LOCO ALARM

CSE470 FINAL REPORT

Team Members

Syed Zafrul Hassan - 15301064
Abu Wakkas Bastob - 15301071
M. Rafiul Alam - 15101130
Nusayer Masud Siddique - 16301102

Submitted to:

MONZUR MUHAMMAD

Lecturer
BRAC UNIVERSITY

Table of Contents

1. INTRODUCTION

Our project is on Location Based Alert System(Geofencing).

Geofencing is a location-based service in which an app or other software uses GPS, RFID, Wi-Fi or cellular data to trigger a pre-programmed action when a mobile device or RFID tag enters or exits a virtual boundary set up around a geographical location, known as a geofence.

1.1. PURPOSE OF THE SYSTEM

Geofencing is the creation of virtual fences around physical locations – stores, entertainment venues, and parks literally anywhere. Turning any phone into virtual marketing real estate for a brand with the added benefit of being where and when the consumer wants it.When one of the app users enters or exits a **geofenced** area, a targeted push notification is sent directly to their device.

Suppose we define a geofence . Once we do that we can use '**enter', 'exit'** or '**dwell'** condition on this geofence as the trigger for any action.

Let's consider a simple example. Suppose we create a geofence around our retail store. Now there could be three simple use-cases:

- Enter- Trigger a notification if user **enters** the geo-fence
- Exit- A notification if he **exits** the geofence
- Dwell- if he **stays** in the geofence for certain amount of time. This has an important benefit. Consider this- suppose the radius of your geofence is in few meters. Now, it might happen that a certain user just quickly drives past it. He stays in the geofence for a very brief period so sending them a notification is equivalent to spam. By putting a dwell condition you target only those users who stay for only certain amount of time thereby segmenting your audience further for good and improved targeting.

Combining the above three with time-stamp and behavioral data can make us think of myriad use-cases depending on the business.

For instance, if we have data of the users who have **visited Alaska** more than **thrice** in **last** year then we would have some certainty that there are potential customers for Salmon fishing kit.

1.2. SCOPE OF THE SYSTEM

Use-cases:

- **Deliveries:** customers waiting for a package could be notified via text message when the delivery vehicle enters a certain radius of their home.
- **Retail:** consumers who walk within a certain distance of a shop can be notified with special offers and incentives to swing by and visit.
- **Security:** mobile tablets that belong to an organisation that deals with sensitive information could be programmed to become disabled when the leave the location.
- **Restaurants:** information on daily specials can be delivered to a customer who is in range of your restaurant.
- **Cinemas:** film showing times could be delivered direct to nearby regular customers.

1.3. OBJECTIVES AND SUCCESS CRITERIA OF THE PROJECT

The usage of our system involves a location-aware device of a location based service(LBS) user entering or exiting a geo-fence. This activity could trigger an alert to the device's user as well as messaging to the geo-fence operator. This info, which could contain the location of the device, could be sent to a mobile telephone or an email account.

Out of the several benefits, it facilitates what is known as **Proximity marketing.**

Proximity marketing is a type of location-based marketing in which you send personalized content to the user depending on how close he is to a particular location. This particularly helped offline retailers to have the same capability as their online counterparts in terms of providing contextual recommendations to users.

Although our system is usually used in marketing, but it’s important to note that this technology can also be used by employees to track the location of their employees.

Another one of the advantageous uses of Geo-Fencing is to keep track of employees when they are out in the field.

Other benefits of using our system include:

- The ability to set up an alert for any unauthorized entry to the site, which is very important if valuable tools and equipment are kept there.
- Avoiding time theft, as employees are not able to spend time in other locations not working while they are pretending to be on the job site.
- Ensuring employees are on time, as they won’t be able to log in and start their shift while they are still on the drive to work.
- Allowing the employer to collect and view data on where their employees are and how they are moving around the job site, which helps to understand how the work is being done.

1.4. OVERVIEW

Our system is a method of defining a virtual barrier on a real geographical location.

It can prompt mobile push notifications, trigger text messages or alerts, send targeted advertisements on social media, allow tracking on vehicle fleets, disable certain technology or deliver location-based marketing data.

It is a superb technology with revolutionary potential to transform so called insignificant local data concerning a person's movement into great business intelligence, security intelligence, workforce monitoring and management, remote surveillance of transport and people, etc.

2. PROPOSED SYSTEM

2.1. OVERVIEW

2.1.1 Programming Languages

For the front-end of our application we have used React Native which is a Javascript framework used to develop native mobile applications. Along with that, it can deliver UI for both Android and iOS devices. The backend REST API services of our application is written in Golang from scratch. It offers great performance and its single binary file compilation makes it easier to deploy Golang applications to any server.

2.1.2 Database

PostgreSQL is used for our project. We have taken the advantages of some advanced features of PostgreSQL ie. Transactions, inheritance etc.

2.2.3 Other Technologies

Our application uses Google Map API to gather location information and show geofenced maps set by the users. Moreover, for tracking bugs Sentry is implemented both in the frontend and backend part of our application.

2.2. FUNCTIONAL REQUIREMENTS

- User login and registration using OTP (One Time Password)
- User can add and manage his tasks or reminder notes
- Categories can be attached to each task
- Locations can be attached to tasks

- Locations set to tasks are geofenced and alert sounds are played whenever user's location is inside the geofence
- By default, alerts are not played when the user is inside a geofenced location where silence is preferred to be maintained
- All the tasks added by the user can be viewed in a calendar
- The calendar can be shown in 4 modes – Day, Week, Month and Year
- Reminders can be set to tasks can notify the users about it before the scheduled time
- Users can set the alert sound that is to be played
- User can check the list of tasks those have been mark as completed or has passed the mark of the current date in task history
- User can remove tasks from task history to remove them permanentl

2.3. NON-FUNCTIONAL REQUIREMENTS

- Facebook Account Kit is used to send OTP to users which is very reliable and secure
- 99% server uptime
- No premade frameworks are used in the backend for faster performance
- User must select a category for each task before adding it
- React Native is used to develop the application for its cross-platform compatibility
- The application is responsive to most of the modern devices
- At least 512MB RAM and 50MB storage space is required to install and run the application
- Google Map API requests are limited to 100,000 requests/day
- Sentry is used to automatically track errors in the production version of the application
- Location permission is required for geofencing and accurate alerts

2.4. SYSTEM MODELS

2.4.1 Scenario

Model	View	Controller
PostgreSQL Tables GORM (Go Postgres ORM) Models React Redux States	React Native UIs	Golang REST API

2.4.2 Model

A PostgreSQL database table is created for each type of data we have used in our application. The schema of this database is also modelled in Golang using an ORM library called GORM. Both the data tables and the ORM Schemas built using GORM are the models of our application. Eg. Users, Tasks, Categories, Locations etc. There is another type of model that is used in the frontend part of our application. React Native uses models to control how data should be displayed in our application. Redux library is responsible for these models. These can be categorized into states and mutations. So, these states and mutations are the models of our React Native application. Thus, redux is also the model of our application. Eg. Fetched User Info, List of Tasks etc.

2.4.3 View

React Native is used to create all the interfaces of our application. All the components for UIs are the View part of our application. Eg. Registration Page, Verify OTP Page, Add Location to Tasks Page etc. Thus, all the components we have written for the UI in React Native are part of the view model of our application.

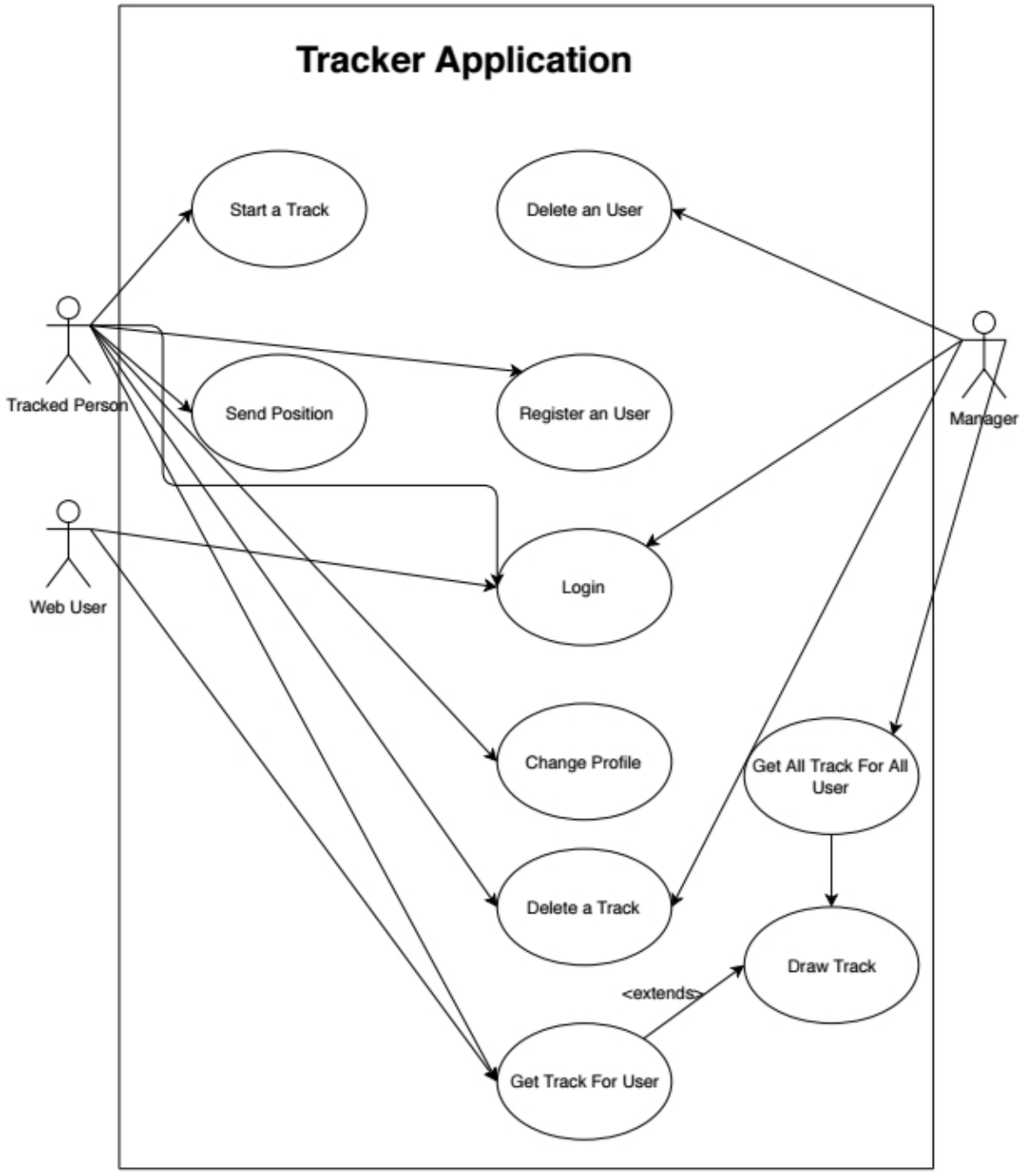
2.4.4 Controller

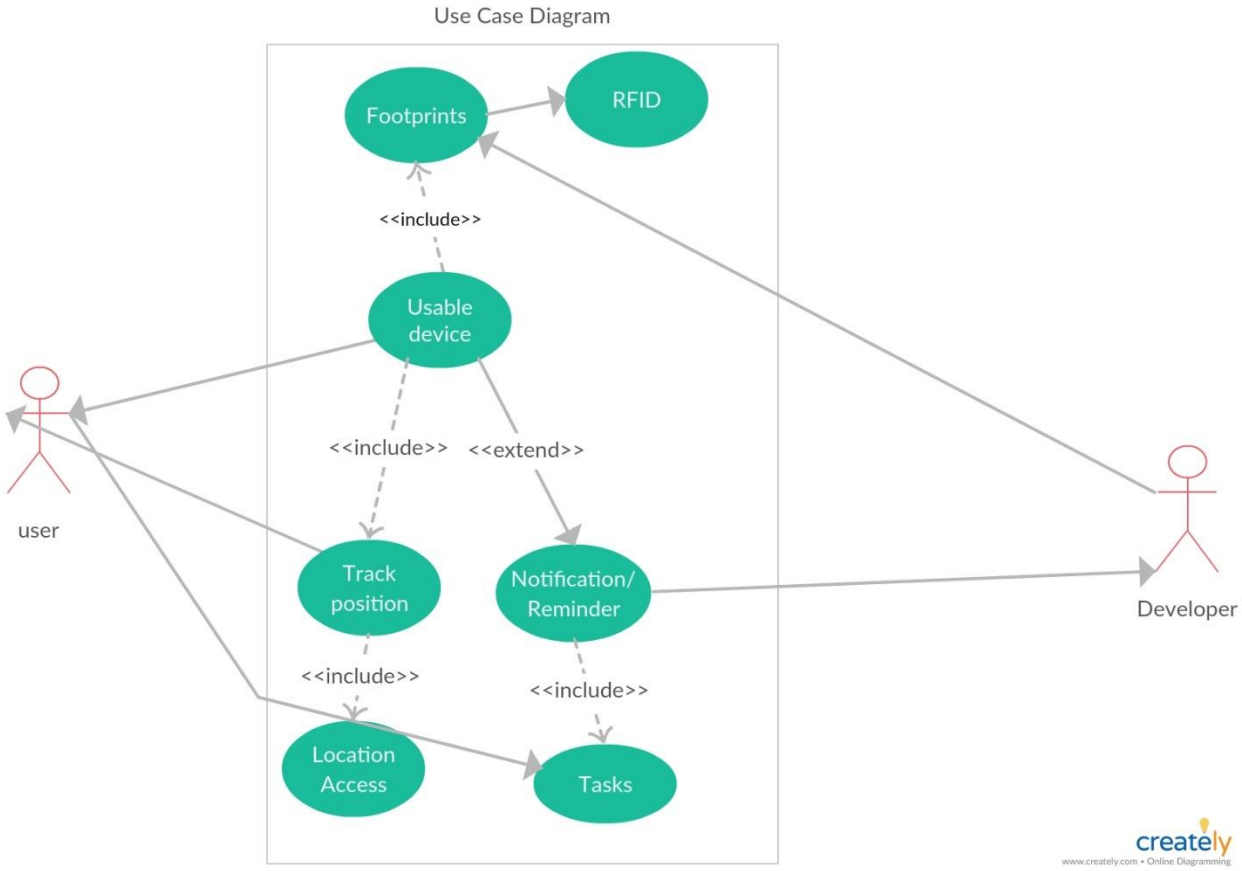
For the view part to show data we require a controller to fetch data from model and represent it to views accordingly. Since, data communication in our application is based on REST API, all of the data are received in the backend by using GET requests. POST/UPDATE/DELETE requests are received by the backend to perform their respective task. The nature of these requests is controlled by the backend API service written in Golang. These services contain multiple controllers for each type of requests. These are the controllers of our application. Eg. AuthController, TaskController, GeofencingController etc.

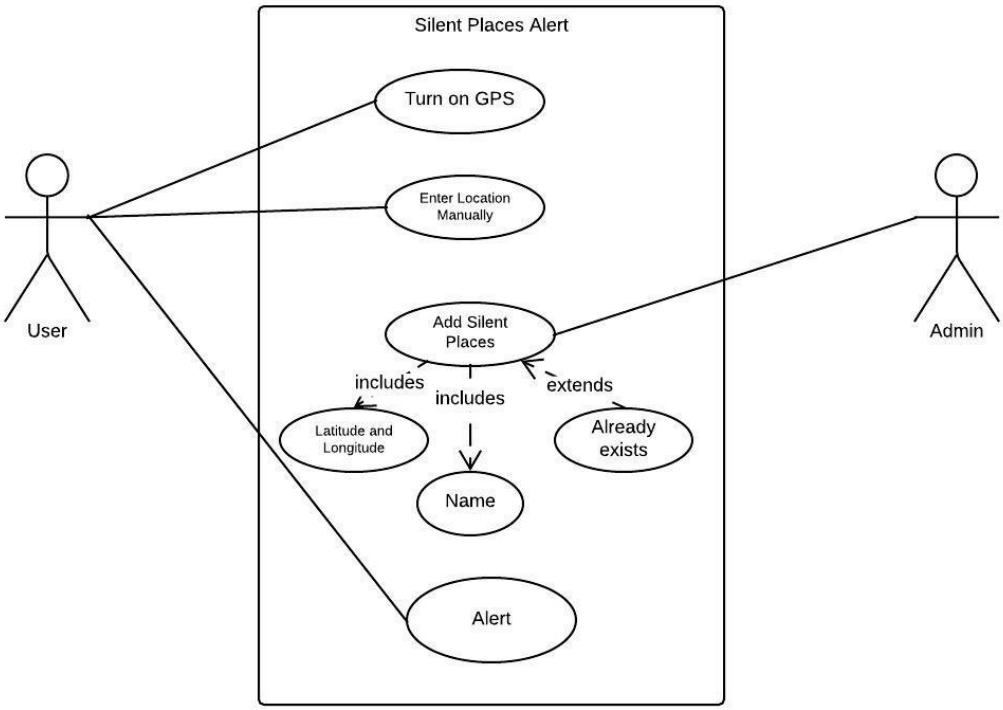
2.4.5 UML Models

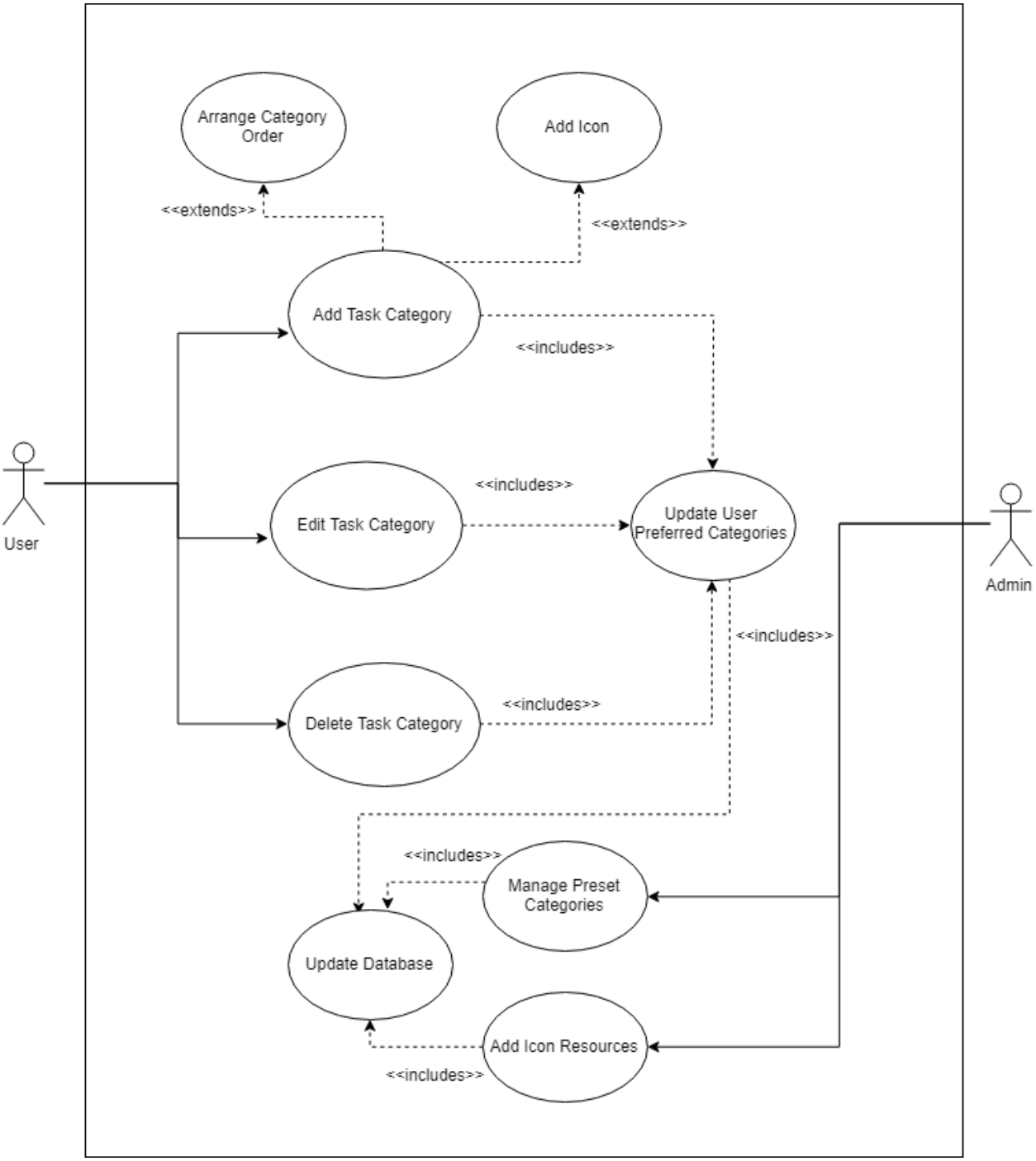
The following pages contains User Case, Sequence and Class diagrams for this application.

2.4.5.1 User Cases

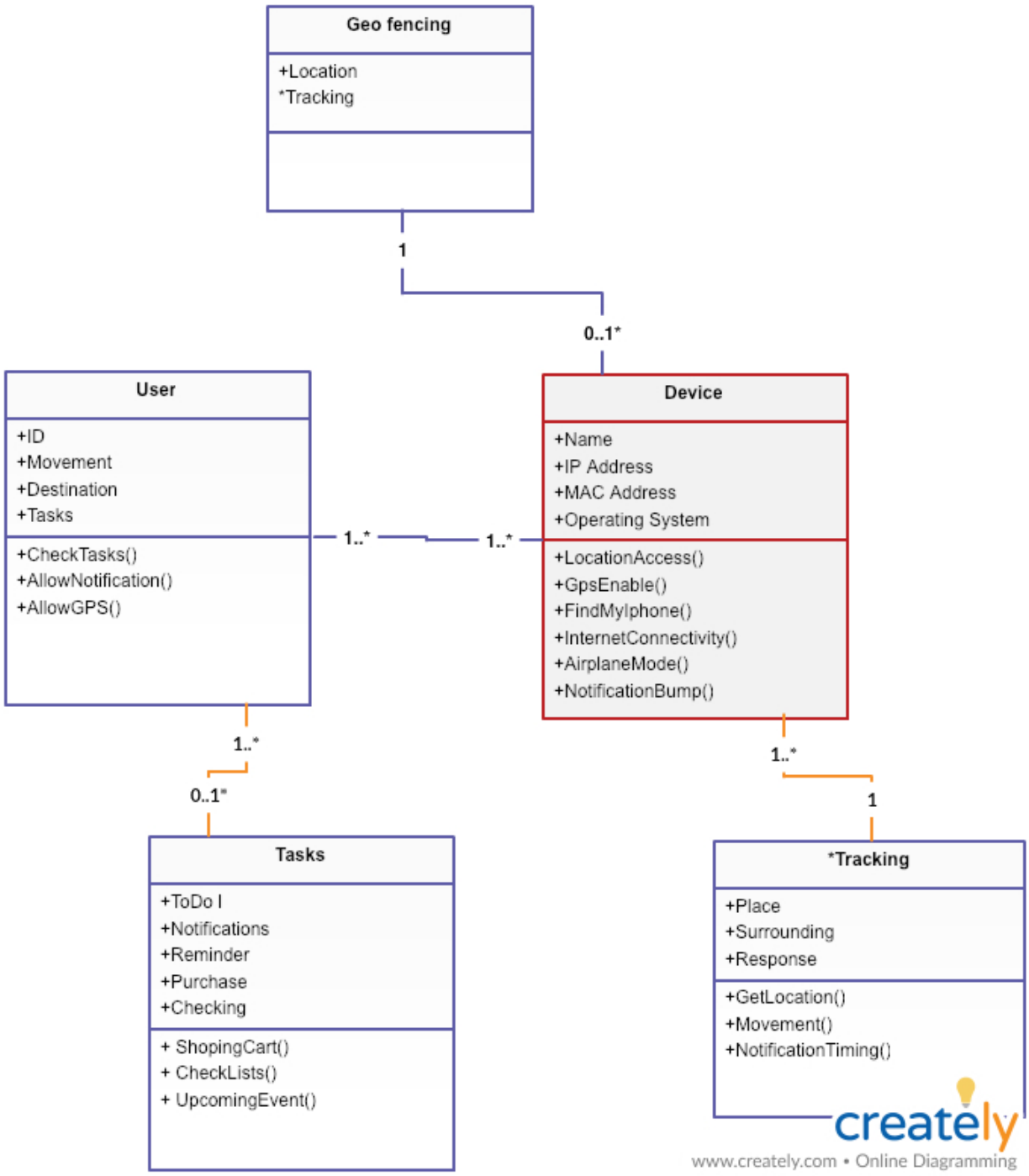


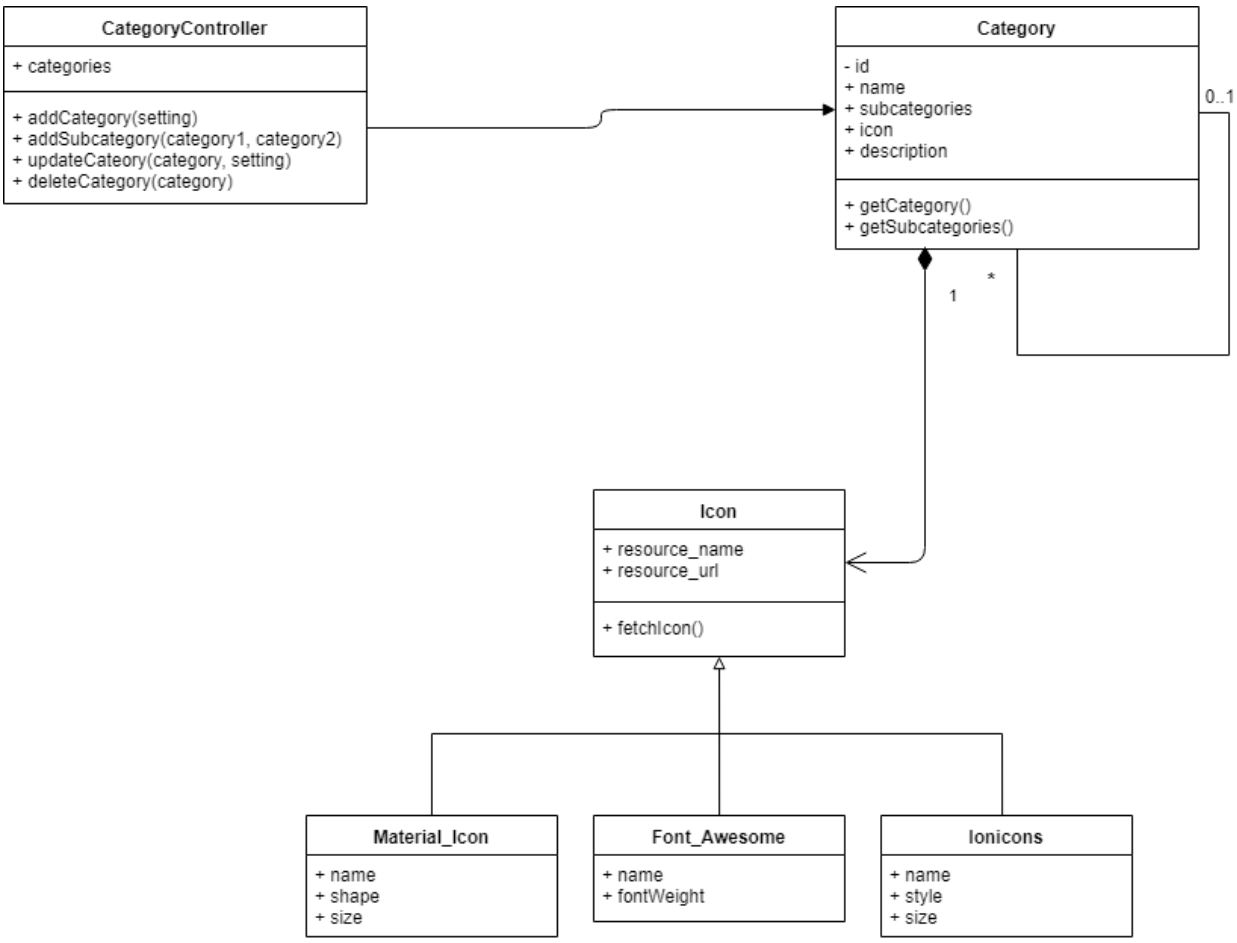

Use Case Diagram
Footprints
RFID
<<include>>
Usable device
<<include>>
<<extend>>
user
Track position
Notification/ Reminder
Developer
<<include>>
<<include>>
Location Access
Tasks
creately
www.creately.com • Online Diagramming

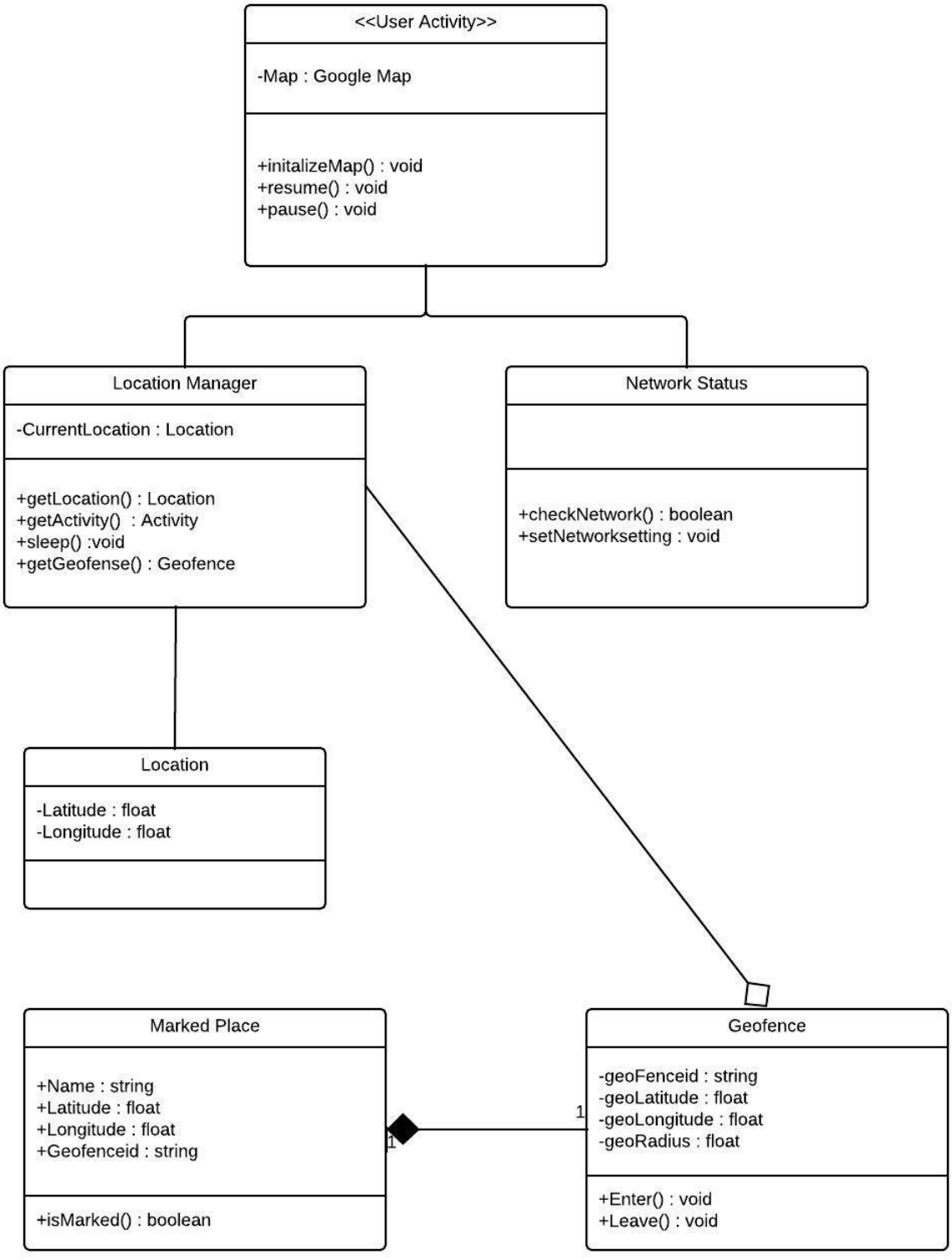

Silent Places Alert
Turn on GPS
Enter Location Manually
User
Admin
Add Silent Places
includes
includes
extends
Latitude and Longitude
Already exists
Name
Alert

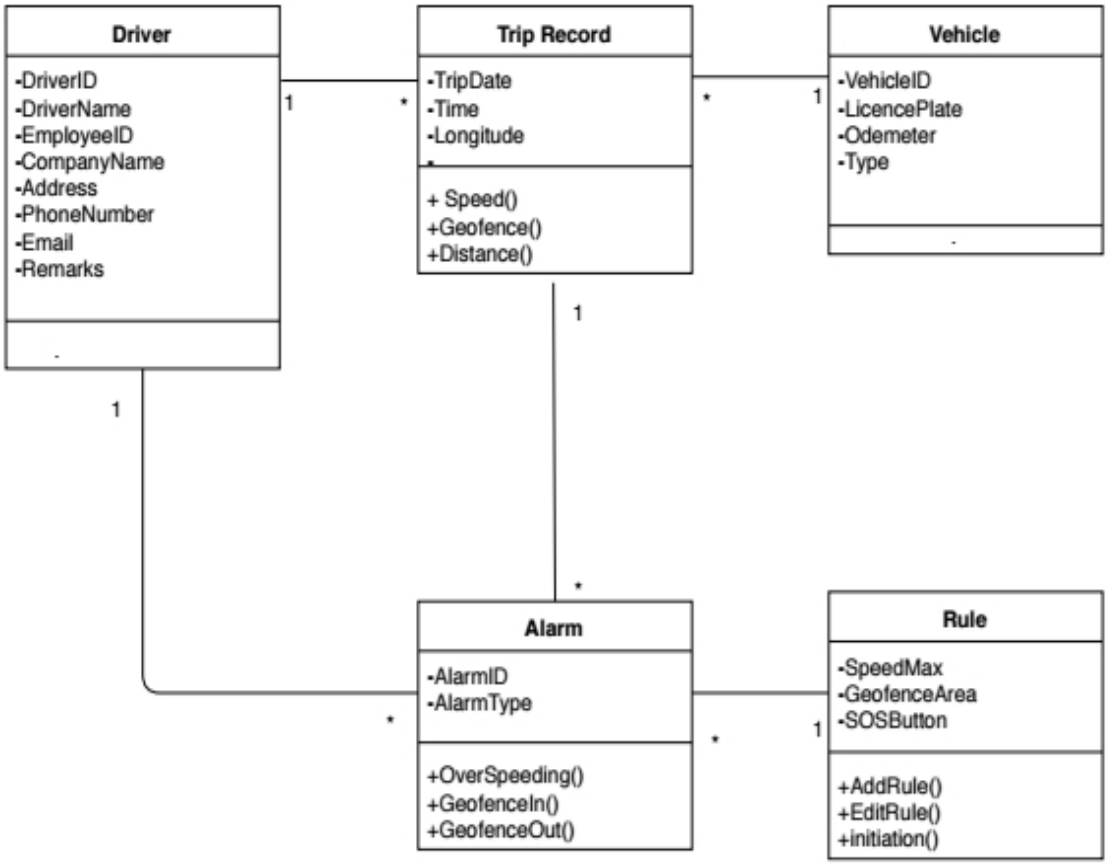

Arrange Category Order
Add Icon
<<extends>>
<<extends>>
Add Task Category
<<includes>>
User
Edit Task Category
<<includes>>
Update User Preferred Categories
Admin
<<includes>>
Delete Task Category
<<includes>>
<<includes>>
Manage Preset Categories
Update Database
Add Icon Resources
<<includes>>

2.4.5.2 Class Diagrams

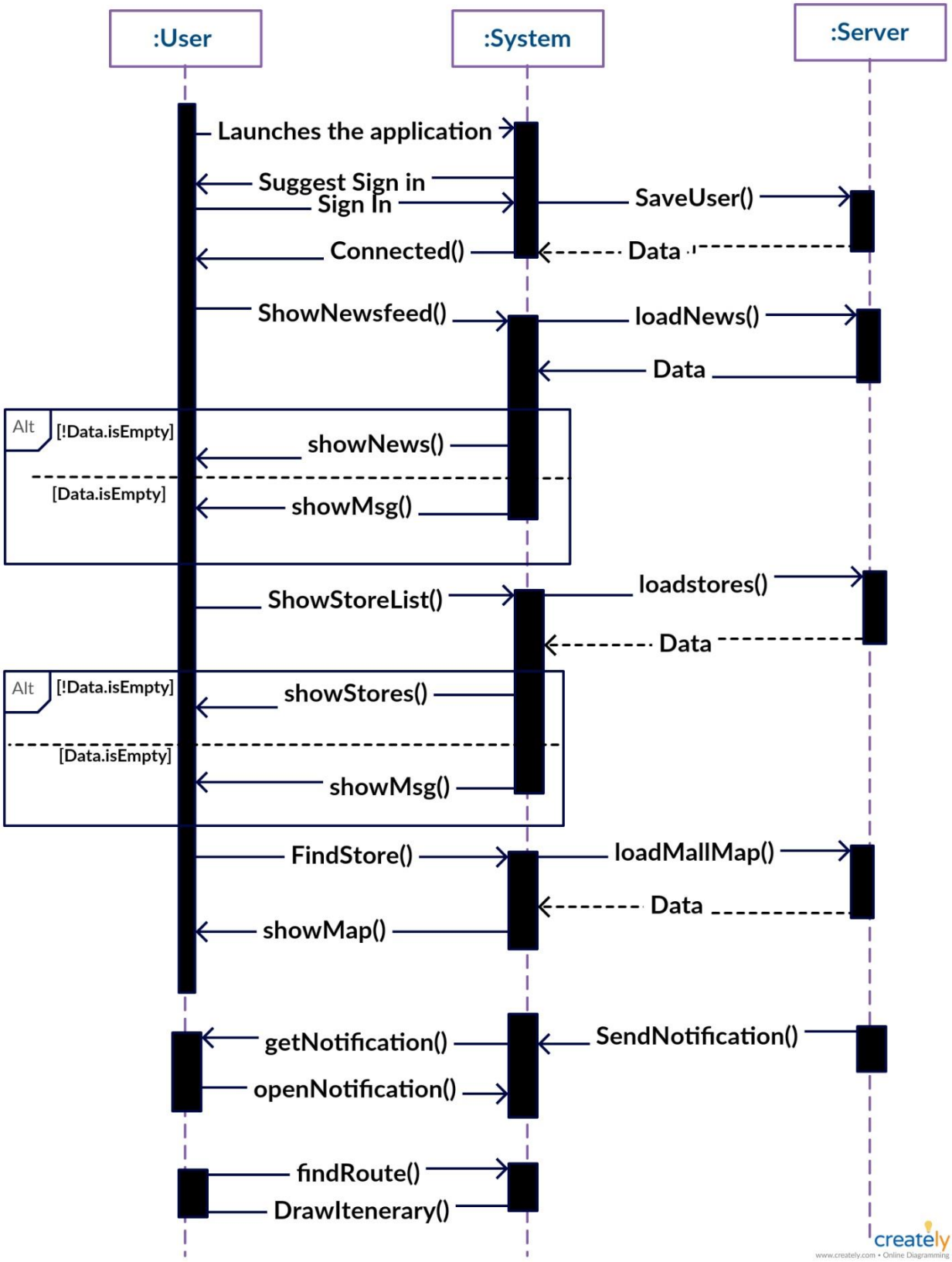


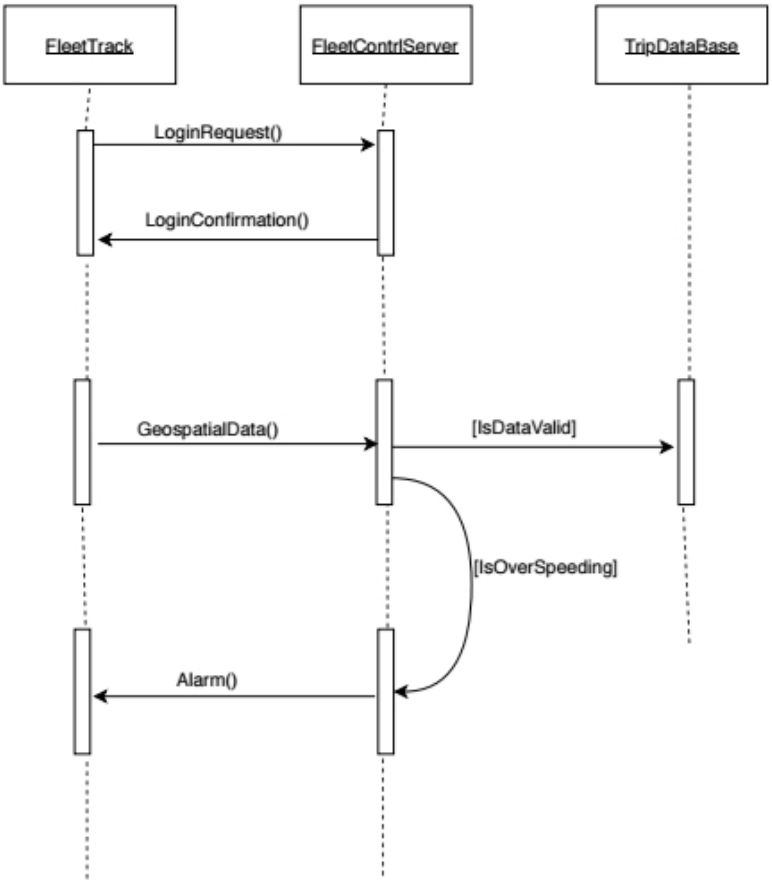

CategoryController
+ categories
+ addCategory(setting)
+ addSubcategory(category1, category2)
+ updateCateory(category, setting)
+ deleteCategory(category)
Category
- id
+ name
+ subcategories
+ icon
+ description
+ getCategory()
+ getSubcategories()
0..1
1
*
Icon
+ resource_name
+ resource_url
+ fetchIcon()
Material_Icon
+ name
+ shape
+ size
Font_Awesome
+ name
+ fontWeight
Ionicons
+ name
+ style
+ size

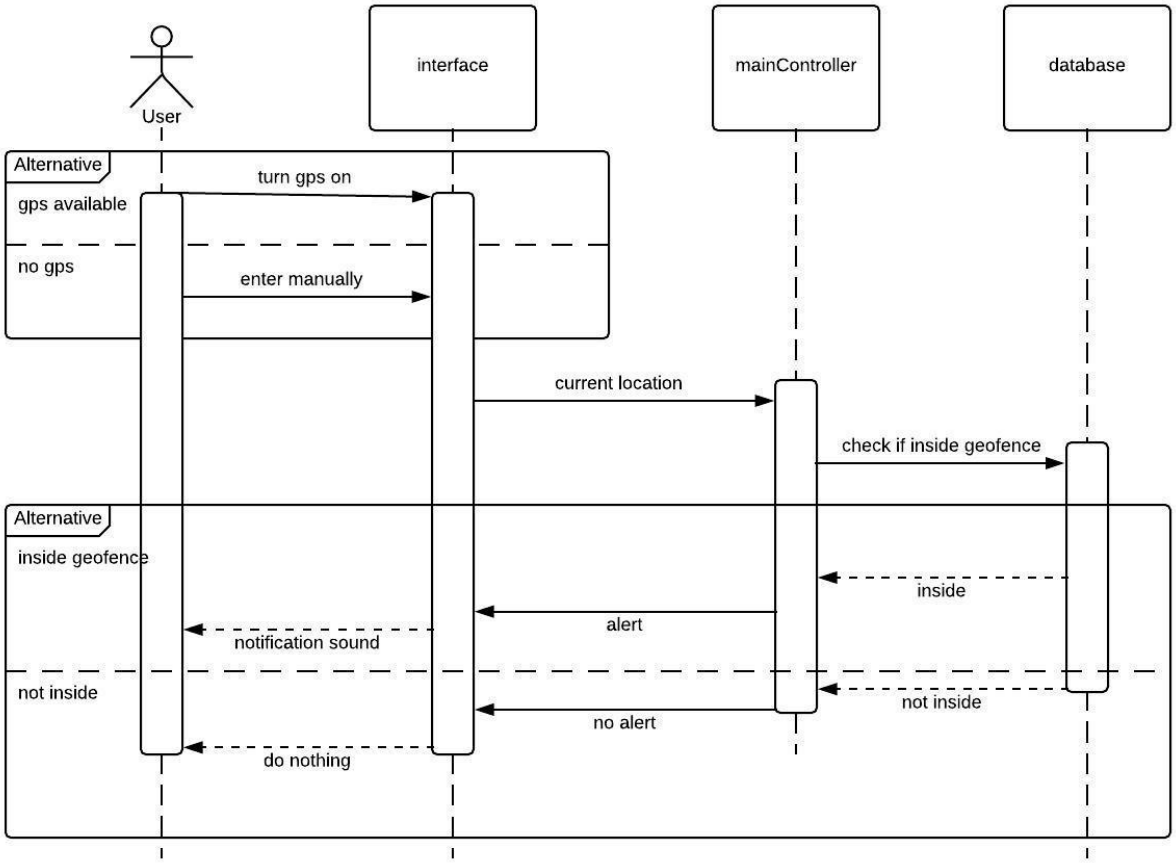

<<User Activity>>
-Map : Google Map
+initalizeMap() : void
+resume() : void
+pause() : void
Location Manager
-CurrentLocation : Location
+getLocation() : Location
+getActivity() : Activity
+sleep() :void
+getGeofense() : Geofence
Network Status
+checkNetwork() : boolean
+setNetworksetting : void
Location
-Latitude : float
-Longitude : float
Marked Place
+Name : string
+Latitude : float
+Longitude : float
+Geofenceid : string
+isMarked() : boolean
Geofence
-geoFenceid : string
-geoLatitude : float
-geoLongitude : float
-geoRadius : float
+Enter() : void
+Leave() : void
1
1

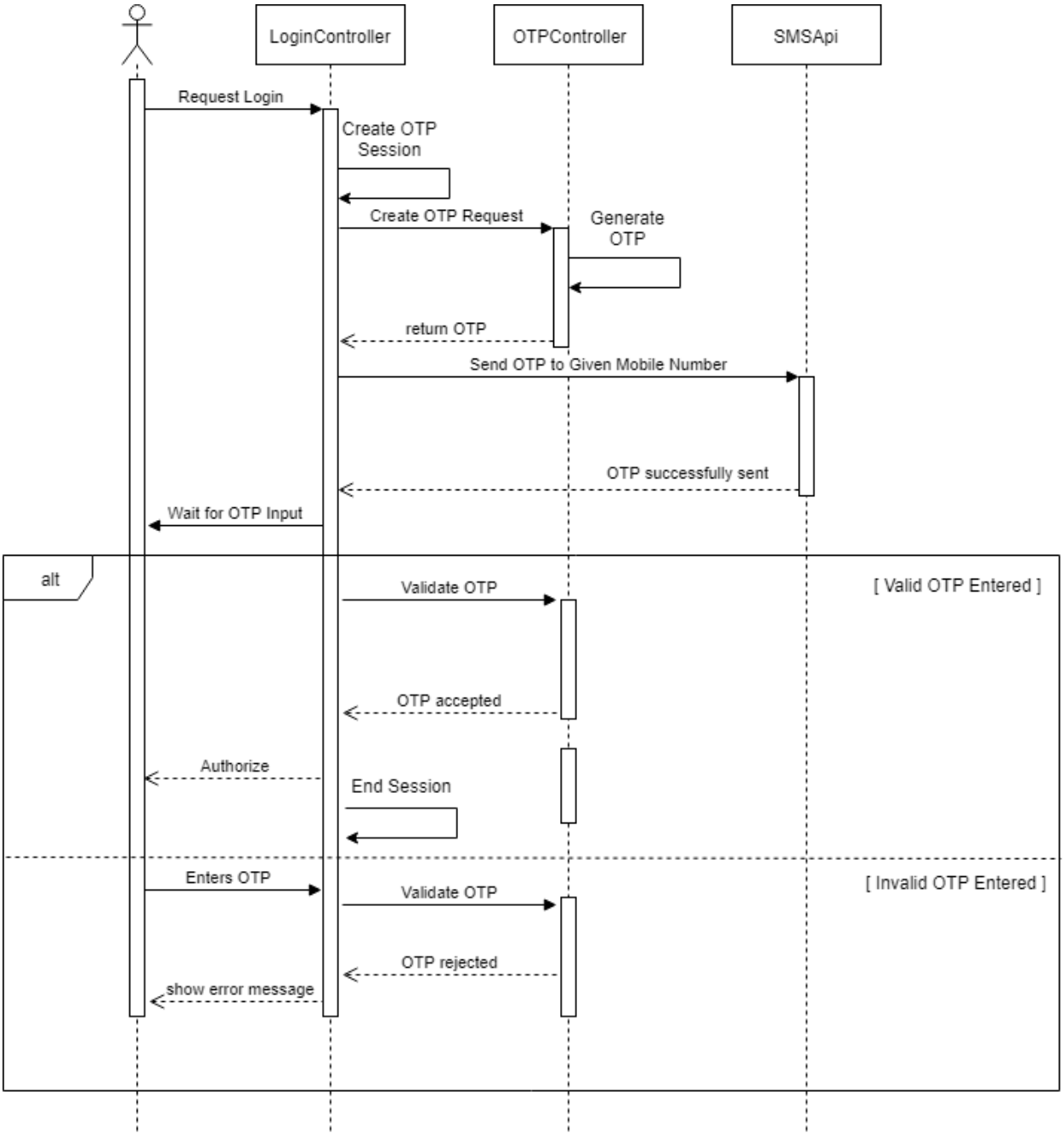

Driver
-DriverID
-DriverName
-EmployeeID
-CompanyName
-Address
-PhoneNumber
-Email
-Remarks
Trip Record
-TripDate
-Time
-Longitude
+ Speed()
+Geofence()
+Distance()
Vehicle
-VehicleID
-LicencePlate
-Odometer
-Type
Alarm
-AlarmID
-AlarmType
+OverSpeeding()
+GeofenceIn()
+GeofenceOut()
Rule
-SpeedMax
-GeofenceArea
-SOSButton
+AddRule()
+EditRule()
+initiation()
1
*
1
*
1
*
1
*
1
*

2.4.5.3 Sequence Diagrams




FleetTrack
FleetContrlServer
TripDataBase
LoginRequest()
LoginConfirmation()
GeospatialData()
[IsDataValid]
[IsOverSpeeding]
Alarm()


User
interface
mainController
database
Alternative
gps available
turn gps on
no gps
enter manually
current location
check if inside geofence
Alternative
inside geofence
inside
alert
notification sound
not inside
not inside
no alert
do nothing


LoginController
OTPController
SMSApi
Request Login
Create OTP Session
Create OTP Request
Generate OTP
return OTP
Send OTP to Given Mobile Number
OTP successfully sent
Wait for OTP Input
alt
Validate OTP
[Valid OTP Entered]
OTP accepted
Authorize
End Session
Enters OTP
Validate OTP
[Invalid OTP Entered]
OTP rejected
show error message

2.4.6 Test Cases

Software Testing

In order to get a good software that can do all the things that we want the software to do, we need to implement different types of software testing tools. Also, in order to get a software with as less bugs as possible we need to do testing throughout the software building life cycle. Below are all sorts of tests that we have tried to do :

Unit Testing

1. **testSilentAlreadyExist() :** This method will try to add a silent place to the database that already exists.
   Return Type : Boolean
   Output : true if the system gives an error and do not let to add the same place twice. False otherwise
2. **testAddSilentPlace()** : This method will add a silent place and check whether that place was added to the database successfully with the help of specific sql query
   Return Type : boolean
   Output : True if the result from the sql query matched with the added place information and false otherwise

Security Test

The system will not let any user to login into account for one hour if the user has entered wrong password three times in a row . We will write an script that tries to login with wrong password three times in a row and check what happens
Output : Prints an error message if unable to login after trying wrong passwords for three times in a row.

Install/Uninstall Test

A script that will install and then uninstall the app. Then again reinstall it and login to a dummy account and check basic functionalities like add a place reminder, add a new note etc
Output : Prints an error message if the any functionalities are not working like login function. Do nothing otherwise.

Unit Tests

A class named “UnitTest” is designed which contains the function for each unit test.
● **testValidLogin**():
Return type: Boolean
Input: This method signs in with a valid username and password i.e. a registered username and password that satisfy all the constraints. For

instance: A constraint is applied on password by specifying that it must contain a number and must be at least 3 characters long.
Output:
1. Pass: If successfully logs in.
2. Fail: If login fails.

● **testInvalidLogin**():
Input: This method signs in with an invalid username and/or password. For example: A username/password that is not registered in the database, or violates a constraint.
Output: Prints an error message if the login was successful.

Load Test

For load test, a class named " LoadTest" is created.

Fields: The class contains a static field "max_user". It keeps track of the number of users operating at the same time.

Functions: Contains a function called testLoad() .The function tests how many concurrent users can the system handle. It creates an increasing number of threads that are started in parallel. Each thread would create a dummy user, log in and carry out a set of tasks that represent regular usage of the system.

Output: After the process causes system failure, it returns the static field "max_user".

Black Box Tests

Update/Add Place

A script will attempt to update or add place info from a "User".
account. In this case, it checks the specific HTML element to see if the action was completed.
Output: If action is not completed from a "User" account it prints an error message.

System Testing

To ensure web service could be accessed from all browsers we checked by loading it in different
browsers. In a specific browser (Opera) it showed a issue as such the page did not show as it should show, instead some tabs with crucial features would not fit to the users view. Thus after this test we provided option to zoom in the website so that opera mini users can zoom in and view all tabs.
Moreover, we issue the instruction whenever the website is accessed from that particular browser.

Unit Testing:

Notification:
Tests notification for friend requests, comments, likes. The method has the following common format:

● testNotification():

An SQL query performs an action that should generate a notification.
Another query fetches the notification table.
Output:
Prints an error message if the content of the fetched table does not contain the expected notification or if the table is empty. Otherwise prints confirmation.

Integration Testing:

For integration test, bottom-up approach would be used mostly. SInce the system is developed as designing components and integrating them, this test will ensure that the system works after each component is added. A class names “IntegrationTest” would be use. New methods would be added to the class as the system is developed and features are added. Each of the methods executes one functional unit. One method “testSystem()” would sequentially execute all the methods. If a failure occurs at any stage of the execution, it prints an error message depending on where the error occurred.

Black Box Testing:

Update Address:
A script will attempt to update address from a user navigation account. In both cases, it checks the specific HTML element to see if the action was completed.
Output:
If action is completed from a “user” account or not completed from a “Developer” account, prints an error message.

Beta Testing:

Since the development process follows agile model, new versions will be released frequently. Before releasing new versions with major changes, a beta version of the system would be released to get user reviews and refining the system accordingly.

Stress Testing:

We have encouraged 100 users to login simultaneously in our application. No performance degradation was noticed.

Integration Testing:

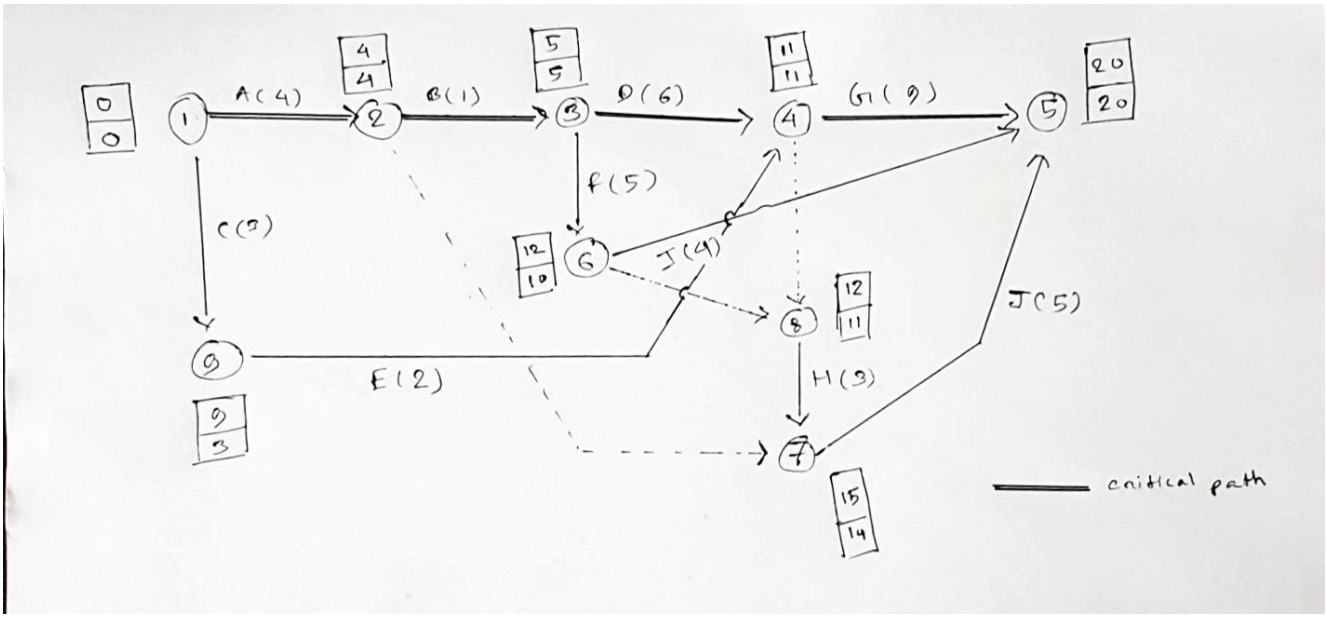
We have made a code to add a user, add multiple tasks with different category with that user. This automated test will test for bugs in the code.

3. PROJECT SCHEDULE

4.1. SCHEDULE DATA

Task	Precedence	Duration	Staff	Slack Time	Staff Week
A	-	4	4	0	16
B	A	1	5	0	5
C	-	3	2	6	6
D	A,B	6	3	0	18
E	C	2	7	6	14
F	A,B	5	4	2	20
G	D,E	9	3	0	27
H	D,E,F	3	2	1	6
I	F	4	6	6	24
J	A,H	5	3	1	15

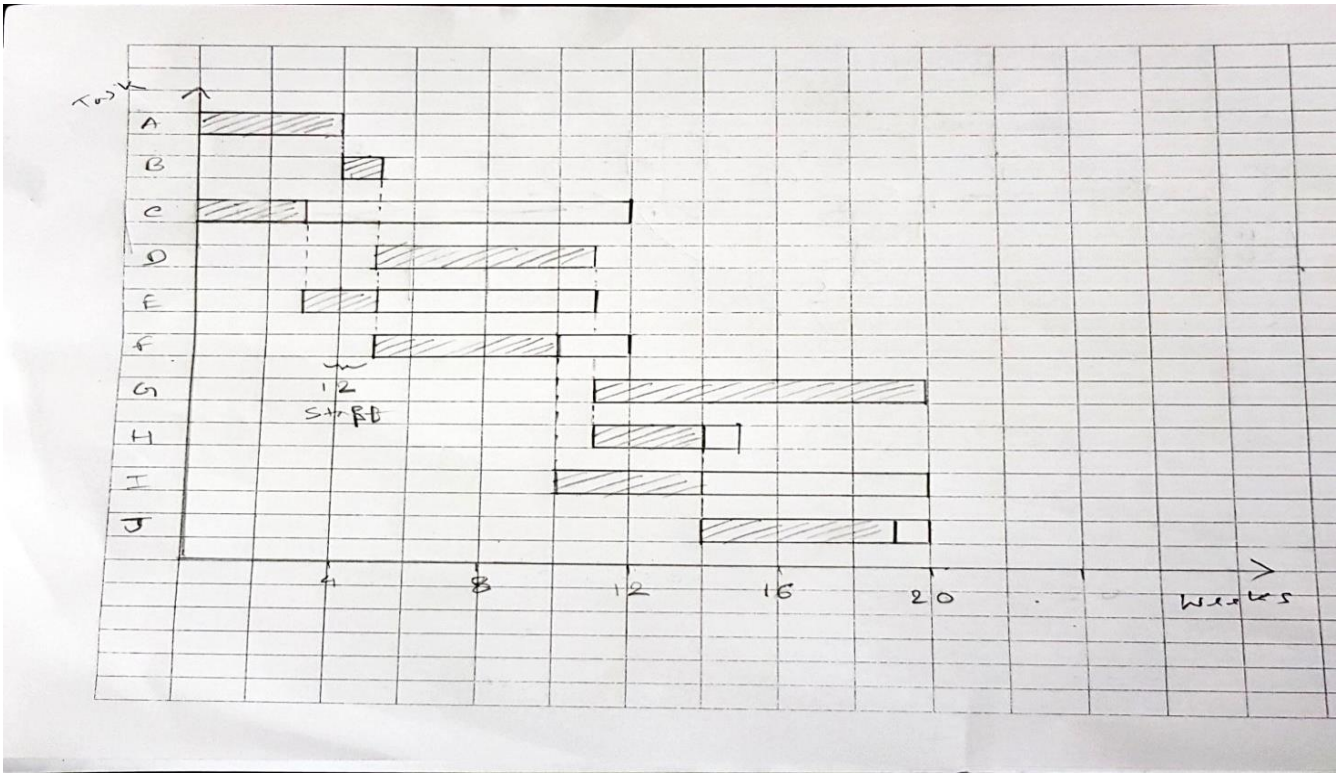
4.1. PERT DIAGRAM



We have divided our project into several different small tasks and some of these tasks are dependent on one another meaning that we cannot start a task without completing the task that it is dependent on. We used CPM and GANTT chart and found out that our earliest completion time would be 20 weeks and we would need to hire minimum 12 staffs for it.

According to the GANTT chart, we can delay task C,E,F,H,I and J a little bit without affecting our overall completion time but we cannot delay the rest of the tasks since they are our critical tasks.

4.2. GANNT CHART



4.3. STAFF UTILIZATION

Total staff week = 151 staff-week
Estimated project length = 20 weeks
Total employee hired = 12
Hence ,
Staff utilization = 151 / (20 x 12) x 100

= 62.9 %

4. MILESTONES

4.1. FIRST PUBLIC BETA

Release the first public beta on the 1st of January of 2019. This release will contain only the core services. Precisely this contains the core features of geofencing technology and simple pop up notification system at the very beginning.

4.2. FEATURE UPDATES AND BUG FIXES

Over the course of the following weeks, our project will be under rapid development with frequent updates that add features and fix bugs. Those updates will be based on users recommendations. If any bugs occur in the system we will fix this as early as possible.

4.3. EXPAND TO OTHER PLATFORMS

By December 2019, project will be released on all major platforms (i.e. Web versions on Windows, Mac, Linux and mobile apps for iOS and Android devices).

4.4. LONG TERM SUPPORT

In the long run, we will incorporate the project and analytics tool for the users and keep the software up to date for a long period of time

5. RISK ASSESSMENT

5.1. RISK FACTORS

5.1.1 Project Risk

- Risks of having faulty project resources or bad scheduling like getting broken data or inaccessible user information.
- Loosing of an potential employee or key employee falling sick can costs us lot of time which can create delay in project delivery.
- Faulty code in between of the process implementation can put us into lots of trouble, it's simple make the whole process vulnerable

5.1.2 Product Risk

- Choosing the right type of database system is a challenging issue. Faulty database can costs us the deletion of user identification or information, thus we will be losing the track of that particular customer.
- Again bad design can creates complexity among users to use our system.

5.1.3 Business Risk

- If we build a bad system it will definitely creates a bad impressions on our customer about our company so we have to be very careful. This a great risk for the future of our company.
- We need to keep track of what's our competitors are doing in order to grab the customer attention. Their better products can make our product less popular.
- We need to evolve our system on the basis of upcoming requirements as situations are not always the same.

5.2. POTENTIAL RISK ASSESSMENT

Risk	Probability
Project schedule will exceed one fiscal year.	**Low**
Project team member(s) will not be in place when required.	**Medium**
Chance that the workstation environment of the intended user will change after requirements are gathered.	**Low**
Chance that changes, in critical personnel on either the government or contractor side, will occur during the life of the project. (As it is Bangladesh)	**High**
Risk to the project resulting from a mandated/mandatory completion date for the project.	**Low**
Risks associated with personnel assigned to the project who may be pulled off anytime for another assignment.	**Low**
Risks to the project caused by requirements that are inadequately defined.	**Medium**

6. REPOSITORY

GITHUB REPOSITORY LINK:

https://github.com/colossus21/CSE470-Final-Report

7. EFFORT BREAKDOWN OF MEMBERS

	Syed Zafrul Hassan	Abu Wakkas	Rafiul	Nusayer
Introduction	✓			
Proposed System			✓	
Project Schedule				✓
Milestones		✓		
Risk Assessment		✓		
Repository			✓	
UML Diagrams, Test Cases	✓	✓	✓	✓