ENTRANCE SECURITY SYSTEM

CSE470 FINAL REPORT

Team Members

Mir Ayman Ali - 15101104
Chayan Saha - 17101415
Nuhash Ahmed - 1630000

Submitted to:

Zahidur Rahaman
Professor
BRAC UNIVERSITY

Table of Contents

1. Introduction

Our project is on Entrance Security System.

Entrance Security System is now a crying need for every organization, institute and so on. To ensure 100% security, every important buildings should have this type of system. The goal of Entrance Security System is to control the individuals who are gaining permission for entry. With this, it also becomes possible to determine the purpose for which an access control system is used.

1.1. Purpose of the system

Whether it is your work place or your home, it is always necessary to restrict the entry of unwanted people in the premises. For this use of manual security system had been in use for decades now but they had certain limitations. It is now possible to overcome those limitations by using automated Entrance Security System. They are now electronic devices that are controlled and maintained by computers. The software is fed with different data and these access system decides whether to permit or not someone for passing through the entry. However, you will find much more access control purpose. They are discussed below.

- Apart from maintaining the basic purpose of controlling the people using the passage it also reduces the man power that is required for controlling the entry and exit.
- It ensures security in the premises and the main goal of access control is to minimize the risk of unauthorized access of individuals to some workplace. Whether it’s a computer data or entry and exit to a particular place, say a laboratory, with the help of access control it is possible to ensure the security of the place better than using any manual system.
- An Entrance Security System also allows internal levels of security. It makes it possible to give access to one individual for any particular level and then if required cannot allow them further to any more level. This can be easily controlled with the software that will make it easier for the management to control access of their employee or even outsiders.
- Another purpose Entrance Security System is to have a check on the time when the entry and exit is made. It makes it possible to keep know even in future about the timings when someone has taken entry or has left the premises.
- An Entrance Security System is also used for taking control where there are multiple entry and exits. With the system working anyone can enter the premises from any point but that will be recorded in the system.

While discussing the Entrance Security System purpose it is clear that an Entrance Security System is a better security system for any organization. Thus, such systems are implemented in private offices as well as government offices. They ensure that the entry and exit of individuals be that employees or visitors is not only controlled but recorded. It gives someone authorization with which they can enter at the required time without even worrying about getting permission from someone as they are authorized for doing so.

1.2. Scope of the system

The fundamental use-cases Entrance Security System include:

- The Entrance Security System shall be located at the main entrance of each building and shall include, but shall not be limited to the following:
- Computer (CPU, flat panel monitor, mouse, keyboard, etc.), card readers, battery backup, data converters, power supplies, badge printing system to print cards for use by staff and faculty members, all hardware and software required for a complete and ready to operate system.
- The system shall be a large networked system for multiple school buildings with the capacity to track when card users come and go.
- The Entrance Security System shall connect to the existing monitored alarm systems and CCTV systems in each school and provide authorized individuals safe and secure access in and out of the building (s) for which their cards are programmed.
- The system shall provide programming of the Entrance Security System software for the door and the interlock functions and provide for integration between the ACS and the fire alarm system.

1.3. Objectives and Success Criteria of The Project

The usage of our system involves a secured and safe entrance system to ensure proper management of all the people entering or exiting the building.

Although our system is usually used for secured entrance, but it’s important to note that this technology can also be used by employees to track their employees, guests arrival and departure time and date.

Other benefits of using our system include:

- The ability to set up an alert for any unauthorized entry to the building, which is very important.
- Ensuring employees are on time in their workplace by checking their arrival and departure time.
- Allowing the employer to collect and view data on number of people passes through the entrance system.

1.4. Overview

Our system is a method of defining a secured entrance system of any institute.

It is a superb technology with revolutionary potential to transform. It can prompt mobile push notifications, trigger text messages or alerts, security intelligence and verify any past visit of employees or guests 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 Java script 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 some third party authorized software to provide best result.

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 permanently

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.

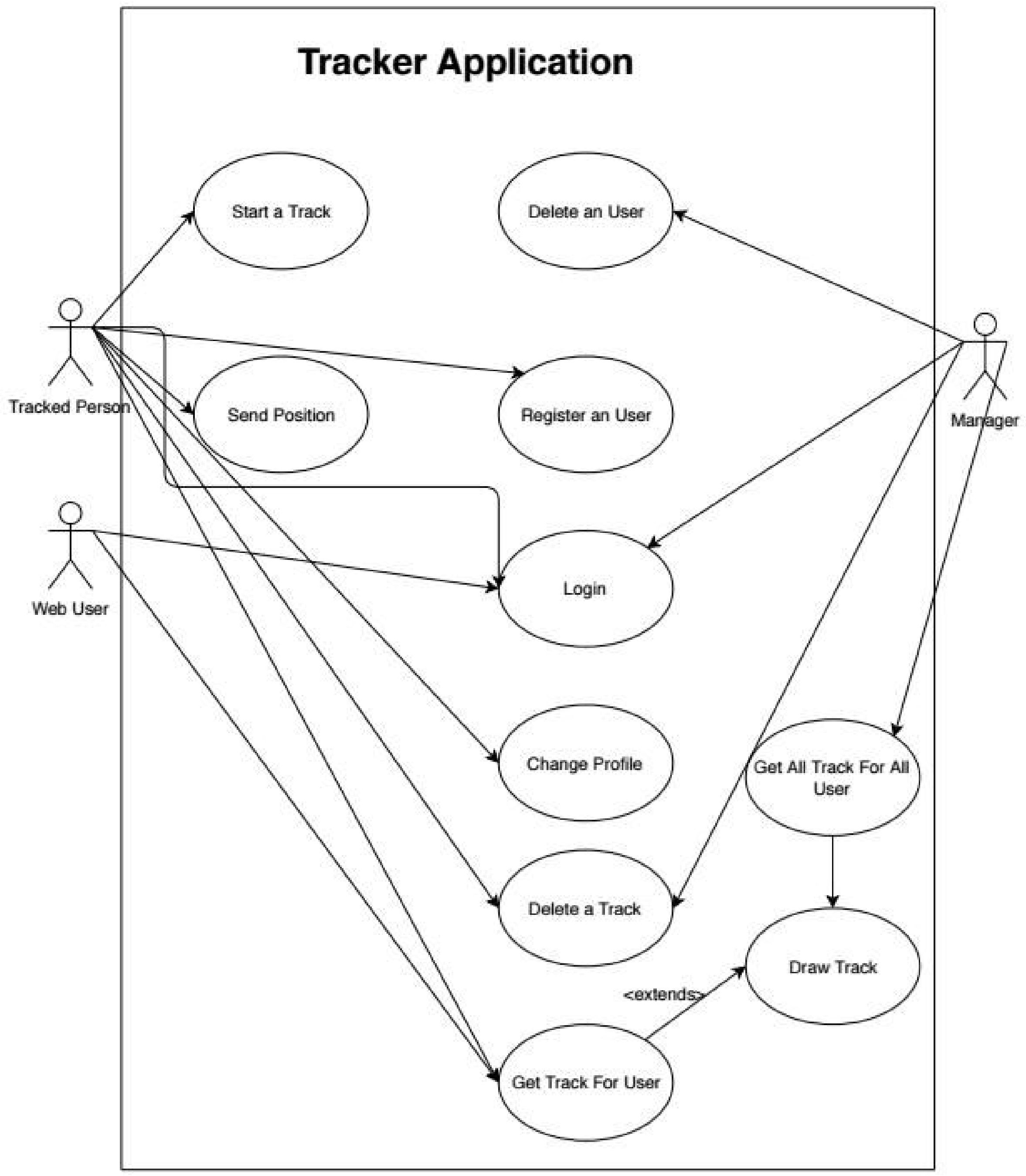
2.4.4 Controller

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.

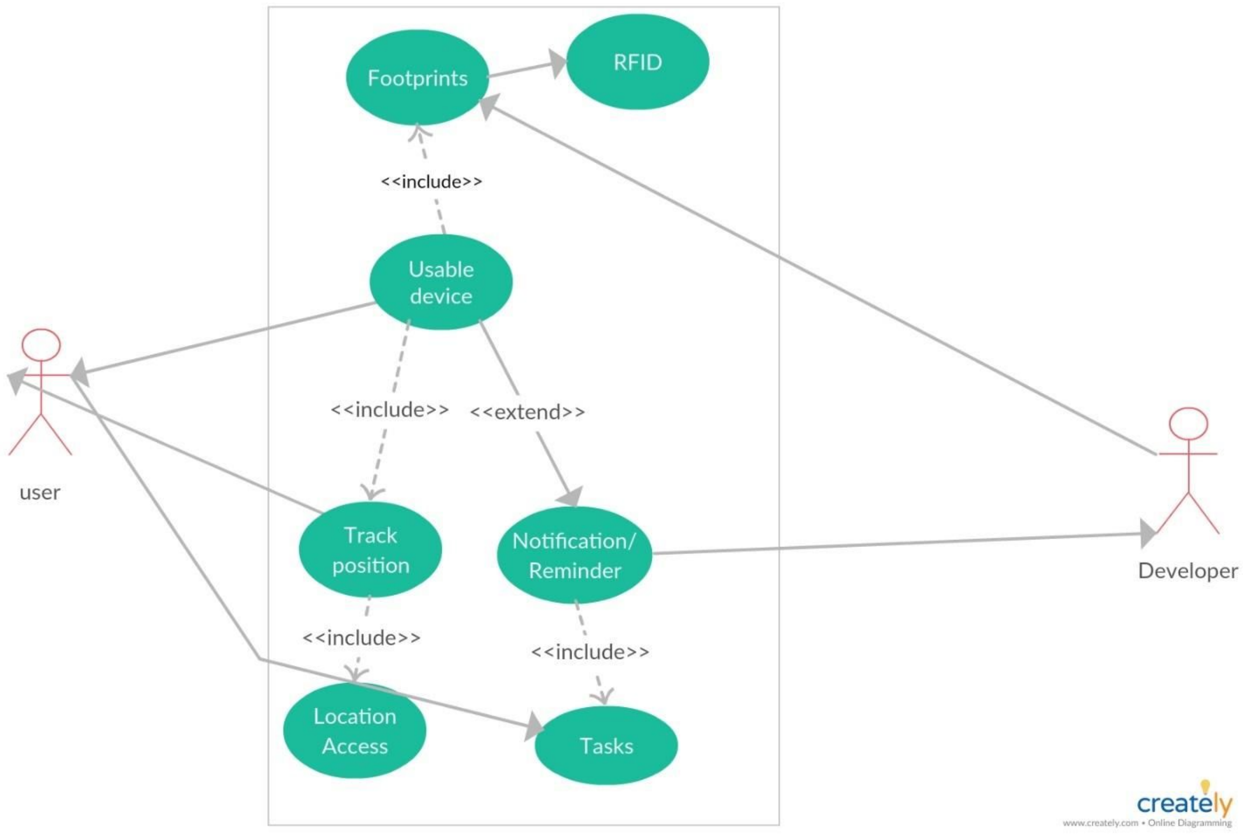
2.4.5 UML Models

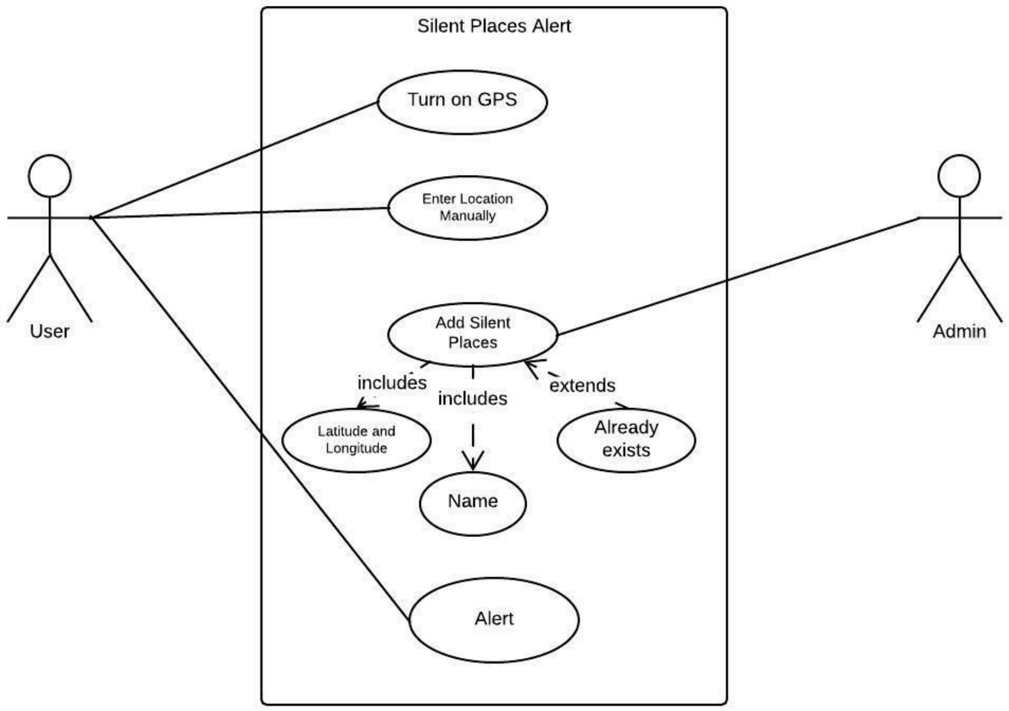
The following pages contains User Case, Sequence and Class diagrams for this application.

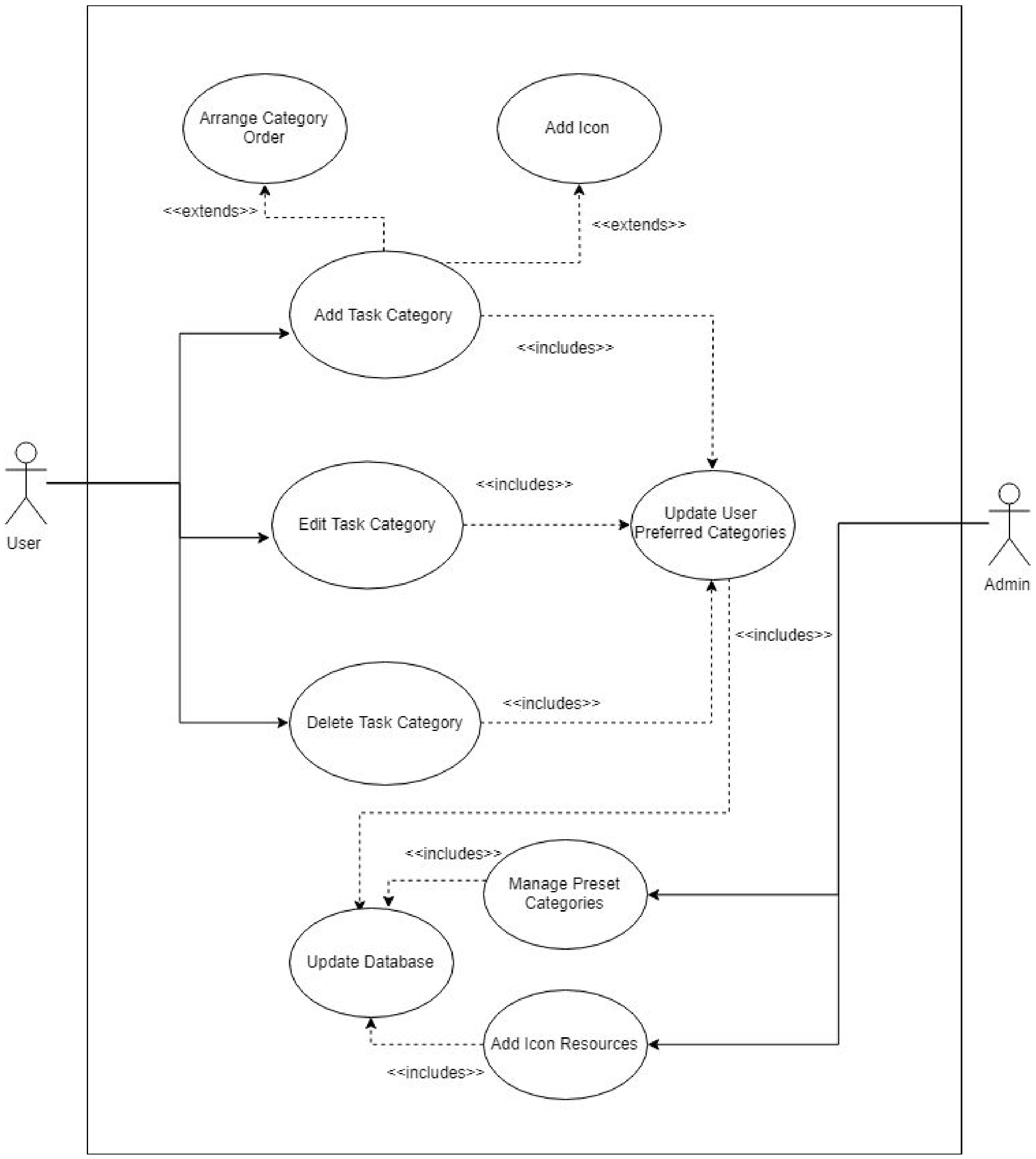
2.4.5.1 User Cases



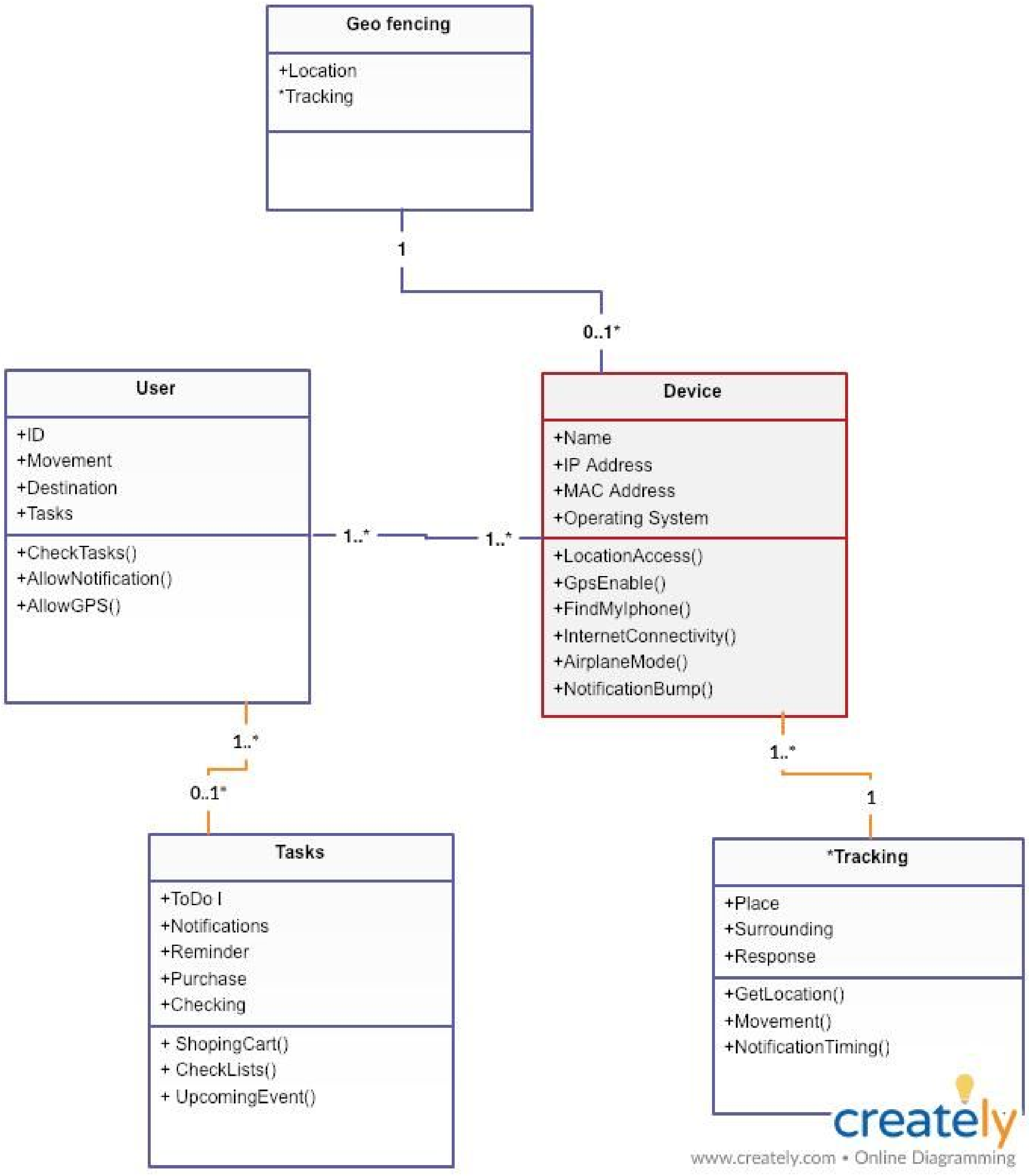
Use Case Diagram

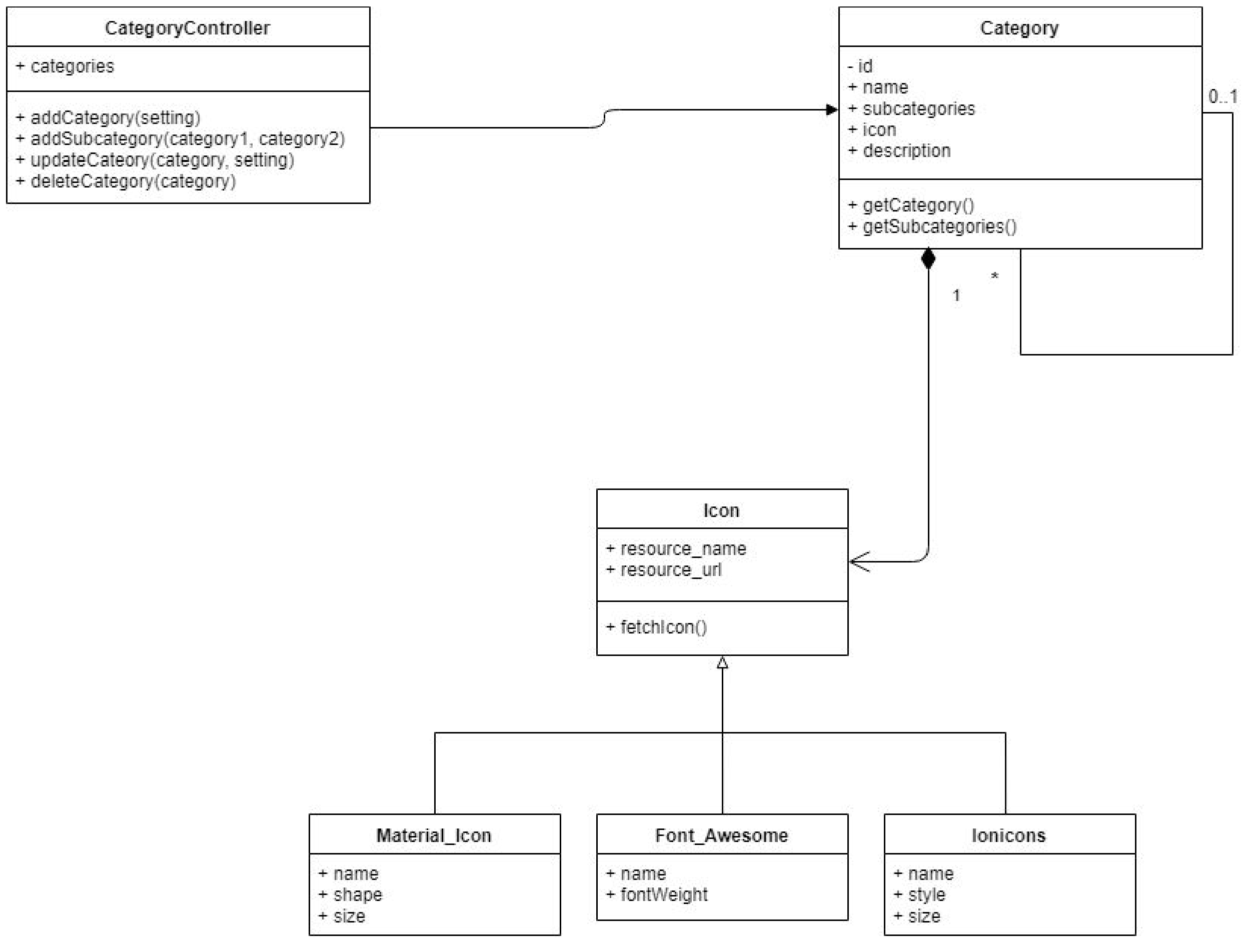


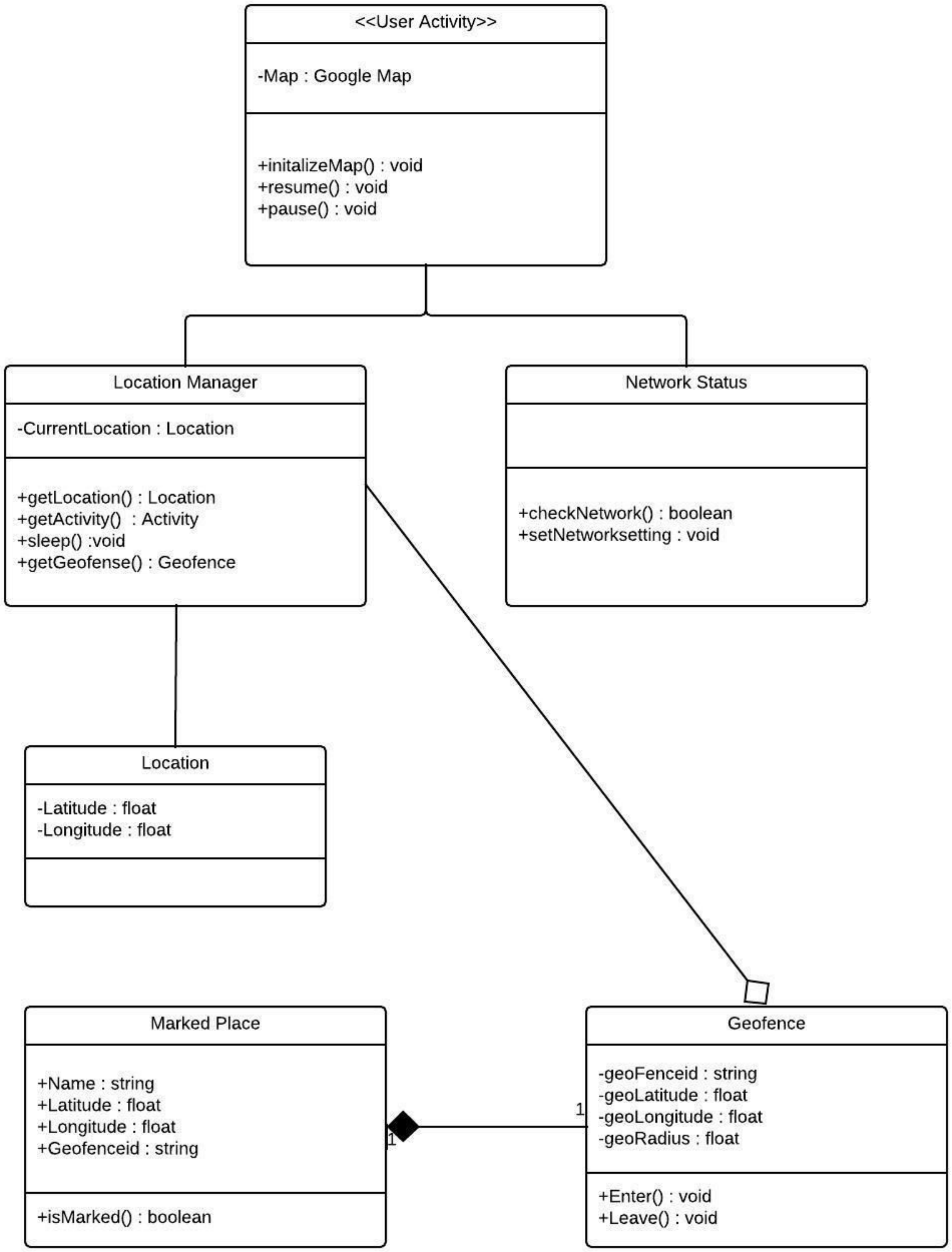

Silent Places Alert
Turn on GPS
Enter Location Manually
Add Silent Places
includes
includes
extends
Latitude and Longitude
Name
Already exists
Alert
User
Admin


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>>

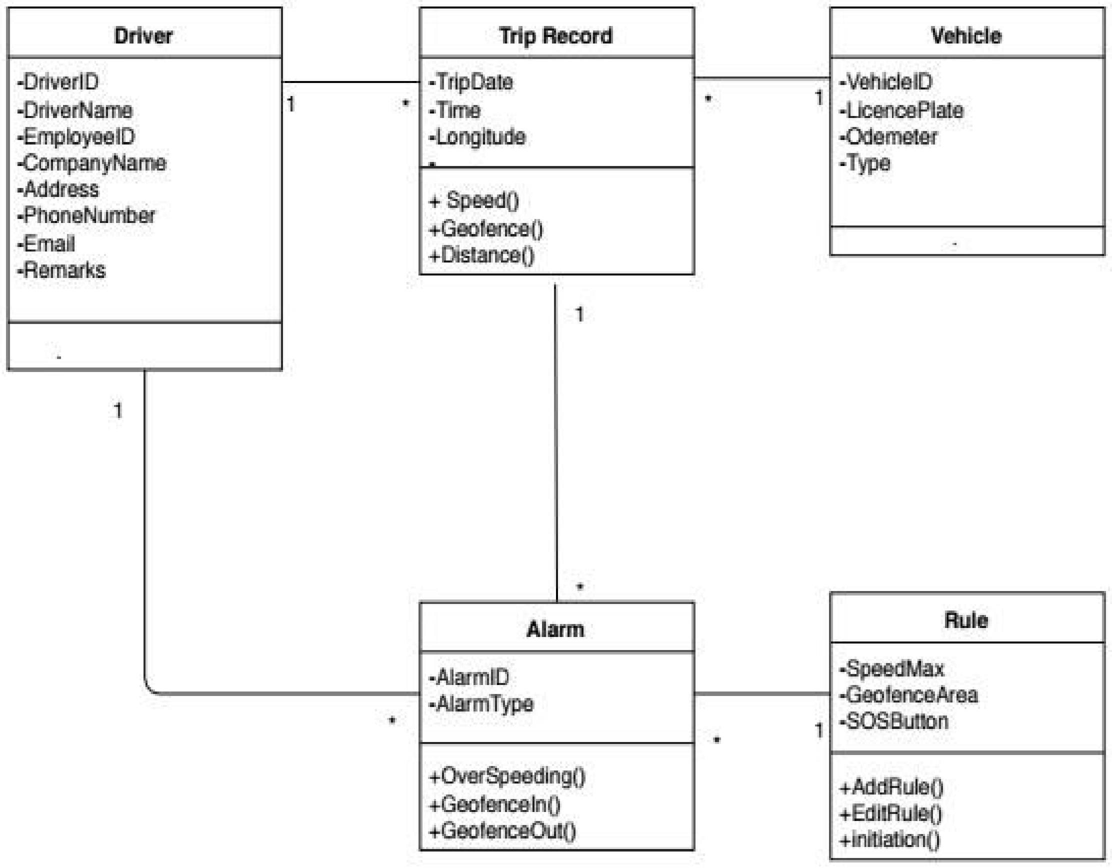
2.4.5.2 Class Diagrams



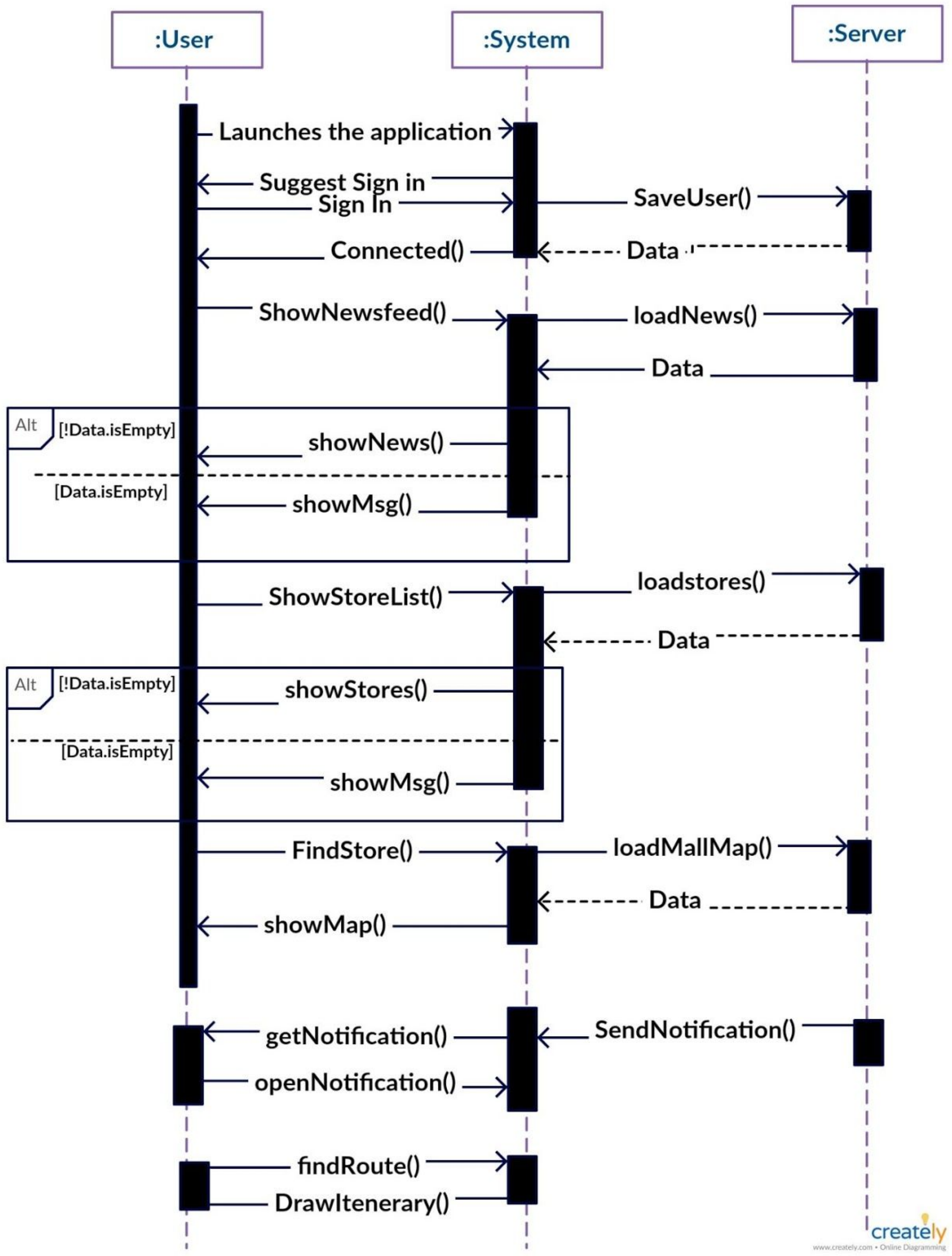

CategoryController
+ categories
+ addCategory(setting)
+ addSubcategory(category1, category2)
+ updateCateory(category, setting)
+ deleteCategory(category)
Category
- id
+ name
+ subcategories
+ icon
+ description
+ getCategory()
+ getSubcategories()
0..1
1
*
Icon
+ resource_name
+ resource_url
+ fetchIcon()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

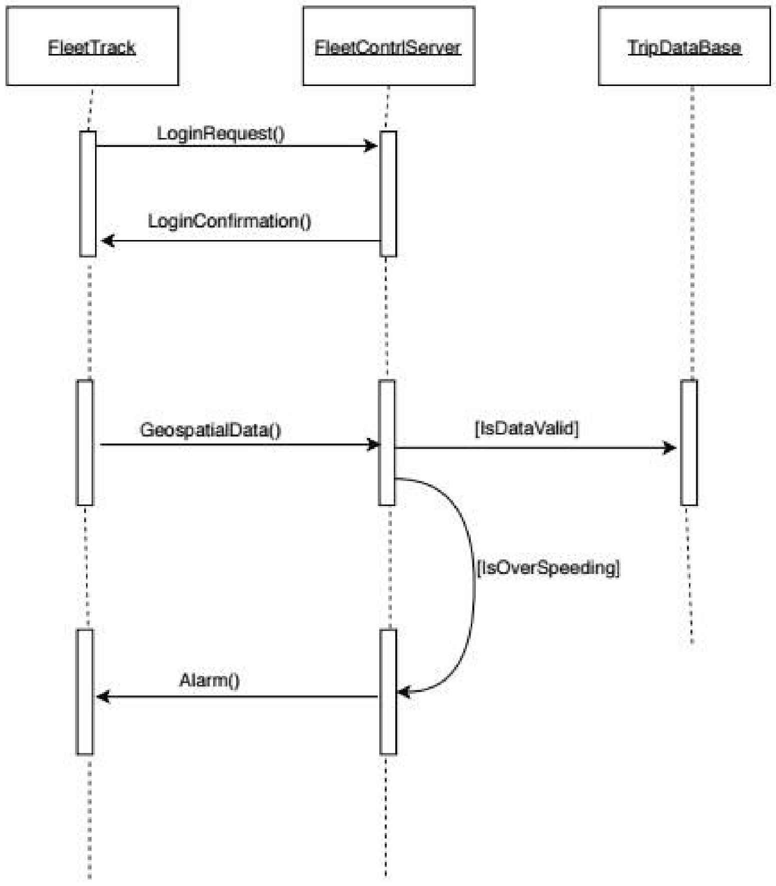


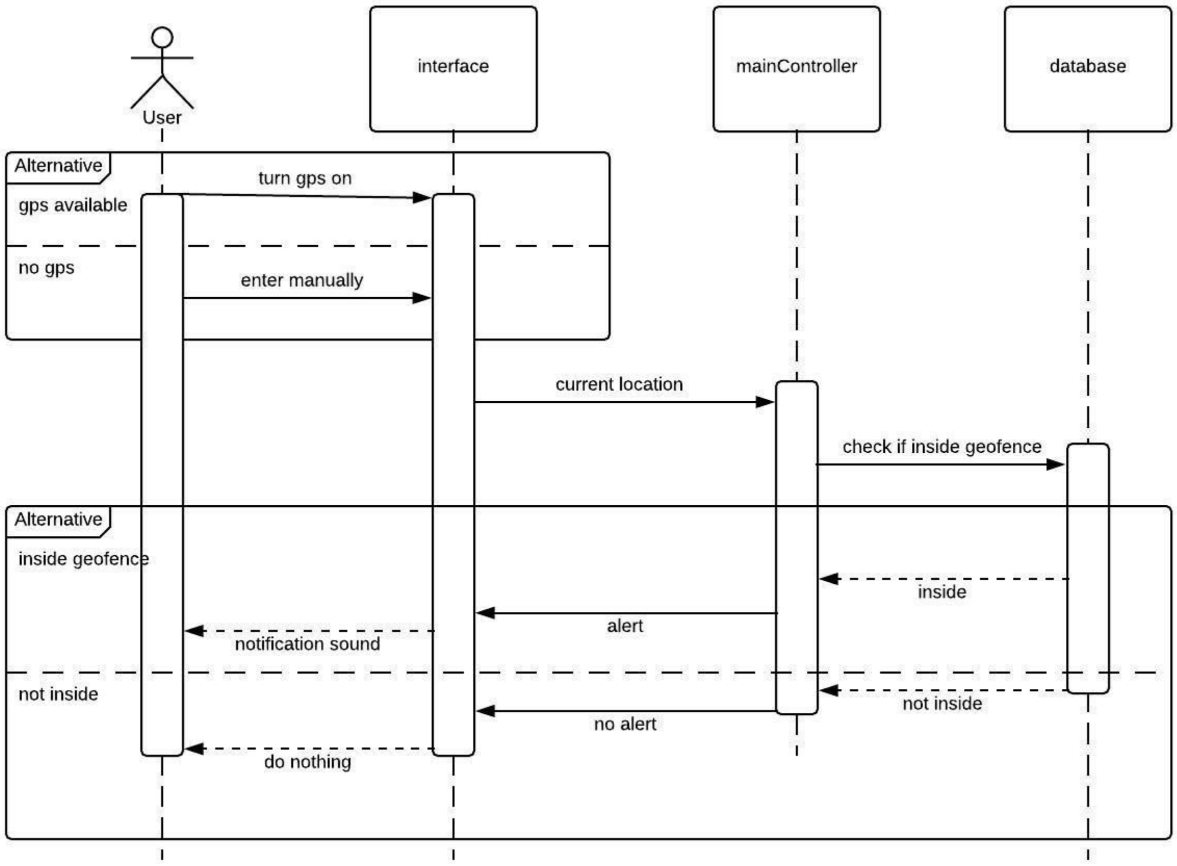
<<User Activity>>
-Map : Google Map
+initalizeMap() : void
+resume() : void
+pause() : void
Location Manager
-CurrentLocation : Location
+getLocation() : Location
+getActivity() : Activity
+sleep() :void
+getGeofense() : Geofence
Network Status
+checkNetwork() : boolean
+setNetworksetting : void
Location
-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

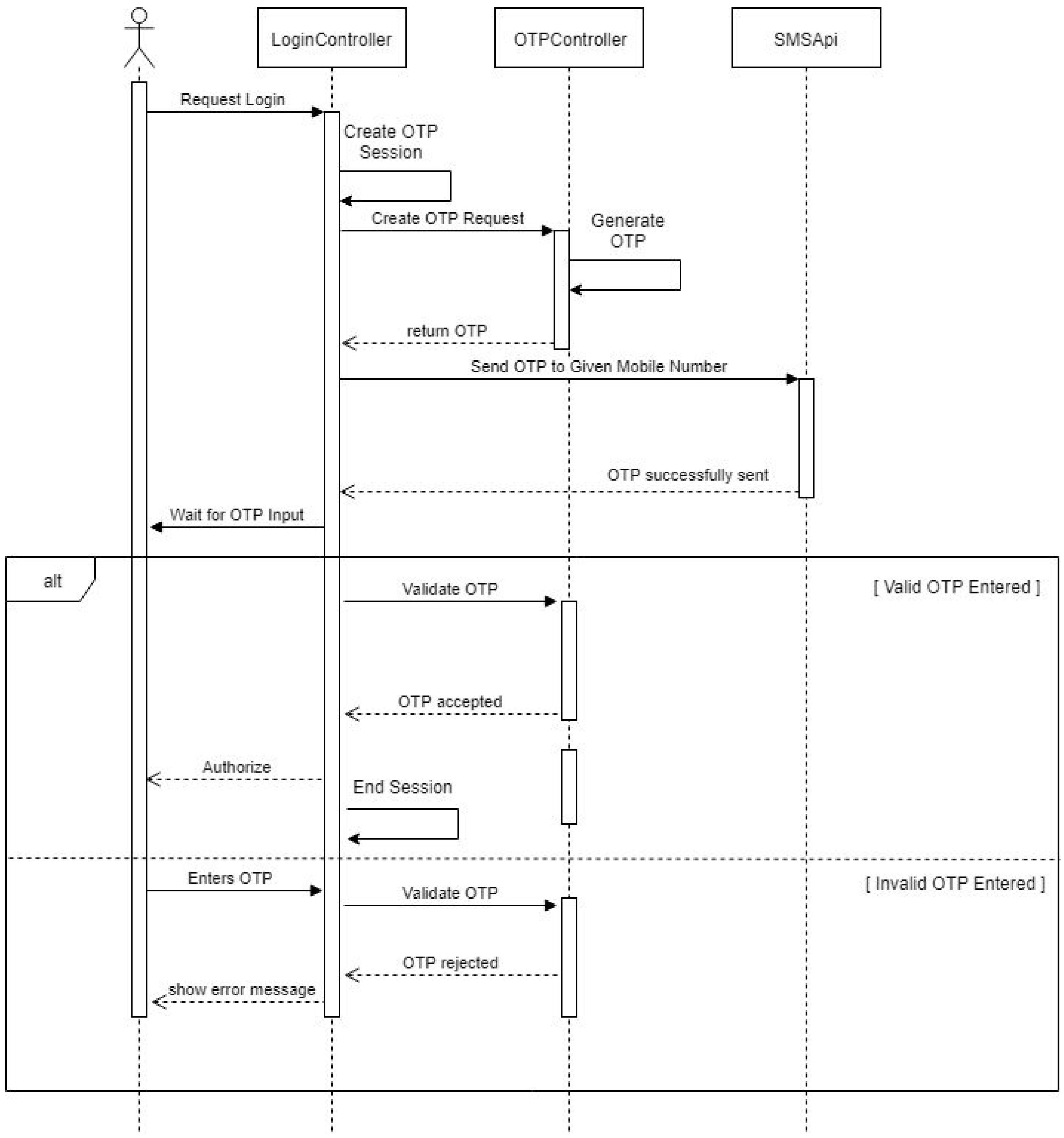

Driver
-DriverID
-DriverName
-EmployeeID
-CompanyName
-Address
-PhoneNumber
-Email
-Remarks
Trip Record
-TripDate
-Time
-Longitude
+ Speed()
+Geofence()
+Distance()
Vehicle
-VehicleID
-LicencePlate
-Odemeter
-Type
Alarm
-AlarmID
-AlarmType
+OverSpeeding()
+GeofenceIn()
+GeofenceOut()
Rule
-SpeedMax
-GeofenceArea
-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
[Valid OTP Entered]
Validate OTP
OTP accepted
Authorize
End Session
[Invalid OTP Entered]
Enters OTP
Validate OTP
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

1. **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 anyone to get into the building for five minutes if the user has done anything wrong three times in a row.
Output : Prints an error message.

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:

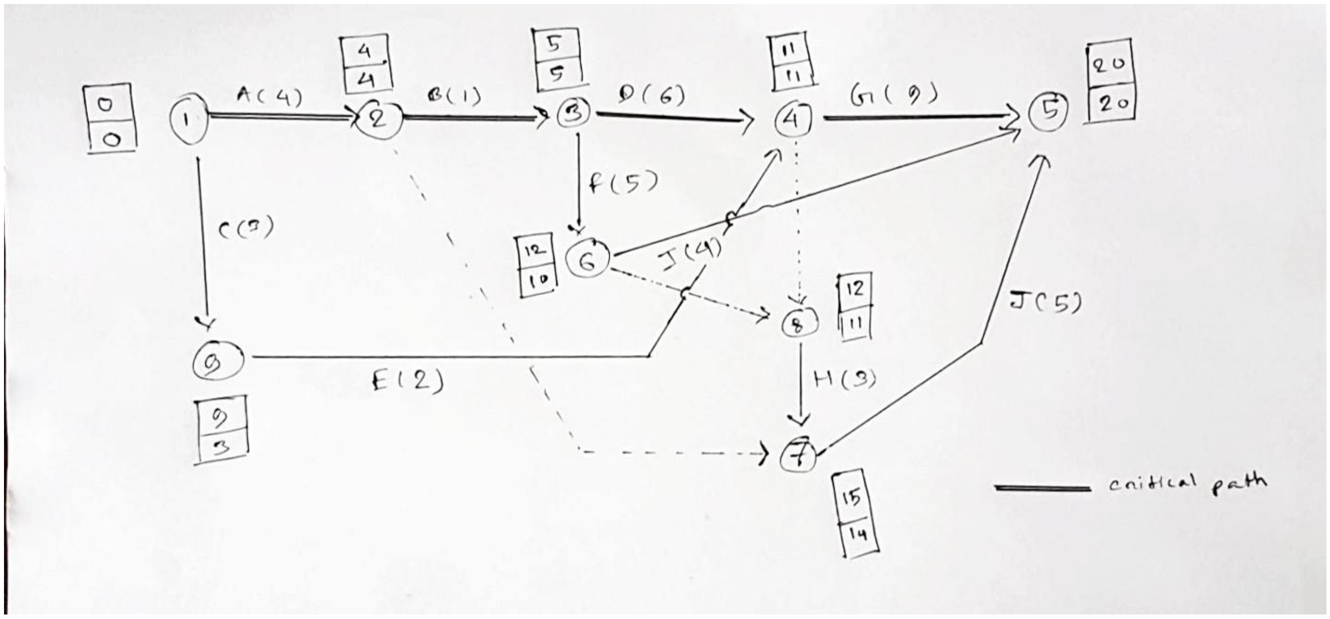
We have made a code to add a user, add multiple tasks with different category with that user. This automated test will test for bugs in the code.

3. Project Schedule

4.1. Schedule Data

Tas k	Precedenc e	Duratio n	Staf f	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

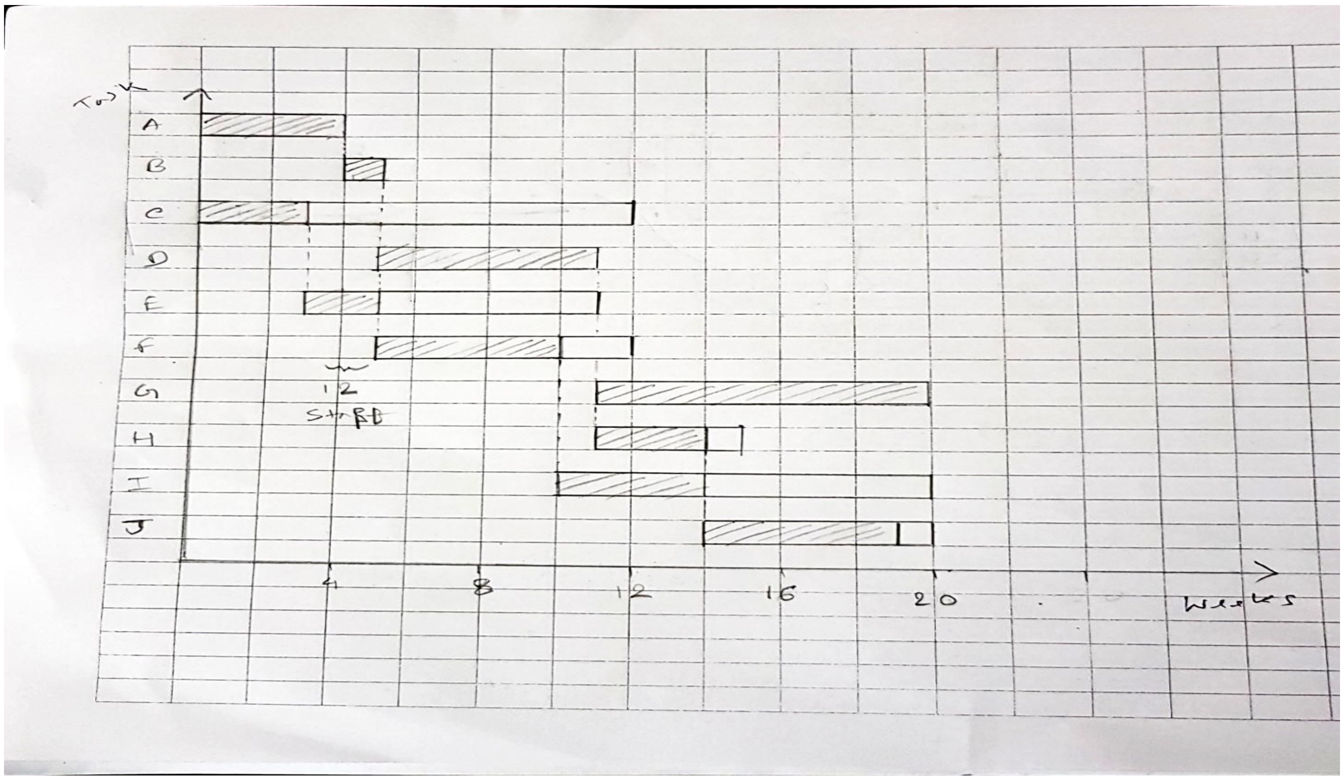
4.1. PERT Diagram



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 2020. 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 2020, 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 a 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 create 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 is 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:

7. Effort Breakdown of Members

	Chayan Saha	Mir Ayman Ali	Nuhash Ahmed
Introduction	✓		
Proposed System			✓
Project Schedule			✓
Milestones	✓		
Risk Assessment		✓	
Repository		✓	
UML Diagrams, Test Cases	✓	✓	✓