams.
- The project managers of the developer team will use this SRS to fix a milestone and time to deliver the software and to ensure that the teams working on this project are on the right path.
- The designers will use this document 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 report 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.

Chapter 2

Inception

In this chapter, I will discuss about the first step of Software Requirements Specifications Analysis, that is, Inception.

2.1 Introduction

Requirement Engineering comprises several sequential steps. Inception is the first one among them. Inception creates the entrance to the project for the requirements analysts. It refers them how the project should get started. It also provides a basic idea to the engineers about the problems ahead which are needed to be solved and how critical obstacles may come during the project. The main target of Inception phase is to identify the people related to the project and their needs. In order to complete this phase, I have focused on---

- Identifying Stakeholders
- Recognizing multiple view points
- Working towards collaboration
- Asking the first questions

2.1.1 Identifying stakeholders

Stakeholders are entities that have an interest in a given project. These stakeholders may be inside or outside an organization which:

- Sponsor a project, or
- Have an interest or a gain upon a successful completion of a project,
- May have a positive or negative influence in the project completion

General Users: General users are the main stakeholders of this Easy Bus Tracker application. People who use bus as a transport service will use this application. This application will provide continuous update of bus that they use for transportation.

Bus Service Provider: Bus service providers will use this application to provide information about their bus to general users. Using this application, they will provide current status of the bus, location, regular route, stoppages to general users.

2.1.2 Recognizing multiple viewpoints

After discussing with the customers: general users and bus service providers, they provide their view points about the application.

General user's viewpoints:

- Easily accessible
- Want accurate status of the bus
- On service route
- Regular stoppages
- Arrival time to stoppages
- Automated nearest stoppage detection
- Notification for a favorite route

Bus service provider's viewpoints:

- Providing location of bus to the customers
- Latest routes are automatically updateable
- Current status of the bus along with traffic condition
- Detecting on service route of a bus from multiple route according to the time

2.1.3 Working towards collaboration

Different stakeholders see things according to their own view points. As a result, many viewpoints may be common as well as many of them may conflict. I followed the following steps to merge the requirements:

- Identify the common and conflicting requirements
- Categorize the requirements
- Take priority points for each requirements from stakeholders and on the basis of this voting prioritize the requirements
- Make the final requirements

Common requirements:

I've found the following requirements which are required by all the stakeholders:

1. Current location of the bus
2. Route of a bus and Stoppages
3. On service route
4. Current status of the bus

Conflicting requirements

I have also found some requirements those are conflicting each other's:

1. Detecting on service route from multiple route
2. Arrival time to stoppages from multiple route

Final Requirements:

We select the final requirements for the system by categorizing and prioritizing them. The finalized requirements are:

1. Easily accessible to the map
2. Current status of the bus
3. On service route
4. Regular stoppages
5. Arrival time to stoppages
6. Automated nearest stoppage detection
7. Notification for a favorite route
8. Detecting on service route
9. Updating the latest route

2.1.4 Asking first questions

I set our first set of context-free questions focuses on the customer and other stakeholders, overall project goals and benefits. These questions helped me to identify all stakeholders, measurable benefit of the successful implementation and possible alternatives to custom software development. Next set of question helped me to gain a better understanding of problem and allows the customer to voice his or her perception about the solution. The final set of question focused on the effectiveness of the communication activity itself.

2.2 Conclusion

Inception phase helped me to establish basic understanding about Easy Bus Tracker application, identify the people who will be ben**e**fited if the application is implemented, define the nature of the Easy Bus Tracker application and establish a preliminary communication with our stakeholders.

Chapter 3
Elicitation

In this chapter, I will briefly discuss about the Elicitation phase of my Easy Bus Tracker application.

3.1 Introduction

Requirements elicitation is recognized as one of the most critical, knowledge-intensive activities of software development; poor execution of elicitation will almost guarantee that the final project is a complete failure. Since project failures are so rampant, it is quite likely that improving how the industry performs elicitation could have a dramatic effect on the success record of the industry. Improving requirements elicitation requires us to first understand it. Although many papers have been written that define elicitation, or prescribe a specific technique to perform during elicitation, nobody has yet defined a unified model of the elicitation process that emphasizes the role of knowledge.

3.2 Eliciting requirements

Earlier we have seen that the methodology used in Inception phase is Question and Answer approach. But Elicitation is quite different in this point of view. The elicitation phase follows a format of eliciting requirements which combines the other four phases namely problem solving, elaboration, negotiation and specification. In order to elicit requirements, I have followed four steps:

- Collaborative Requirements gathering
- Quality Function Deployment (QFD)
- Usage Scenarios
- Elicitation work product

3.3 Collaborative requirements gathering

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

1. The meetings were conducted with the bus service provider administrator of "University of Dhaka" and several general users, mainly students who uses university bus for transportation. They were questioned about their requirements and expectations from the "Easy Bus Tracker" application.
2. They were asked about the problems they are facing with the current manual system.
3. At last I have selected final requirement list from the meetings.

3.4 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.

3.4.1 Normal requirements

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

1. Accessible via the Internet
2. Allow System to check items for valid users.
3. Allow valid users (Bus service provider) to login and logout.
4. Restrict access to tracker from general users.
5. Help feature to explain what they are looking for.
6. A product reference manual describing how to use this automated software.

3.4.2 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

1. Application will run in any Android device
2. User can easily view the map
3. Maintain a route list
4. Provide route in the map
5. Show stoppages of a route
6. View current location of the bus
7. Transfer continuous GPS trace to database
8. Get the status of the bus

3.4.3 Exciting requirements

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

1. Update the recent route of a bus automatically
2. Detect the on service route of a bus from multiple routes
3. Notification system for general users
4. Automatically detection of nearest bus stoppages

3.5 User scenario

Now-a-days Traffic has become major issue, people have to wait hours and hours to catch a bus. Having the current status of bus and arrival time estimation to current location, will save a lots of time. This system will help users to know the route, stoppages, current location of bus, arrival time prediction to current location of a bus from a particular bus service.

Easy Bus Tracker system consists of two types of users. They are: general user and bus service provider. General user who just views the map and gets the current status of bus. All smartphone users who will use this system are considered as general user. On the other hand, bus service providers use this system to provide service to general users.

USER TYPE: GENERAL USER

General user views the route, stoppages and current position of a bus in the map. General user doesn't need to sign up or log in for using the service. General user also able to get the status of a bus in a particular route. General user gets the on service route and stoppages of that bus along with bus information. However, Arrival time from current location of the bus to any stoppages will also available to user. General user selects a route as a favorite route. If he/she comes near to any stoppage of the selected route, user is notified. User sets a time interval for getting the notification.

ARRIVAL TIME:

Arrival time of the bus to the stoppage of user's nearby location is provided in the map view. Map view also contains arrival time to other stoppages in a route. General user selects other stoppage to view the arrival time of selected stoppage.

USER TYPE: BUS SERVICE PROVIDER

Bus service provider needs to sign up to use the Easy Bust Tracker System. Bus service provider signs up using Third party authentication system. After signing up, the bus will be added into a particular route. Then general users can view information of that bus. Registered bus only uses the location service provider. Bus service provider logs in their account using appropriate credentials to start location service provider.

LOCATION SERVICE PROVIDER

This service provides the current position of the bus to the general users. Using regular location traces of a bus, location service provider extracts the route and stoppages of the bus and provide them to General users. It also gives the current status of the bus.

3.6 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:

- Set of usage scenarios.
- Description of the system's technical environment.
- Make a bounded statement of scope for my system.
- Make a list of user and other stakeholder who participated in requirements.
- Make a statement of the requirements for Easy Bus Tracker application.

Chapter 4
Scenario Based Model

This chapter is about the scenario based model of Easy Bus Tracker application.

4.1 Introduction

Scenario based modelling is an inexpensive rapid prototyping technique. This method is effective when systems are being built with the requirements vaguely known at the outset. Users are involved right from the start, to build prototypes evolving towards the final product. The users are also involved with the testing of the prototypes which is essential for the validation of requirements and help the users to gain an initial experience of the final system during the development itself. This method involves techniques which are applied by one or more professionals working alongside users who are expected to provide and specify their requirements at the beginning as well as evaluate and approve the system upon completion. The user (in a passive capacity) and the designer/builder (an active partner) cooperate to reach a working model where the means of communications are by the examination of preliminary models such as the initial narratives, paper models and graphical representations built to represent the final system functions.

Primary Actor:

Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software.

Secondary Actor:

Secondary actors support the system so that primary actors can do their work. They either produce or consume information.

4.2 Use case diagrams

Use case diagrams give the non-technical view of overall system.

There are 3 subsystems in the Easy Bus Tracker application. They are

1) Location Provider Service,

2) Display Location and

3) Notification.

These subsystems have different tasks. The use case diagram of each subsystem and their description will be discussed in this section. The following table summarizes the subsystems and the actor of each subsystem.

Level 0	Level 1	Level 2	Actor
Easy Bus Tracker	Location Provider Service	Bus current position	Bus service provider
		Provide status	Bus service provider
		Extract route	-
		Extract stoppages	-
	Display Location	Select route, stoppage, bus	Bus service provider
		Show arrival time	Location provider
	Notification	Set route & interval	User
		Stat notification	User

Table 1: Use case scenario of Easy Bus Tracker

4.2.1 System Description from Level-0 use case:

After analyzing the user story, I have found two actors who will directly use the system as a system operator. Primary actors are those who will play action and get a reply from the system whereas secondary actors only produce or consume information.

Following are the actors of Easy Bus Tracker application –

1. General users
2. Bus service providers

4.2.1.1 Level-0 Use Case Diagram

In this level of use case diagram describes the overall system and the actors interacting with the system. Here in our system I have three actors interacting with the system.

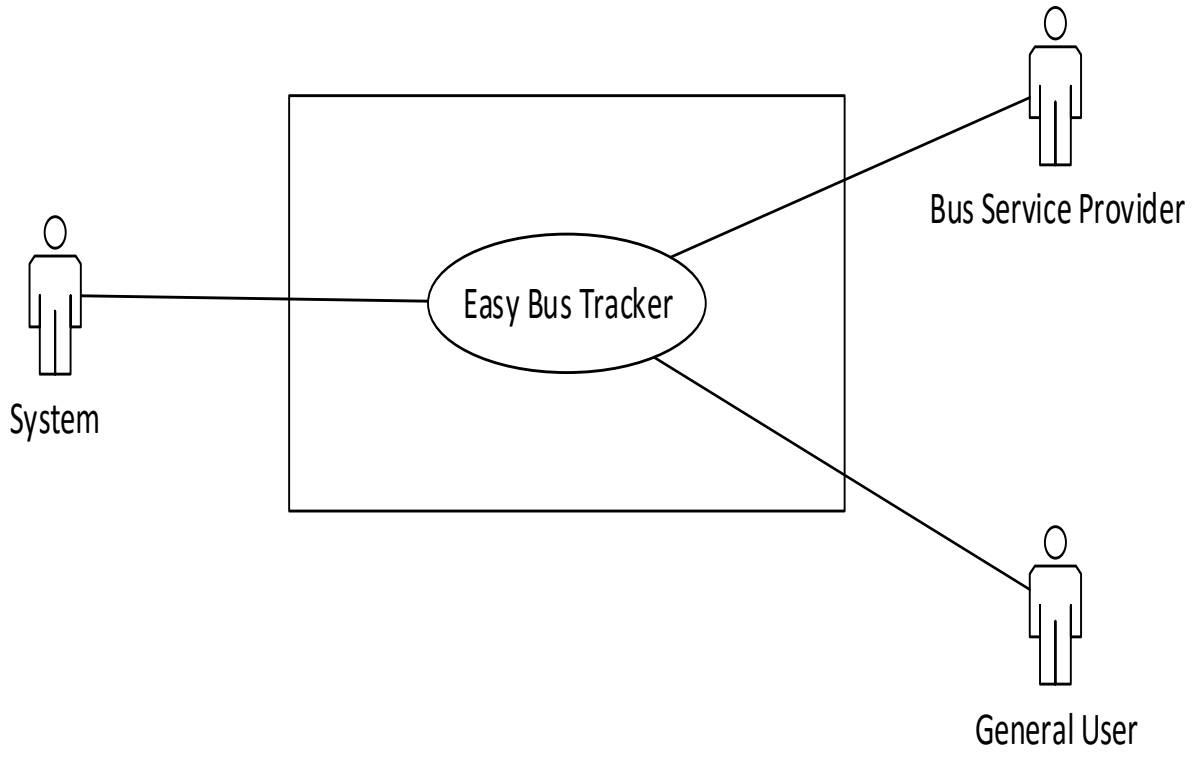

Figure 1: Use case Level 0 of Easy Bus Tracker System

4.2.1.2 Level-1 Use Case Diagram

The actors of Easy Bus Tracker application play different actions and system will reply according to these actions –

General User:

Action1: Set Notification

Reply1: Select favorite route and set interval

Action2: View map

Reply2: Select route and busses on that route

Action3: Show arrival time

Reply3: View the arrival time to the stoppages from bus location in the map

Action4: Show my position

Reply4: View user's position in the map

Bus service provider:

Action1: Register

Reply1: Select a third party authentication

Action2: Start tracker

Reply2: Tracker for the bus started tracking

Action3: Provide Bus information

Reply3: Enter bus id, name, number and other description

4.2.2 Use case diagram and the description of subsystems

This section will discuss about the subsystems found as well as the role of each subsystem on the whole system. Let's first see how the whole systems can be divided into subsystems by the following use case diagram:

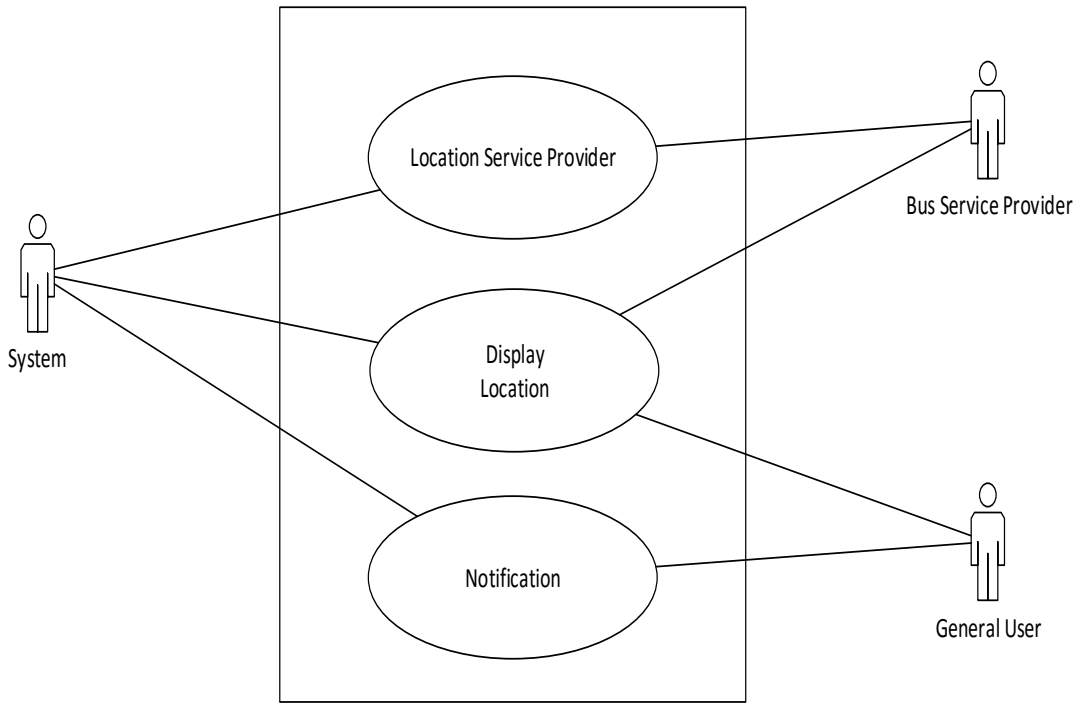

Figure 2: Use Case Level 1 Diagram Easy Bus Tracker System

4.2.2.1 Location Service Provider

This subsystem performs four tasks; they are:

1. Providing Bus current position
2. Providing Status
3. Extracting Route
4. Extracting stoppages

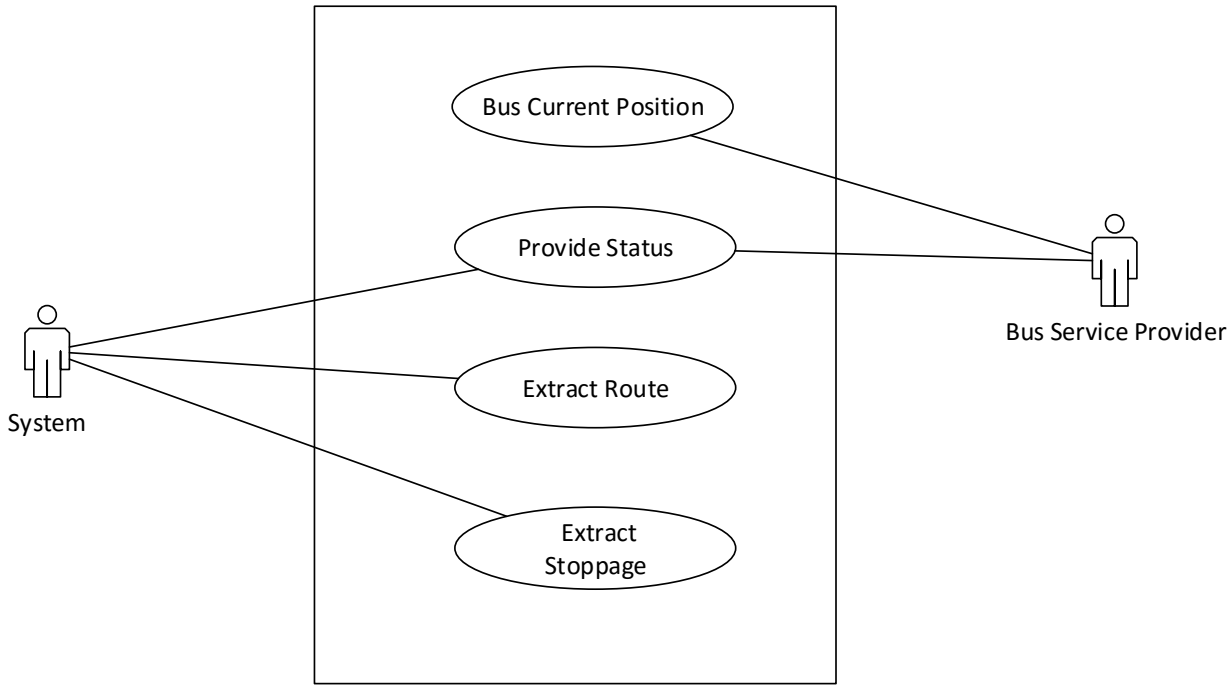

Figure 3: Use Case Diagram Level 1.1 Location service provider subsystem

4.2.2.2 Display Location

This subsystem shows the map and other attributes to the users. User selects route, stoppages, bus and view arrival time to the users. Location provider service interact with the display location subsystem to provide calculated arrival time to Display location subsystem.

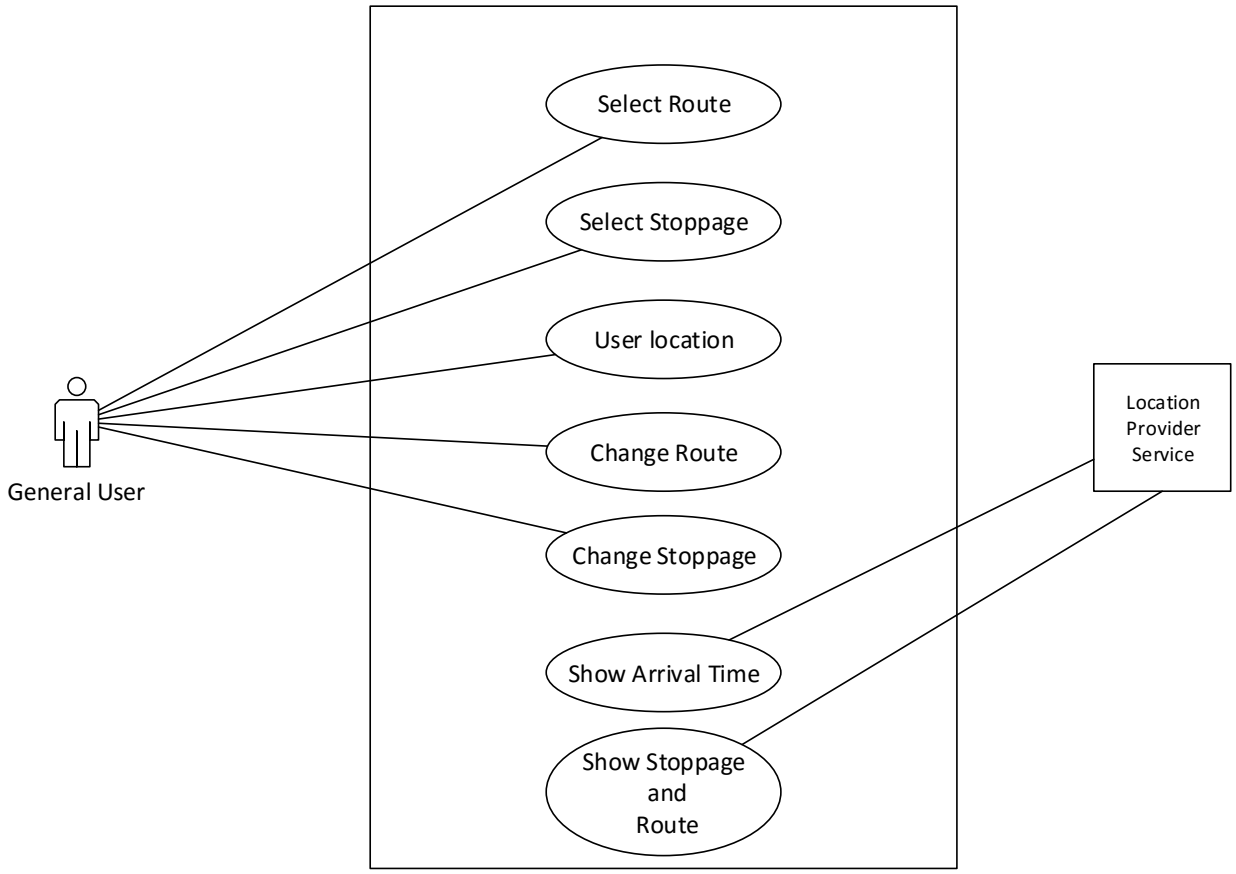

Figure 4: Use Case Diagram Level 1.2 Display Location Subsystem

4.2.2.2.1 Subsystems of Showing arrival time:

Show Arrival time calculate the arrival time to the stoppages from current location of the bus. It also uses the location service provider subsystem to collect the current location of bus.

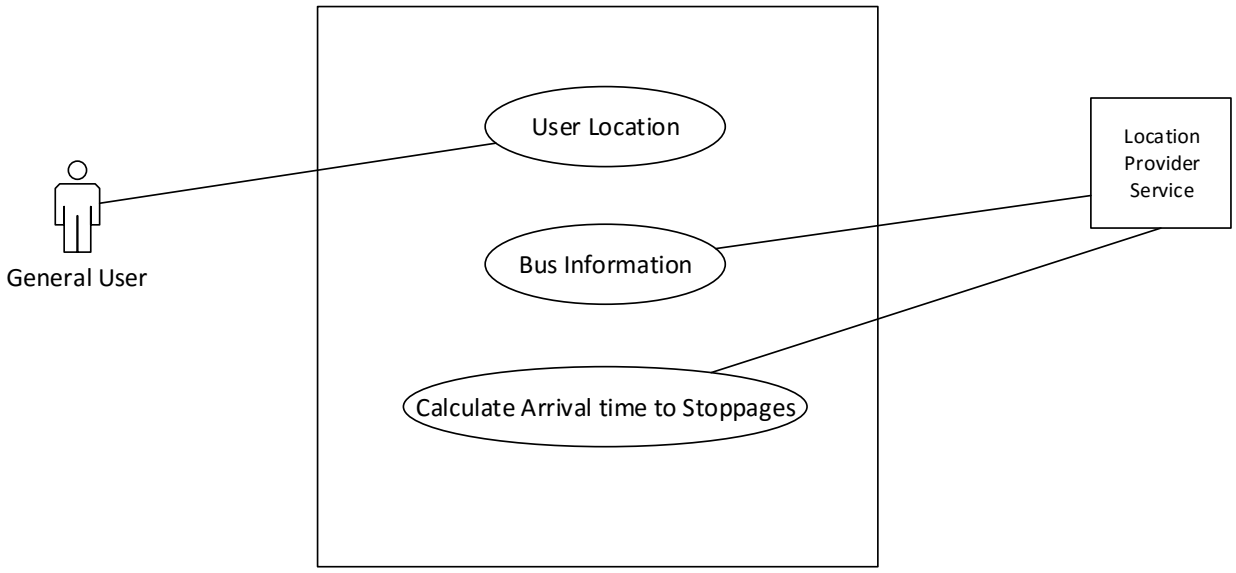

Figure 5: Use Case Diagram Level 1.2.1 Show arrival time Subsystem

4.2.2.3 Notification

This the final subsystem of the application. It provides notification to users. Users select their favorite route and also sets a time interval for getting the notification. This subsystem notifies the users if they come near to any stoppage of their selected route.

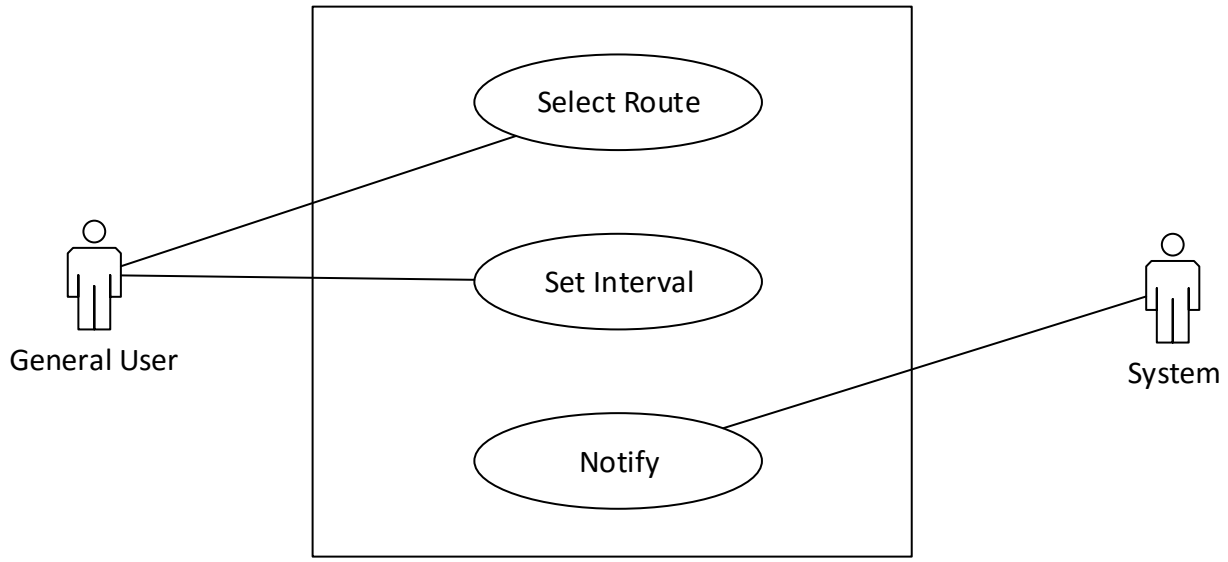

Figure 6: Use Case Diagram Level 1.3 Notification

4.3 Activity diagram and swim lane diagram

Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

A swim lane (also known as swimlane) diagram is a type of flowchart. Like a flowchart, it diagrams a process from start to finish, but it also divides these steps into categories to help distinguish which departments or employees are responsible for each set of actions.

These lanes are columns that keep actions visually separated from others. A swim lane diagram makes responsibilities more clear than a regular flowchart. When looking to improve processes, knowing which department is responsible for what can help speed up the process of correcting inefficiencies and eliminating delays.

In this section I will show the activity and swimlane diagrams for each use case of the Easy Bus Tracker application.

Use case Level 1: Easy Bus Tracker system

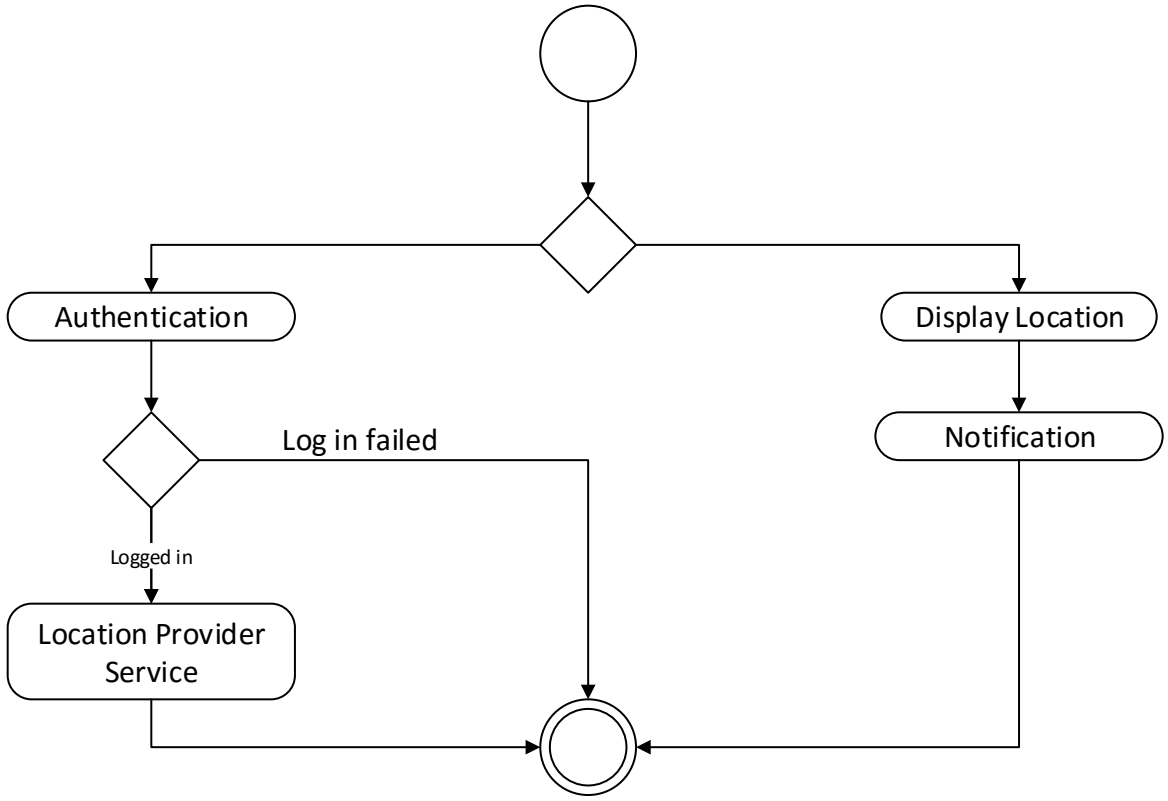

Figure 7: Activity Diagram for Easy Bus Tracker system

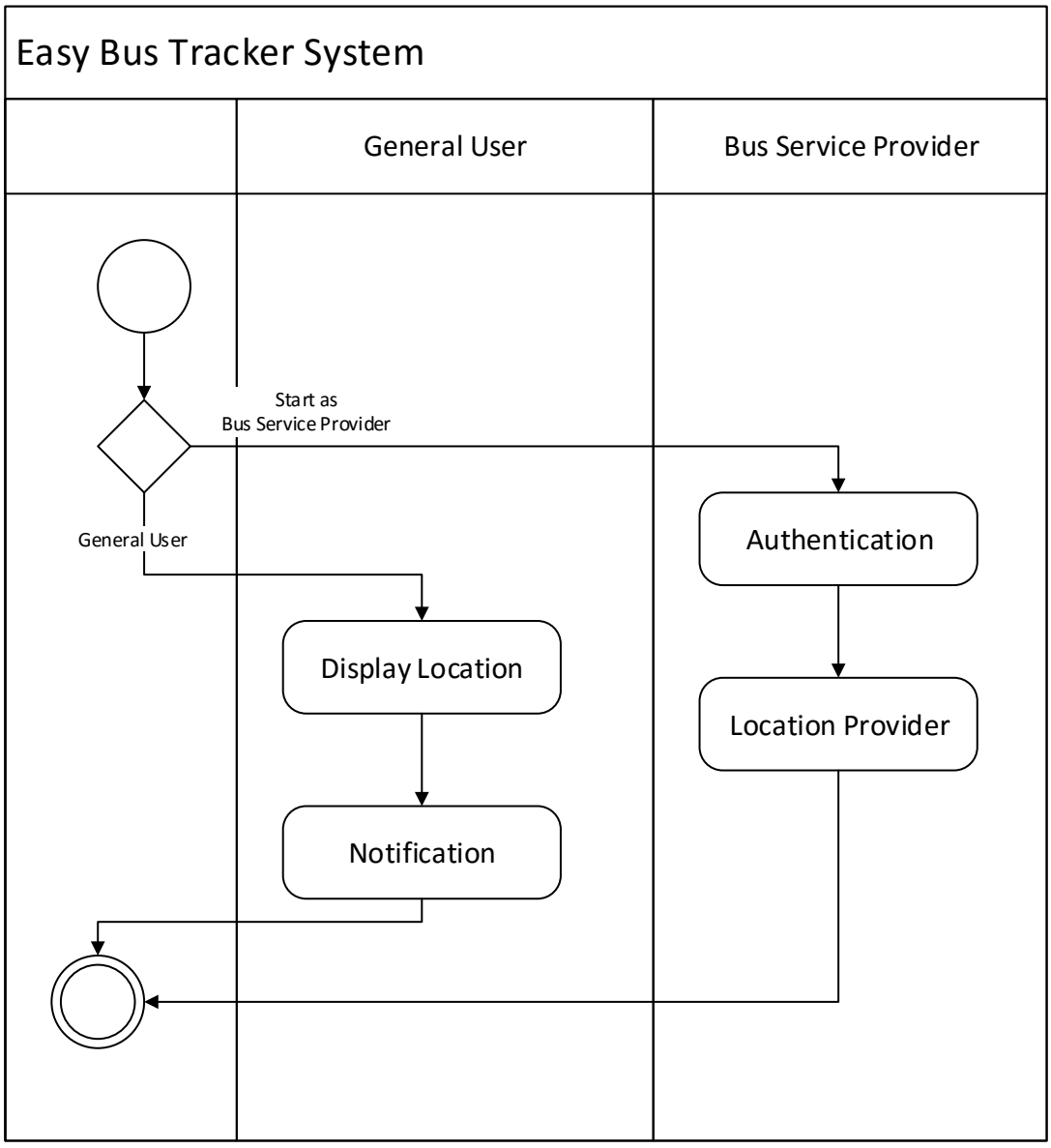

Figure 8: Swim lane diagram for Easy Bus Tracker System

Use case Level 1.1: Location Service Provider

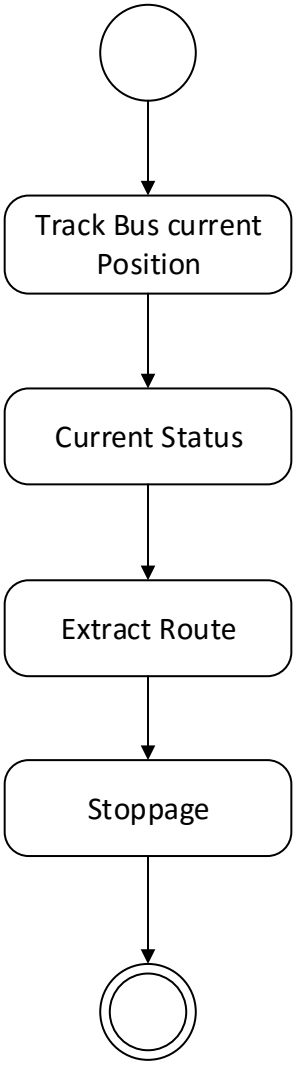

Figure 9: Activity Diagram for Location provider service use case

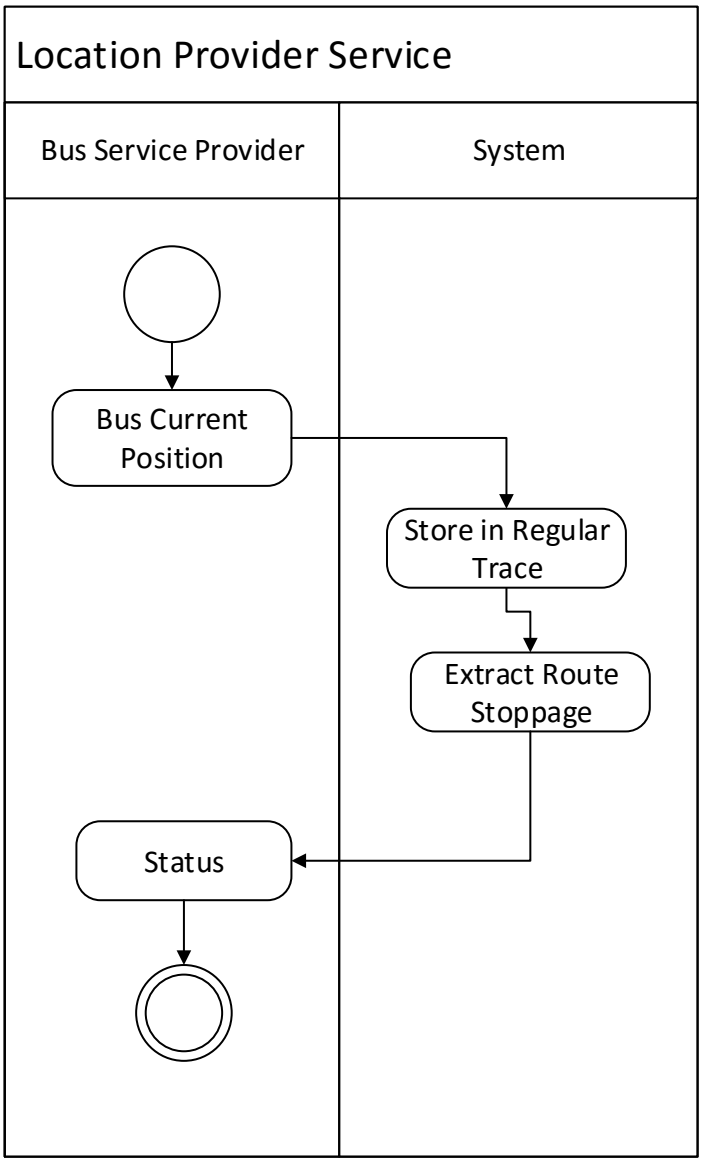

Figure 10: Swim lane diagram for Location Provider Service

Use case Level 1.2: Display Location

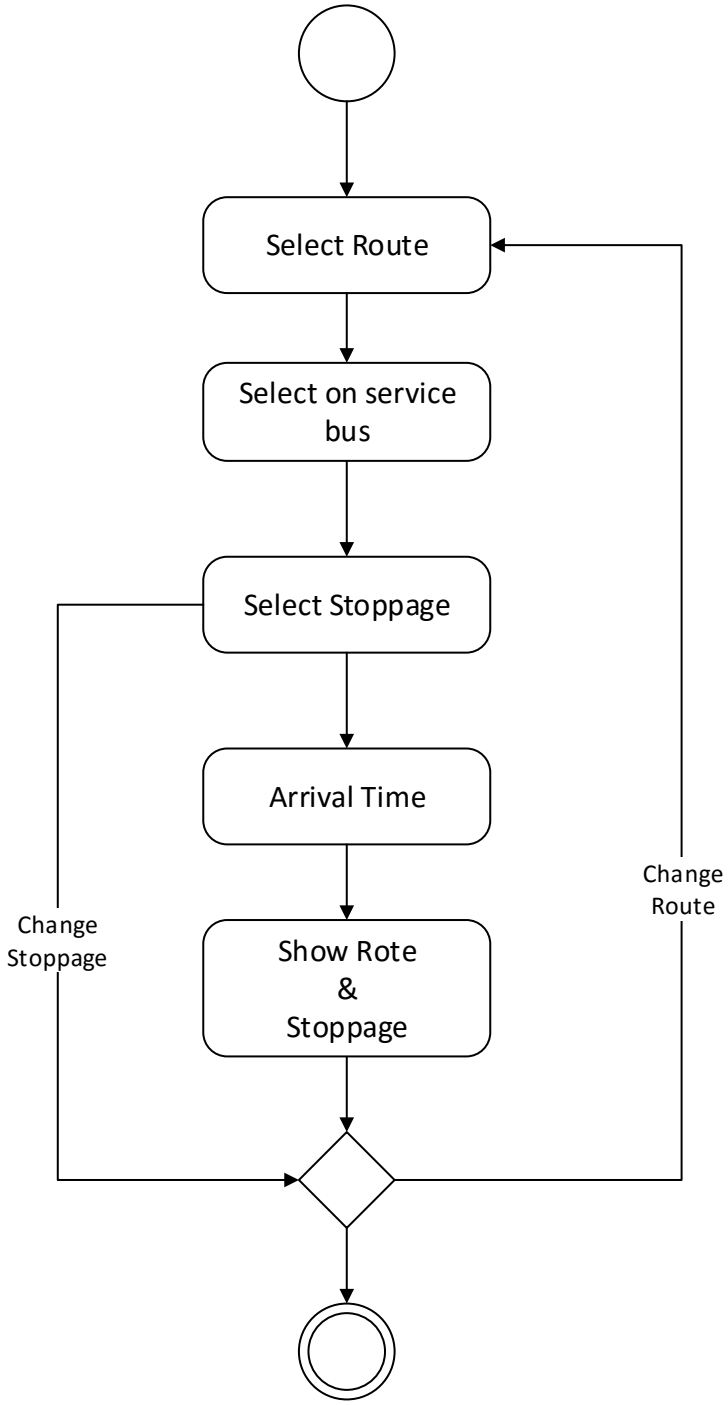

Figure 11: Activity Diagram for Use Case Display Location

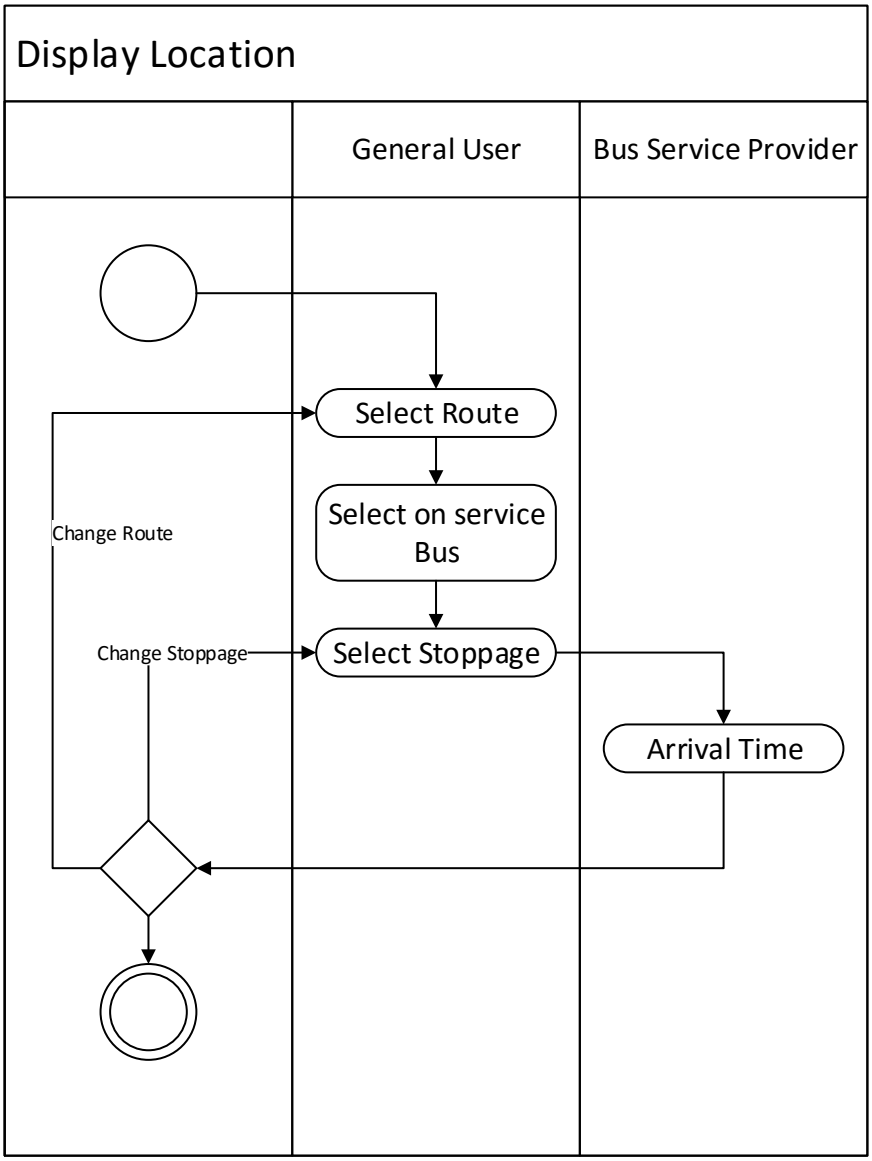

Figure 12: Swim lane Diagram for Use Case Display Location

Use case Level 1.2.1: Show Arrival Time

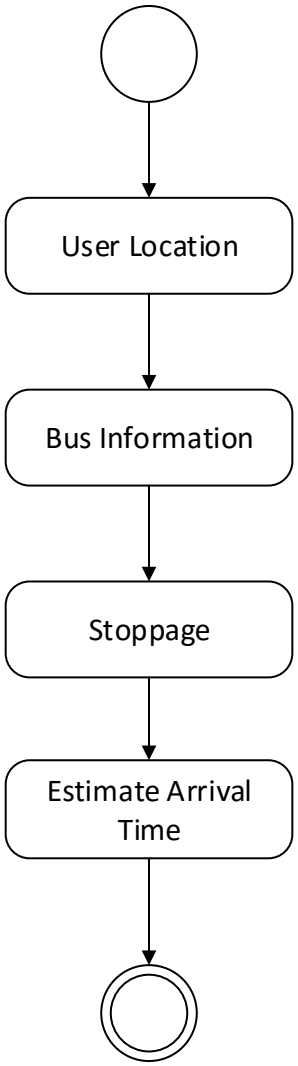

Figure 13: Activity Diagram for Use Case Show Arrival Time

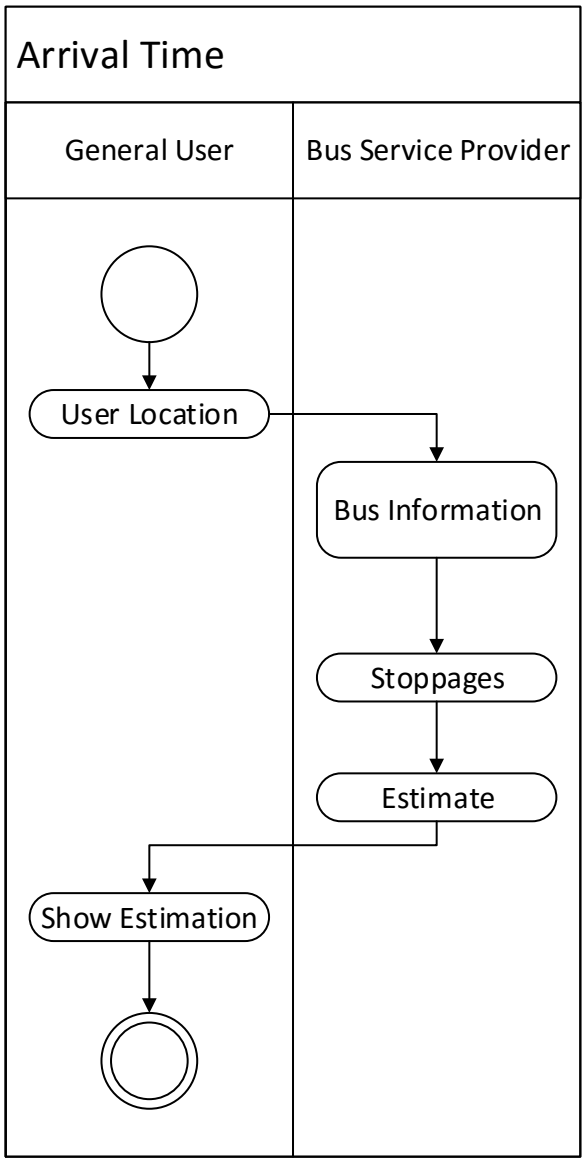

Figure 14: Swim lane Diagram for Use Case Show Arrival Time

Use case Level 1.3: Notification

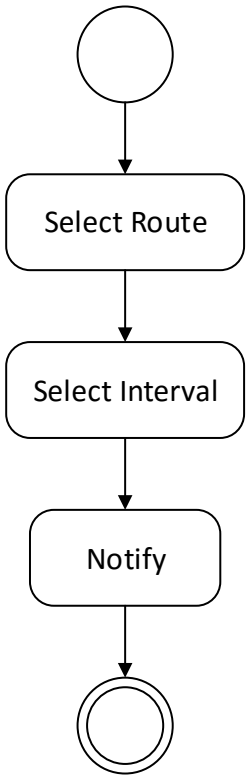

Figure 15: Activity Diagram for Use Case Notification

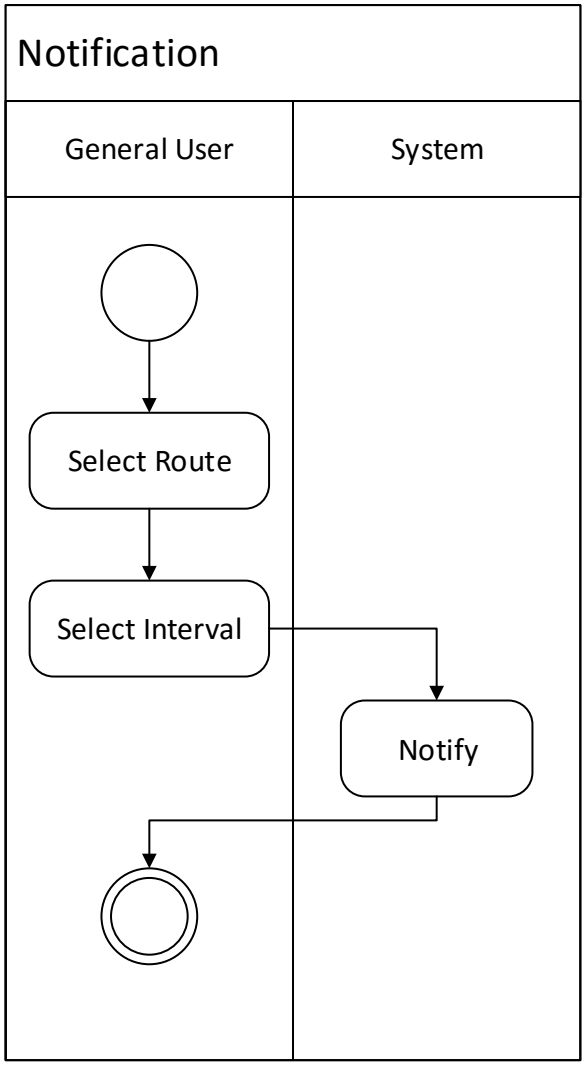

Figure 16: Swim lane Diagram for Use Case Notification

Chapter 5
Data Model

5.1 Introduction

A data model is a conceptual representation of data structures (tables) required for a database and is very powerful in expressing and communicating the business requirements. In this chapter, I will discuss about the data model for Easy Bus Tracker application.

5.2 Data object

A data object is a representation of information which has different properties or attributes that must be understood by software. I have found the following data objects in my Easy Bus Tracker application:

- **Bus Service Provider:**

  Attributes:

  - ✓ Bus ID
  - ✓ Bus name

- **Trace:**

  Attributes:

  - ✓ GPS Trace ID
  - ✓ Latitude
  - ✓ Longitude
  - ✓ Time & Date

- **Route:**

  Attributes:

- ✓ Route ID

- **Stoppages:**

  Attributes:

  - ✓ Stoppage ID
  - ✓ Stoppage Name
  - ✓ Stoppage Latitude
  - ✓ Stoppage Longitude

- **User:**

  Attributes:

  - ✓ User ID
  - ✓ Favorite Route
  - ✓ Current Location

5.3 ER diagram

Based on the data objects found, in order to show the relationship among the data objects, ER diagram that is Entity-Relationship Diagram is used widely. Here, the ER Diagram based on the data objects of Easy Bus Tracker application is shown.

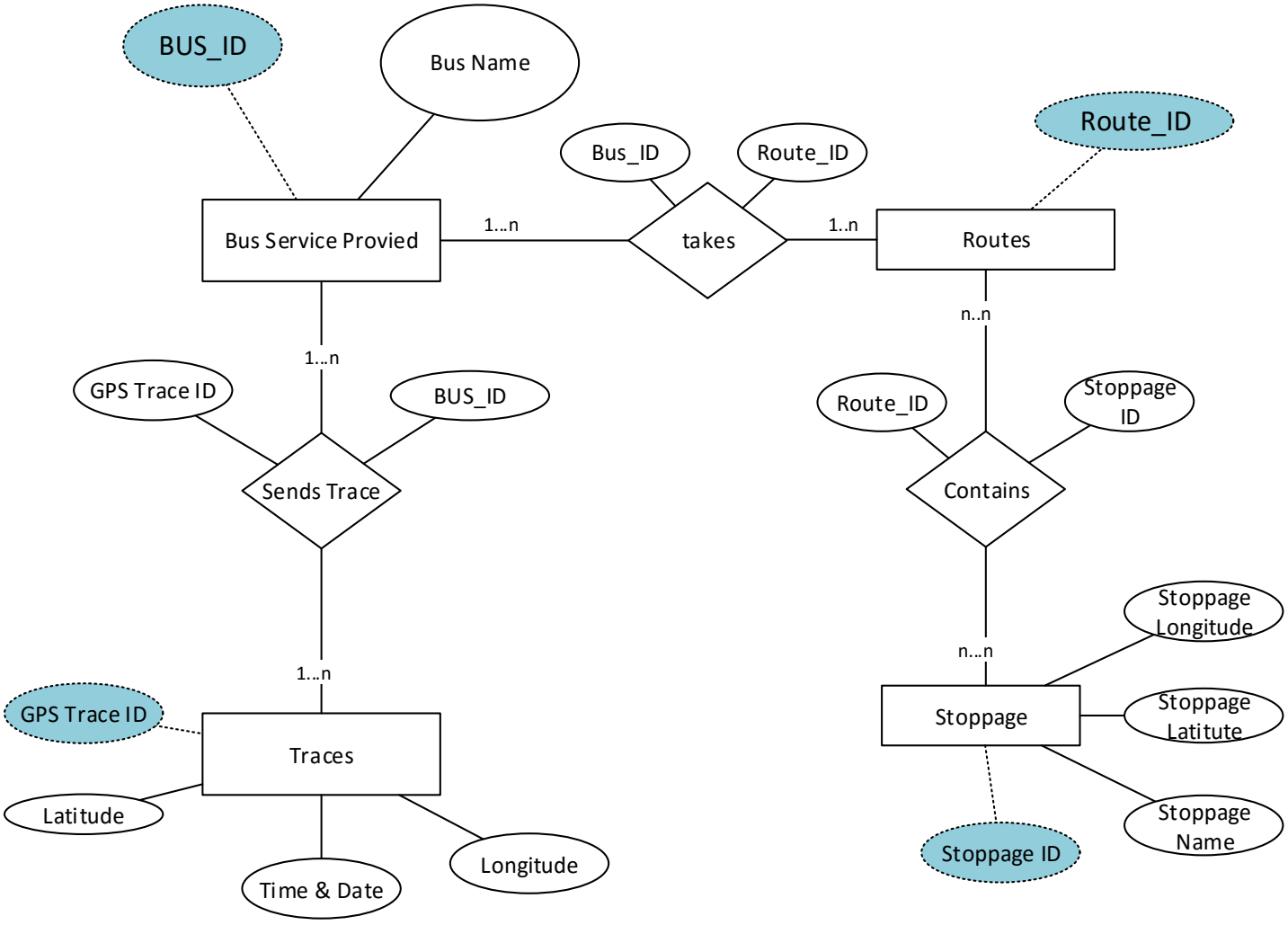

Figure 17: ER Diagram

5.4 Data schema table

Based on the data objects, the following data schema tables can be created:

Bus Service Provider	
Attributes	**Type (Size)**
Bus Name	Varchar(20)
Bus ID	Number (15)

Table 2: Data schema: Bus Service Provider

Stoppages	
Attributes	**Type (Size)**
Stoppage Latitude	Double(20)
Stoppage Longitude	Double(20)
Stoppage ID	Number (10)
Stoppage Name	Varchar(20)

Table 3: Data schema: Stoppages

Traces	
Attributes	**Type (Size)**
Latitude	Double(20)
Longitude	Double(20)
GPS Trace ID	Number (10)
Time & Date	Varchar(20)

Table 4: Data schema: Traces

Route	
Attributes	**Type (Size)**
Route ID	Number (10)

Table 5: Data schema: Route

BusTrace Relation	
Attributes	**Type (Size)**
Bus ID	Number (15)
GPS Trace ID	Number (10)

Table 6: Data schema: BusTrace Relation

BusRoute Relation	
Attributes	**Type (Size)**
Route ID	Number (10)
Bus ID	Number (15)

Table 7: Data schema: BusRoute Relation

RouteStoppages Relation	
Attributes	**Type (Size)**
Stoppage ID	Number (10)
Route ID	Number (10)

Table 8: Data schema: Route Stoppage Relation

5.5 Relationship between data objects

The pair to pair relationship among the data objects are shown below:

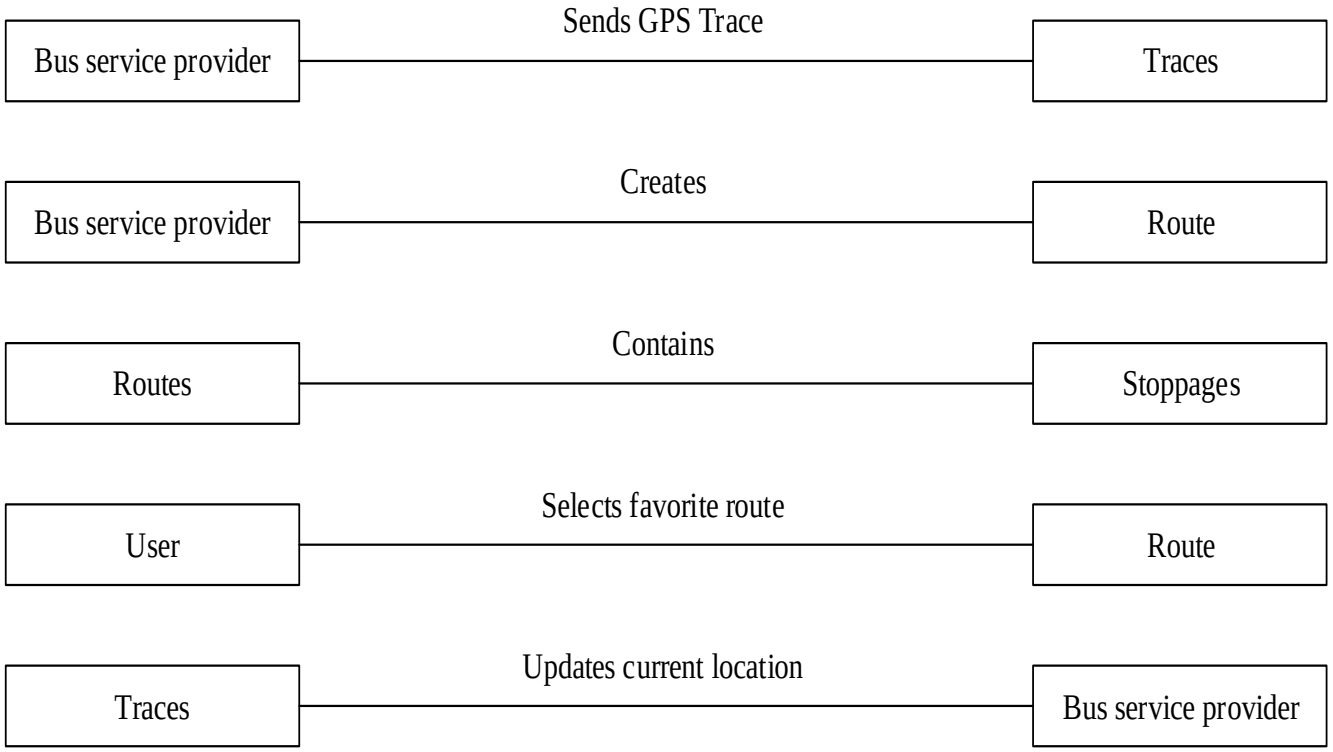

Table 9: Pair to pair relation

Chapter 6
Class based model

This Chapter is to describe class based modeling of Easy Bus Tracker application.

6.1 Class based modeling concept

Class-based modeling represents the objects that the system will manipulate, the operations that will applied to the objects, relationships between the objects and the collaborations that occur between the classes that are defined. The elements of class-based model include classes and objects, attributes, operations, CRC models, collaboration diagram and packages.

6.2 Identifying analysis classes

Step-1: Identifying and categorize all nouns.

General Criteria (GC):

Priority name	ID
External entities	1
Things	2
Occurrences or Events	3
Roles	4
Organizational Units	5
Places	6
Structures	7

Selection Criteria (SC):

Property name	**ID**
Retained information	1
Needed services	2
Multiple attributes	3
Common attributes	4
Common operations	5
Essential requirements	6

NID	**Noun**	**P/S**	**G.C.**	**S.C.**
1*	User	S	4	1, 2, 3, 4, 5, 6
2*	Map	S	7	1, 2, 3, 4, 5, 6
3*	Bus Service Provider	S	4, 5	1, 2, 3, 4, 5, 6
4	Route	S	7	1, 2
5	Position	S	6	1
6	Trip Time	P	-	-
7	Stoppages	S	6	6
8	Arrival Time	S	3	2
9*	Notification	S	3	1, 2
10	Location	S	6	6
11*	Location service provider	S	2	1, 2, 6
12*	Database	S	2, 4, 7	1, 2, 3, 4, 5, 6
13	Authentication	P	-	-

Table 10: Table to identify potential classes

The asterisk (*) marks noun that is a potential class

6.2.1 Accepted Classes:

- User
- Bus Service Provider
- Location service provider
- Notification
- Database

6.2.2 Attribute Identification:

Class Name	Attributes
User	User current location, Favorite Route, Notification Interval
Map	Route ID, Stoppage Name, Stoppage List
Bus service provider	Bus ID, Bus name, Status, Current Position, Route, Stoppages
Location service provider	Route GPS Trace, Stoppage Traces, Bus GPS trace, Arrival Time,
Notification	Selected Route
Database	

Table 11: Attribute Selection Table

6.2.3 Method Identification:

Class Name	Methods
User	ViewMap(), ProvideCurrentLocation(), SelectRoute(), SelectBus(), SelectStoppage(), SetFavouriteRoute(), StartNotification(), SetNotificationInterval()
Map	ShowCurrentPositionOfBus(), ShowRoute(), ShowStoppages(), ShowArrivalTime(), SendSelectedRoute(), ReceiveSelectedRoute()
Bus service provider	StartTracker(), SetBusInformation(), SetStatus(), StoreRoute(), StoreStoppages(), UpdateCurrentPosition(), UpdateMap()
Location service provider	StartGPSTracker(), SendRoute(), SendStopages(), SendCurrentPosistion(), ExtractRoute(), ExtractStoppages(), ProvideStatus(), CollectBusInformation(), CalculateArrivalTime()
Notification	GetUsersCurrentLocation(), getCurrentLocationOfBus(), StoreInterval(), StoreSelectedRoute(), Notify(), StartNotification()
Database	Store(), Delete(), Update(), Select(), Send()

Table 12: Method Selection Table

6.3 CRC modeling

CRC modeling stands for Class Responsibility Collaboration modeling. CRC modeling includes class cards and CRC diagram.

6.3.1 Class cards

Class cards show attributes, methods and collaborating class names along with their responsibility. A responsibility comprises one or more methods together. The class cards of the classes of Easy Bus Tracker application are shown in this section.

6.3.2 CRC diagram

A CRC model is really a collection of standard index cards that represent classes. The cards are divided into three sections. Along the top of the card you write the name of the class. In the body of the card you list the class responsibilities on the left and the collaborators on the right.

The CRC Diagram for Easy Bus Tracker application has been shown in this section. The diagram shows the relationship among the classes as well as the collaborations.

User	
Attributes	**Methods**
User current location, Favorite Route, Notification Interval	+ViewMap() +ProvideCurrentLocation() +SelectRoute() +SelectBus() +SelectStoppage() +SetFavouriteRoute() +StartNotification() +SetNotificationInterval()
Responsibilities	**Collaborators**
1. View the map 2. Provide current location 3. Selects route, bus and stoppages 4. Start and set notification	1. Map 2. Notification, Location service provider 3. Map 4. Notification

Table 13: Class card of User Class

Map	
Attributes	**Methods**
Route ID Stoppage Name Stoppage List	ShowCurrentPositionOfBus() ShowRoute() ShowStoppages() ShowArrivalTime() SendSelectedRoute() ReceiveSelectedRoute()
Responsibilities	**Collaborators**
1. Shows current location of bus, route, stoppages 2. Provides arrival time	1. Bus service provider 2. Location service provider

Table 14: Class card of Map Class

Bus service provider	
Attributes	**Methods**
Bus ID Bus name Status Current Position Route Stoppages	StartTracker() SetBusInformation() SetStatus() StoreRoute() StoreStoppages() UpdateCurrentPosition() UpdateMap()
Responsibilities	**Collaborators**
1. Uses the GPS tracker 2. Provide current location of the bus 3. Provide GPS traces to calculate arrival time	1. Location Service provider 2. Map 3. Location Service provider

Table 15: Class card of Bus service provider Class

Notification	
Attributes	**Methods**
Selected Route	GetUsersCurrentLocation() getCurrentLocationOfBus() StoreInterval() StoreSelectedRoute() Notify () StartNotification()
Responsibilities	**Collaborators**
1. Notify user	1. User

Table 16: Class card of Notification Class

Location service provider	
Attributes	**Methods**
Route GPS Trace Stoppage Traces Bus GPS trace Arrival Time	StartGPSTracker() SendRoute() SendStopages() SendCurrentPosistion() ExtractRoute() ExtractStoppages() ProvideStatus() CollectBusInformation() CalculateArrivalTime()
Responsibilities	**Collaborators**
1. Extract route, stoppages 2. Update current location of the bus 3. Update the current status of the bus	1. Bus service provider 2. Map 3. Bus service provider

Table 17: Class card of Location Service Provider Class

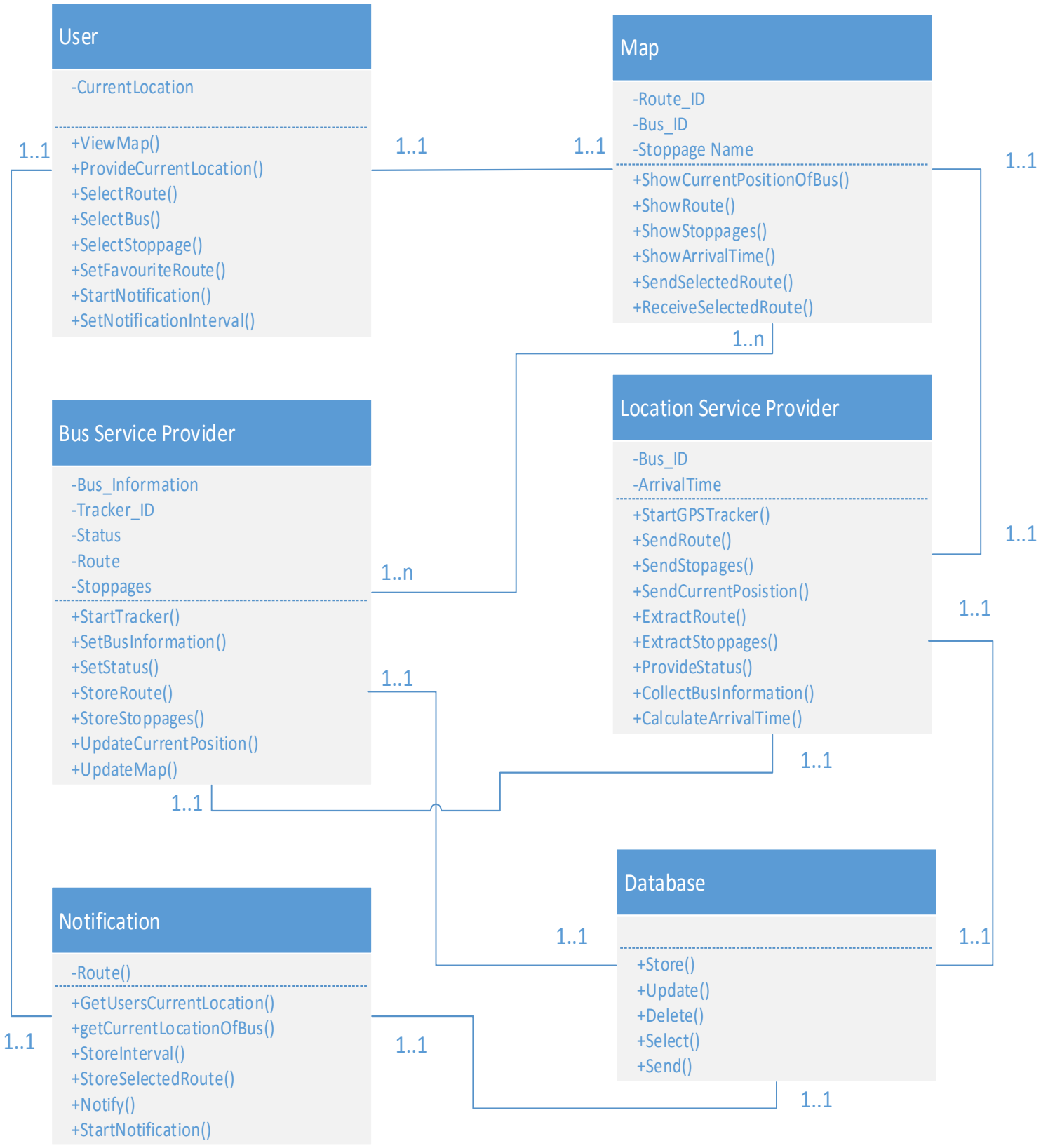

Figure 18: Class Responsibility Collaboration

Chapter 7
Flow Oriented Modeling

7.1 Introduction

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 notation in use today.

7.2 Data flow diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as preliminary step create an overview of the system. The DFD takes an input-process-output view of a system. Data object flow into the software, are transformed by processing elements and resultant data objects flow out of the software. Data object are represented by labeled arrows and transformations are represented by circles. A DFD shows what kind of information will be input and output from the system, where the data will come from and go to, and where the data will be store.

7.2.1 Grammatical parsing

Nouns: User, Map, Bus service provider, Route, Stoppages, Position, Location, Arrival Time, Notification, Location service provider, registration

Verbs: View, Show, Provide, Calculate, Set, Track, Store, Update, Serve

7.2.2 DFD of Level 0 Use Case: Easy Bus Tracker system

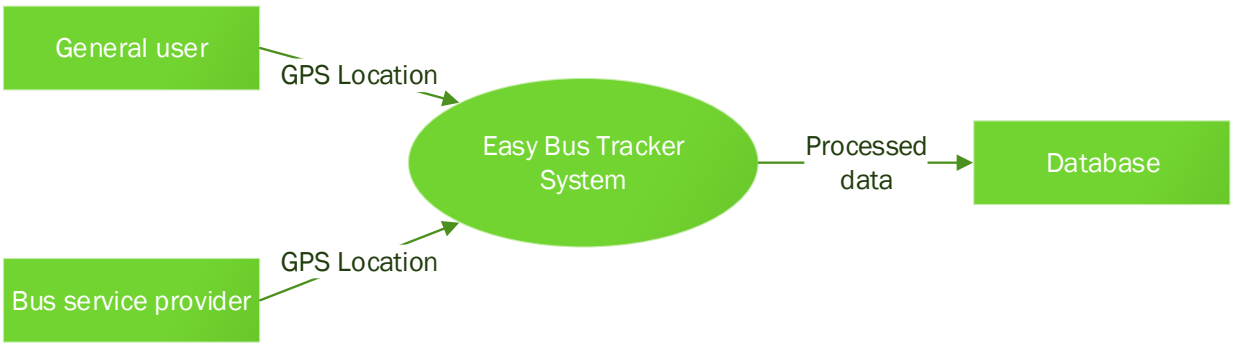

Figure 19: DFD Level 0 of Easy Bus Tracker System

7.2.3 DFD of Level 1 Use Case: Easy Bus Tracker system

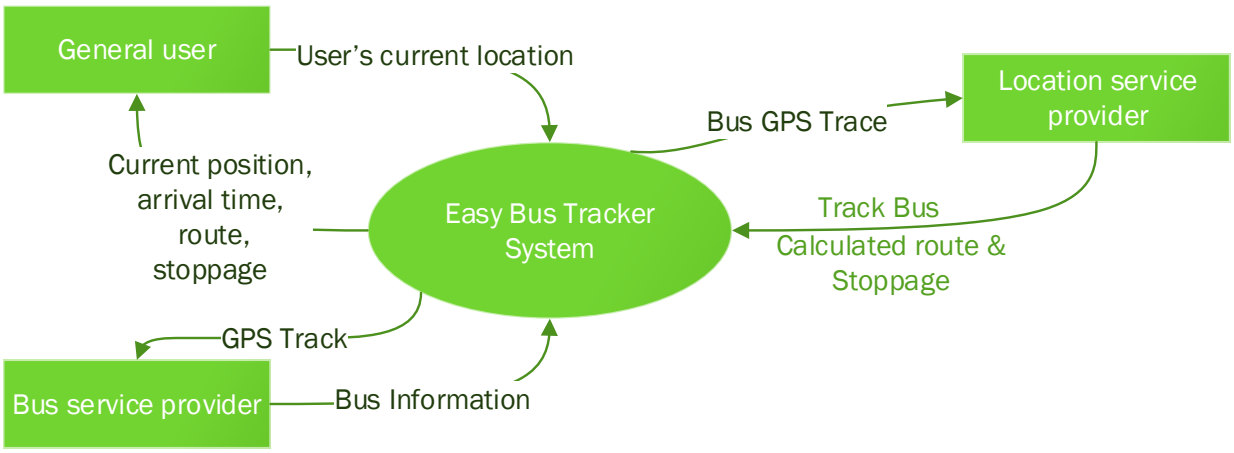

Figure 20: DFD Level 1 of Easy Bus Tracker System

7.2.4 DFD of Level 1.1 Use Case: User

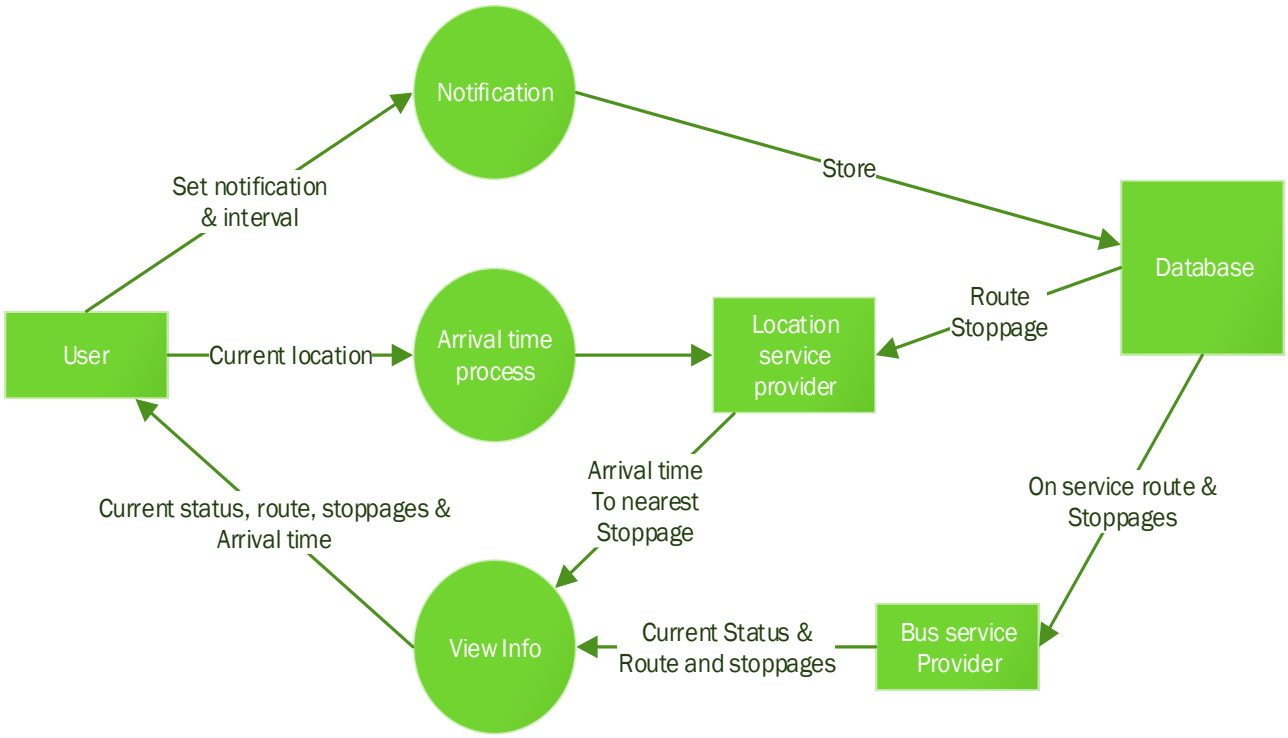

Figure 21: DFD Level 1 of User

7.2.5 DFD of Level 1.2 Use Case: Bus service provider

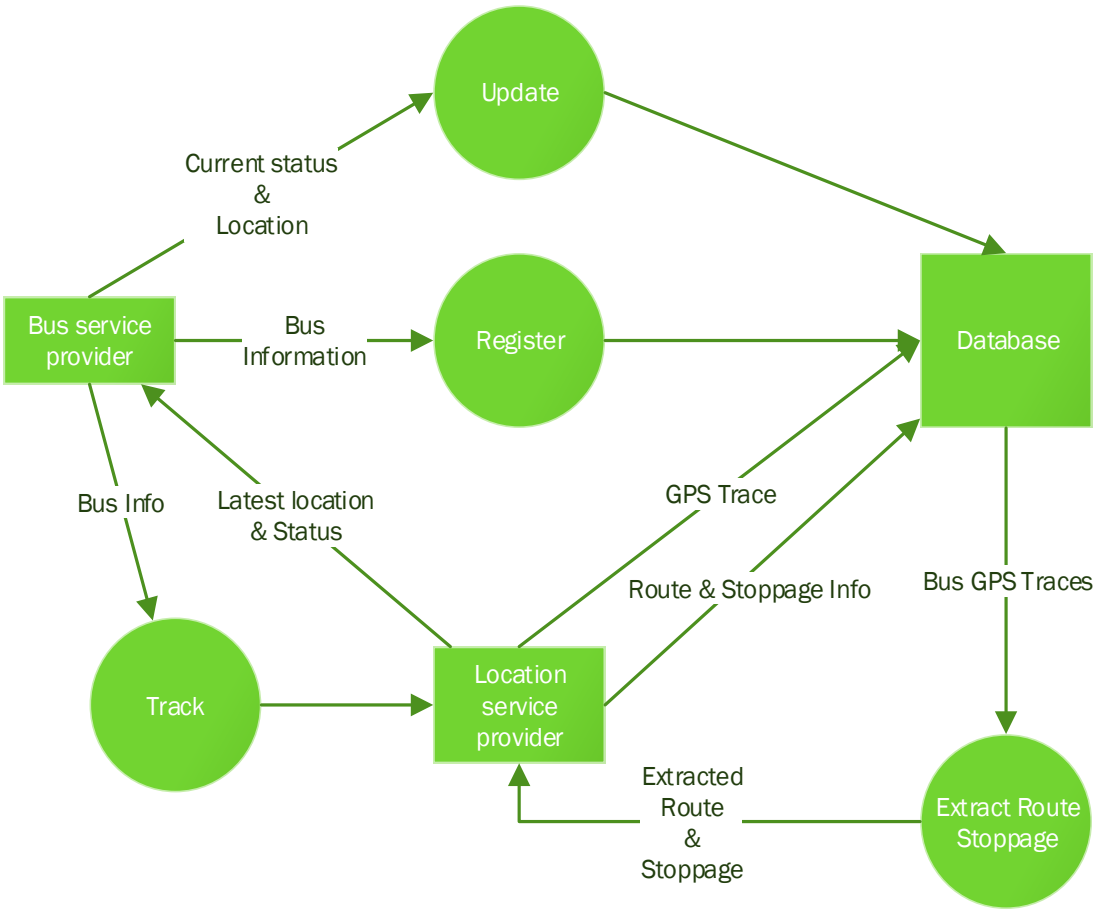

Figure 22: DFD Level 1 of Bus service provider

Chapter 8
Behavioral Model

8.1 Introduction

Behavior modeling is also referred to as State modeling, State machines and State transition matrix. Behavior modeling is when one thinks of his ideas in terms of states and transitions. This requires both identifying all of the interesting states of being that software or its components are likely to be in. And also, at a high level, abstracting what events are likely to cause software or its components to change between states of being.

8.2 Identifying events

Here we have identified events from the **usage scenario** and listed their corresponding initiators and collaborators.

Events	Initiator	Collaborative class
Collects the selected route	Map	User
Sends the route for route map and stoppages	Map	Bus service provider
Show the route and stoppages, current location of the bus	Map	Bus service provider, Database
Show arrival time	Map	Location service provider
Start tracker for bus	Bus service provider	Location service provider

Update current position and status of the bus	Location service provider	Bus service provider
Store extracted route and stoppage	Bus service provider	Location service provider, Database
Events	**Initiator**	**Collaborative class**
Provide bus information for registering	Bus service provider	Database
Select favorite route	User	Database
Set notification and time interval	User	Notification
Provide the current location of the user	User	Location service provider
Viewing the map	User	Map
Calculate arrival time	Location service provider	Map, Bus service provider, Database
Notify user	Notification	

Table 18: Table of Event identification

8.3 State diagram

State diagram represents active states for each class the events (triggers). The following section will show state diagrams for each class in the Easy Bus Tracker app

8.3.1 State Transition Diagram of User

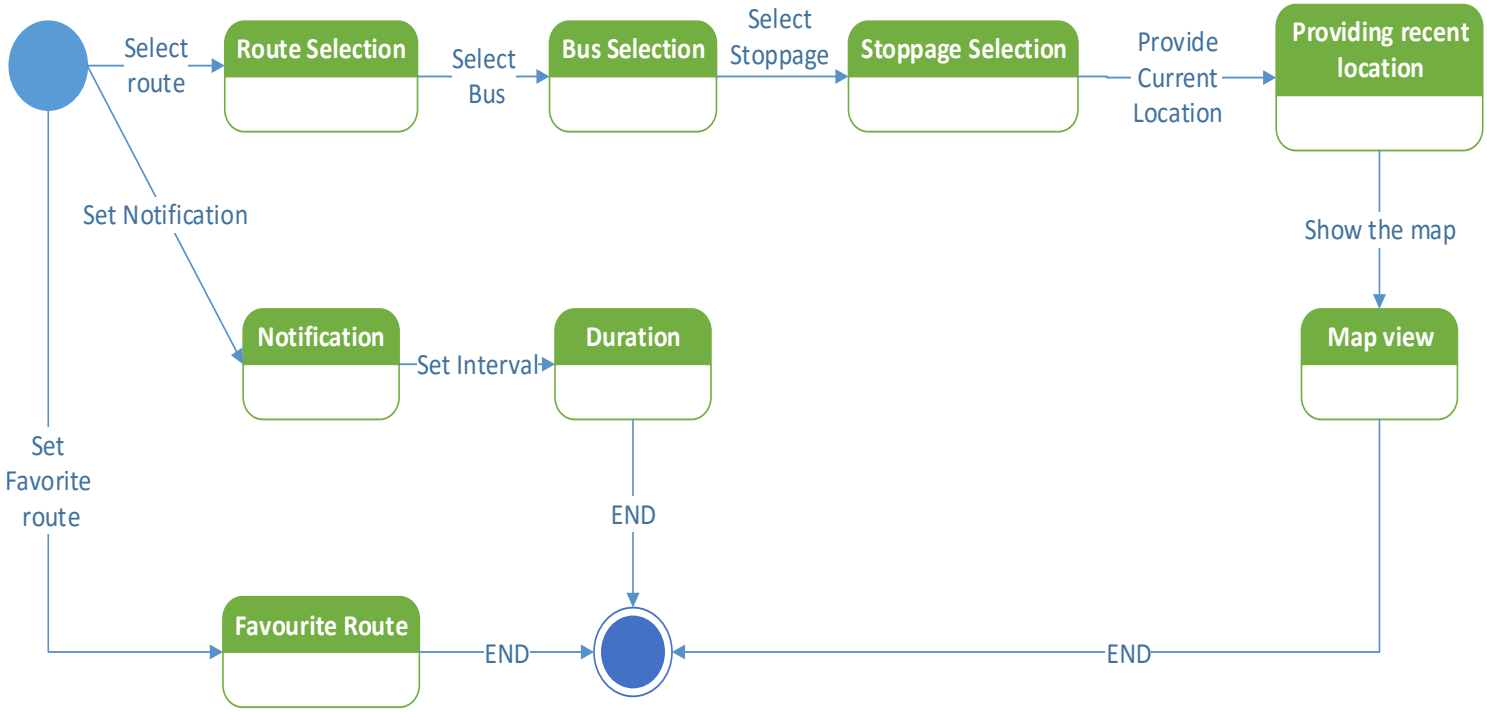

Figure 23: State transition diagram of User

8.3.2 State Transition Diagram of Map

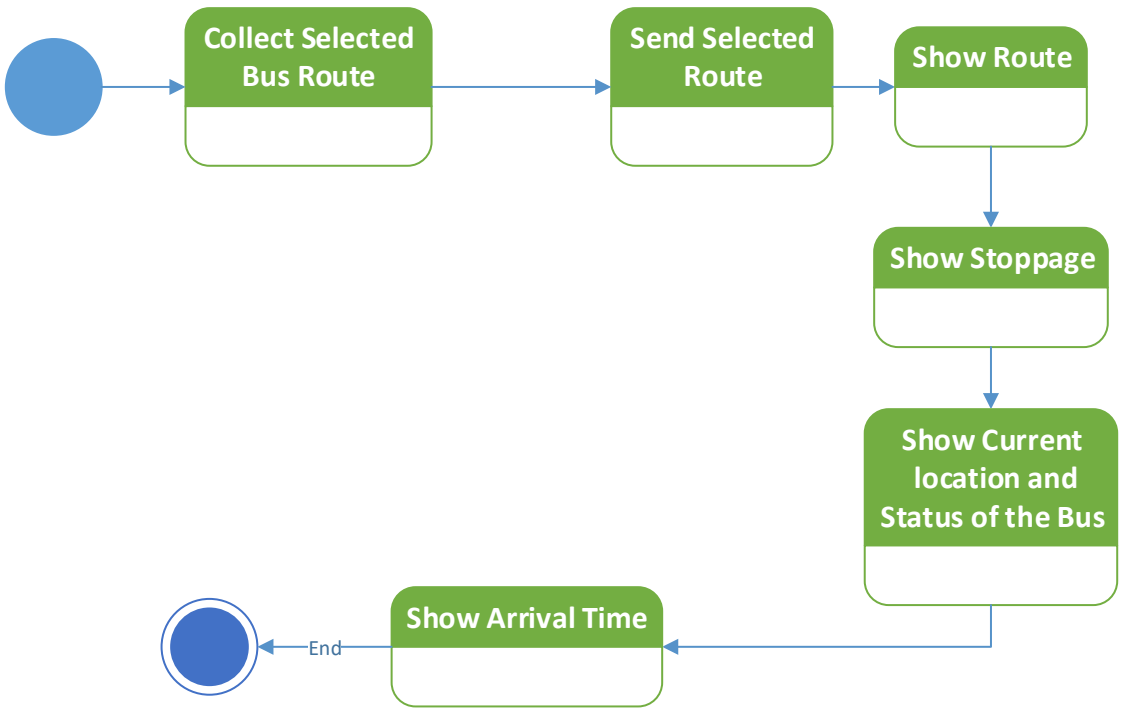

Figure 24: State Transition Diagram of Map

8.3.3 State Transition Diagram of Bus service provider

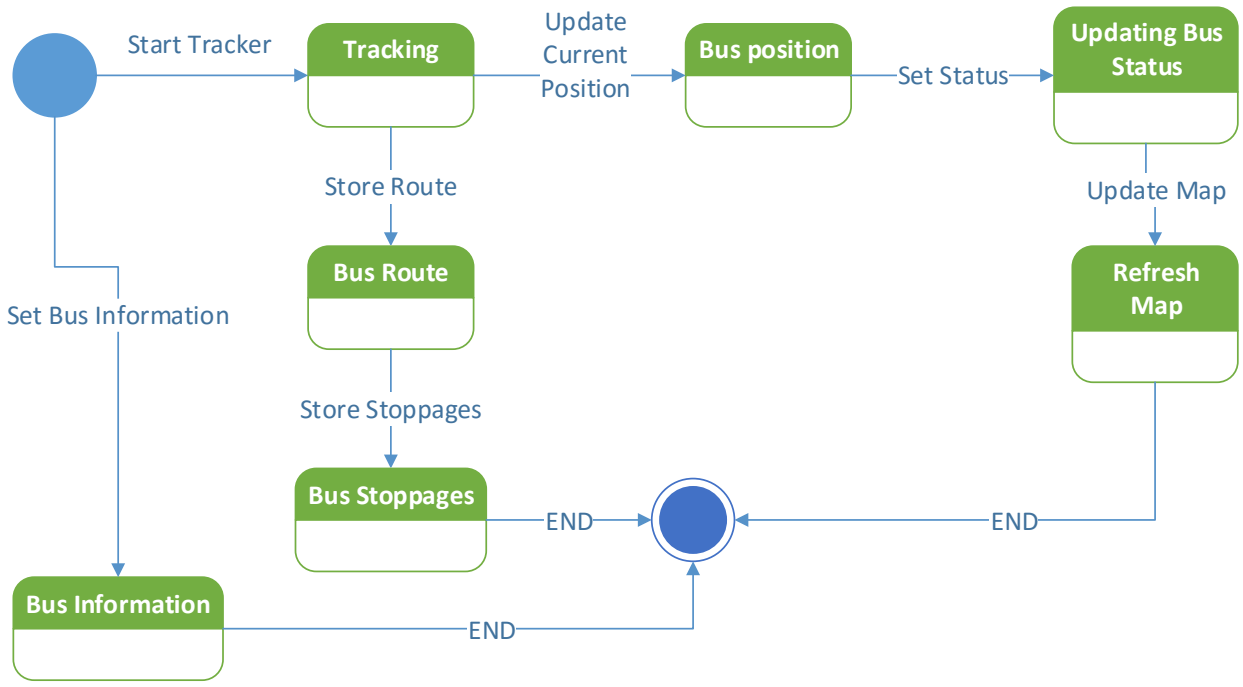

Figure 25: State transition diagram of Bus service provider

8.3.4 State Transition Diagram of Location service provider

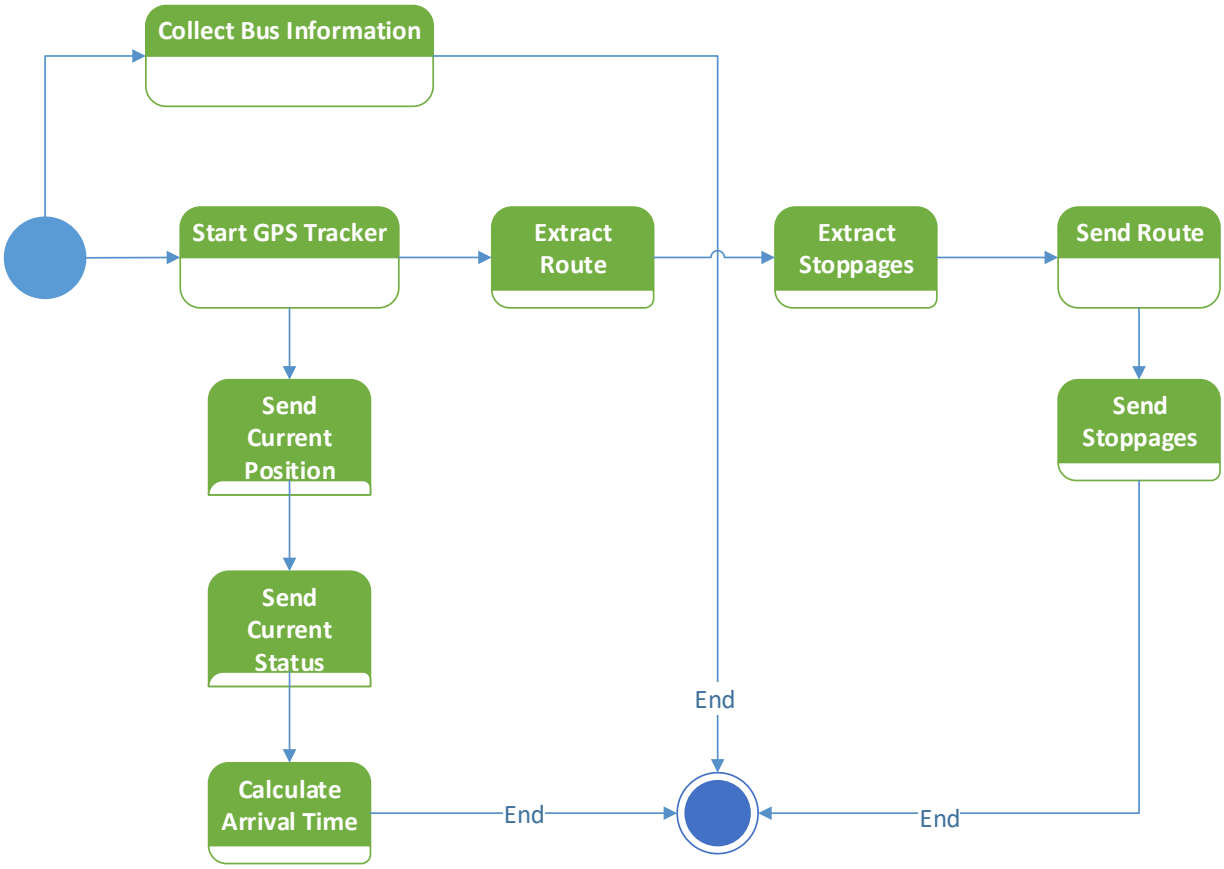

Figure 26: State transition diagram of Location service provider

8.3.5 State Transition Diagram of Notification

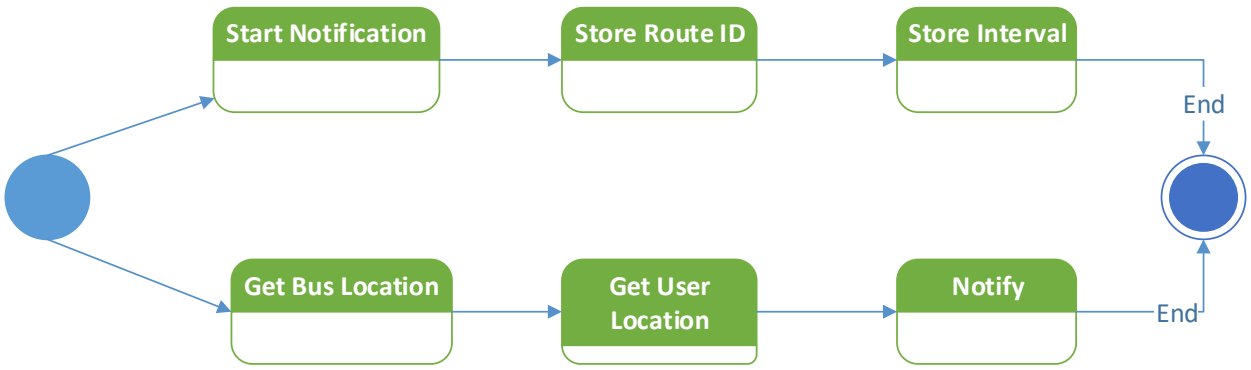

Figure 27: State transition diagram of Notification

8.3.6 State Transition Diagram of Database

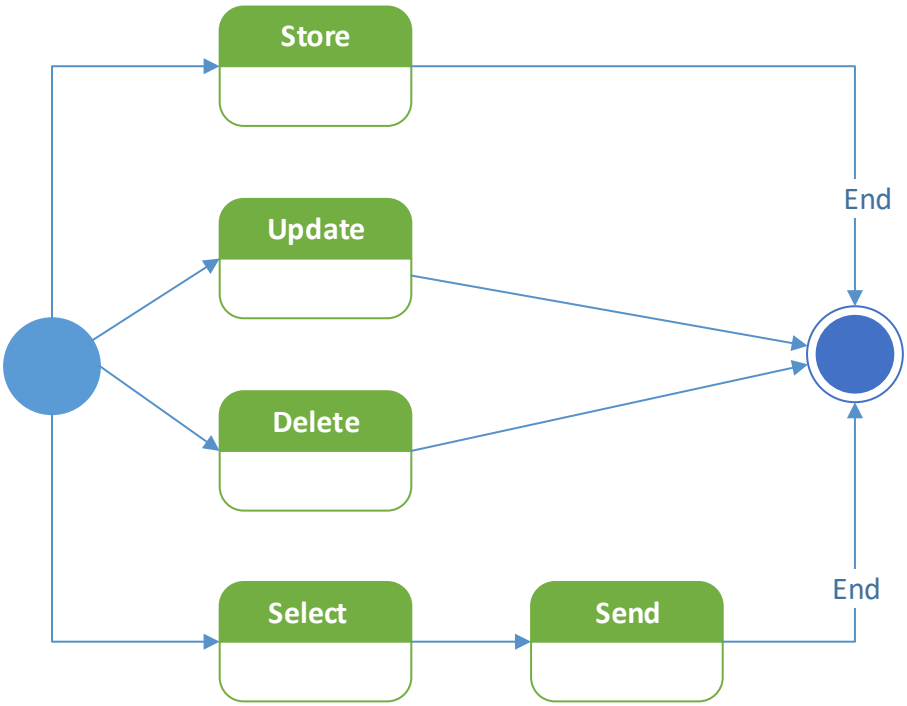

Figure 28: State transition diagram of Database

8.4 Sequence diagram

The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them. However, an organization's business staff can find sequence diagrams useful to communicate how the business currently works by showing how various business objects interact. Besides documenting an organization's current affairs, a business-level sequence diagram can be used as a requirements document to communicate requirements for a future system implementation. During the requirements phase of a project, analysts can take use cases to the next level by providing a more formal level of refinement. When that occurs, use cases are often refined into one or more sequence diagrams.

An organization's technical staff can find sequence diagrams useful in documenting how a future system should behave. During the design phase, architects and developers can use the diagram to force out the system's object interactions, thus fleshing out overall system design.

One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams. In addition to their use in designing new systems, sequence diagrams can be used to document how objects in an existing (call it "legacy") system currently interact. This documentation is very useful when transitioning a system to another person or organization.

The Sequence Diagram of Easy Bus tracker application has been shown here:

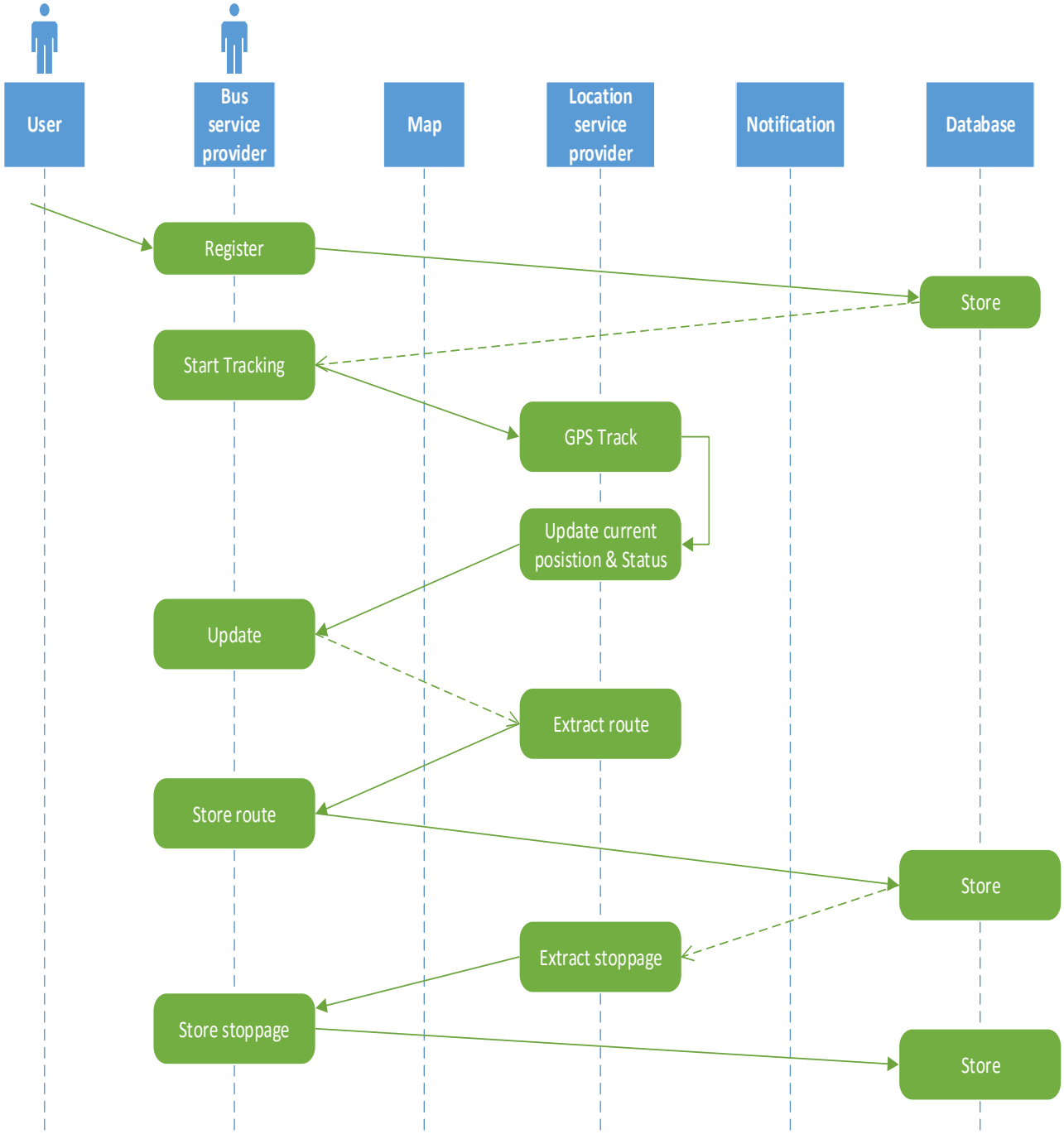

Figure 29: Sequence Diagram of Easy Bus Tracker system: Part 1

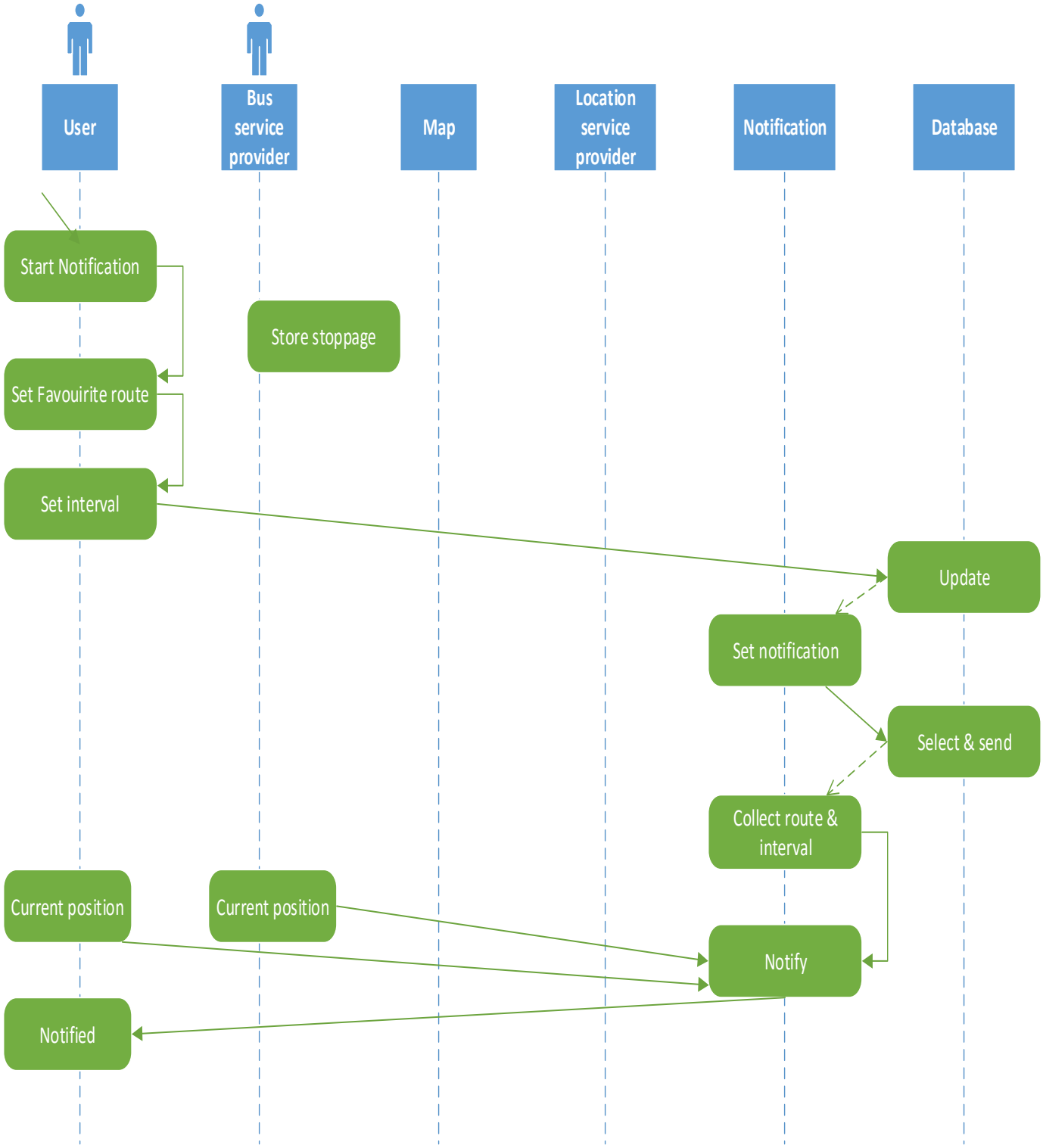

Figure 30: Sequence Diagram of Easy Bus Tracker system: Part 2

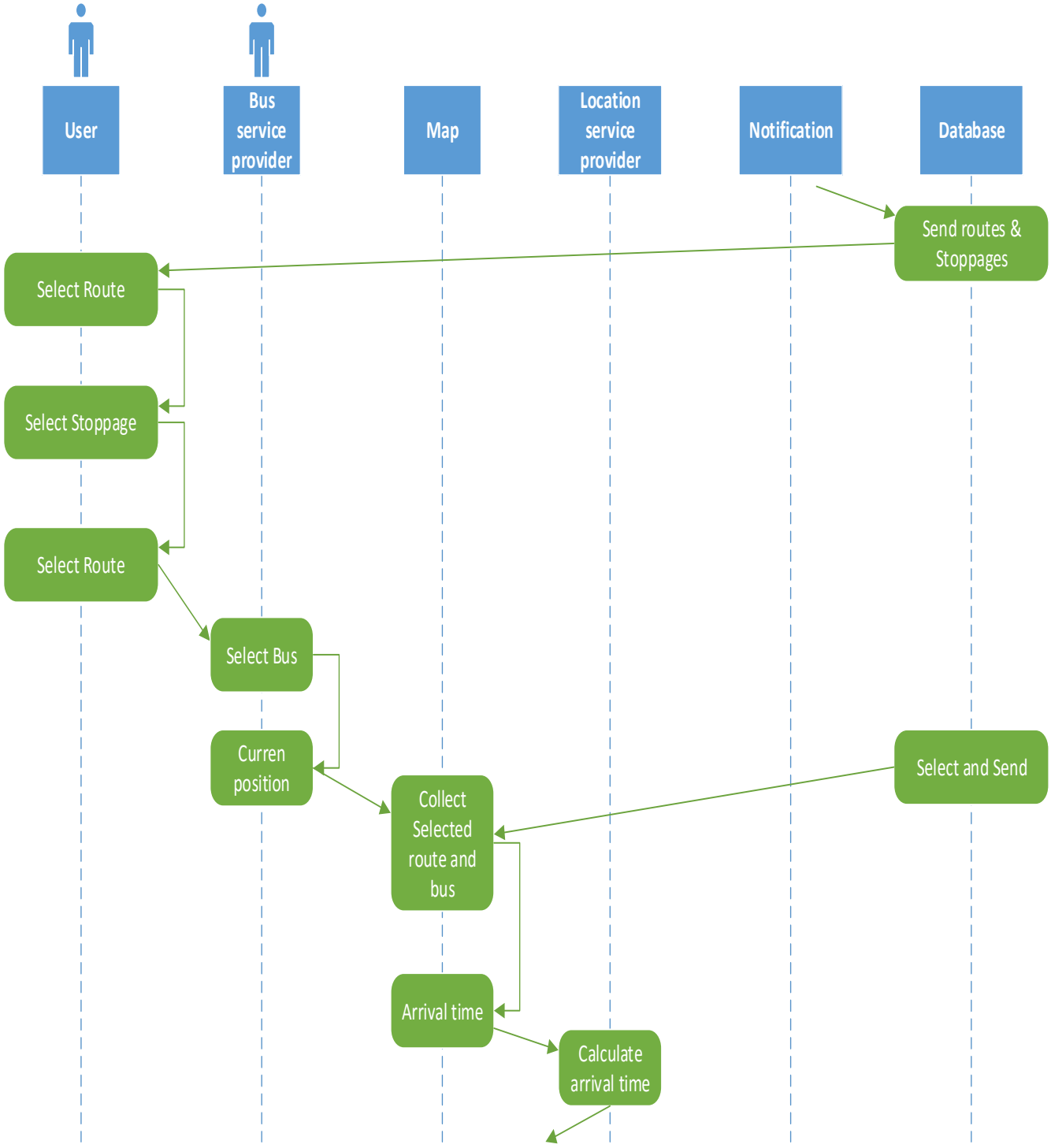

Figure 31: Sequence Diagram of Easy Bus Tracker system: Part 3

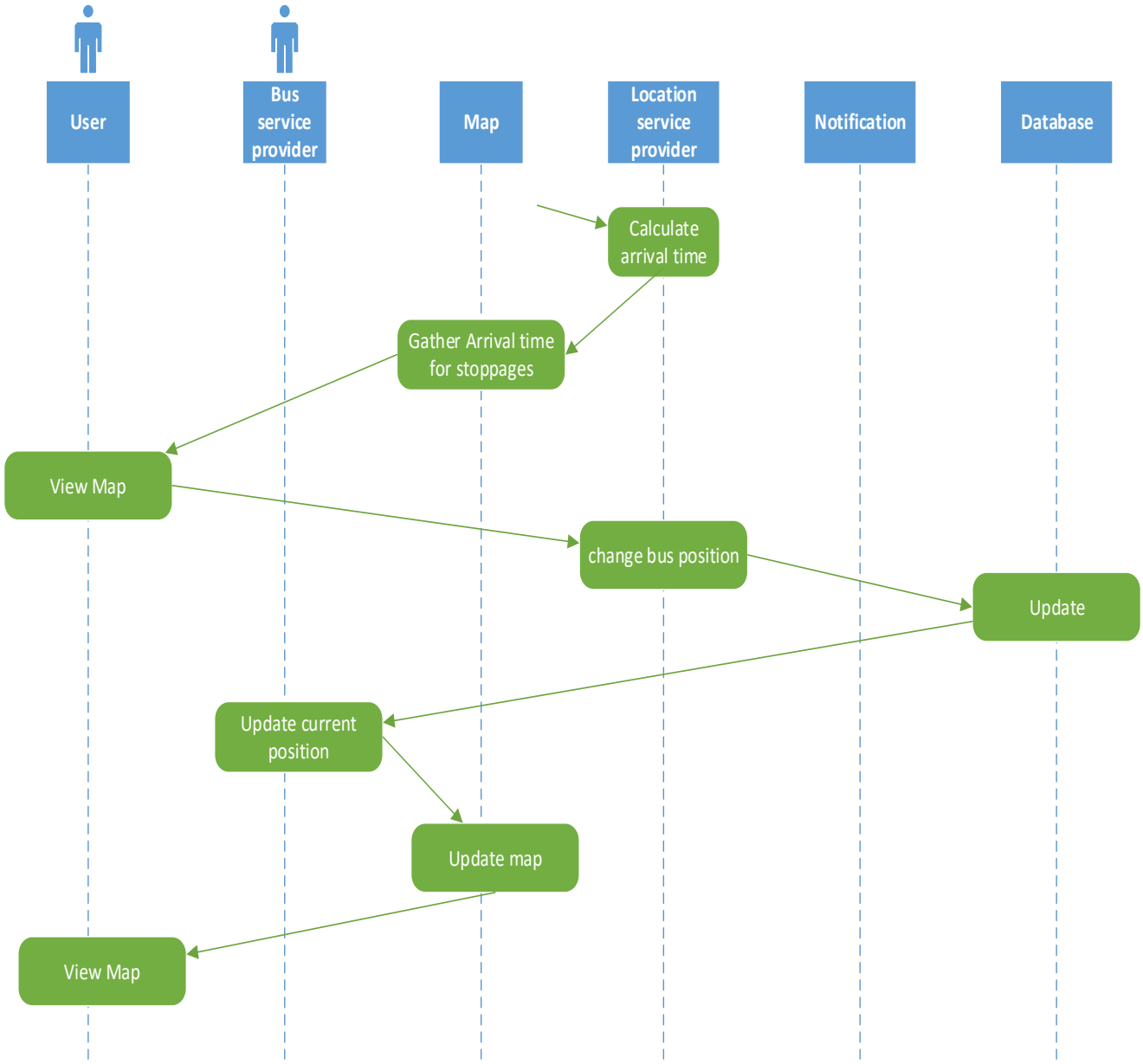

Figure 32: Sequence Diagram of Easy Bus Tracker system: Part 4

Chapter 9
Conclusion

It’s really difficult to represent something real with pen and paper. I am pleased that I have been able to complete the requirements analysis of Easy Bus Tracker app and publish the SRS of the project. I think that this report has been written in an easy-to-read way as well as with full information required to have e good concept over the idea. I hope that any reader going through this document can easily understand the whole idea behind the Easy Bus Tracker app. I also hope that it will be an easy path-showing document for the implementation of the app!

Appendix

References

1. Pressman, Roger S. Software Engineering: A Practitioner's Approach (7th ed.). Boston, Mass: McGraw-Hill. ISBN 0-07-285318-2
2. Ralph, Paul (2012). "The Illusion of Requirements in Software Development". Requirements Engineering
3. Sommerville, I. Software Engineering, 7th ed. Harlow, UK: Addison Wesley, 2006
4. http://www.mnhe.com/pressman, accessed on 3rd April, 2016
5. http://www.mks.com/solutions/discipline/rm/requirements-engineering, accessed on 6th, April, 2016