UNIVERSITY OF TECHNOLOGY,

MAURITIUS

SCHOOL OF INNOVATIVE TECHNOLOGIES AND ENGINEERING

DEPARTMENT OF INDUSTRIAL SYSTEM ENGINEERING

RAILWAY TRACKING AND ARRIVAL TIME PREDICTION

FINAL YEAR PROJECT

Final Year Project Supervisor: Mr Tse Kai Wai Dudley

Student Name: BISSESSUR Aïshani Prama

Student ID: 170731

Cohort: BCNS17BFT

Date of submission: 17.08.2020

Table of Contents

Table of Figures

Table of Tables

ACKNOWLEGMENT

First of all, I would like to thank my parents, whose love and guidance are always with me. They are the ultimate role models.

I would like to express my special thanks of gratitude to my lecturer for their knowledge and experience they share with us. Furthermore, I would like to thank my project supervisor Mr. Tse Kai Wai Dudley whose expertise was invaluable during my whole final year project and for his guidance.

ABSTRACT

The arrival time of trains has been some kind of problem in rail way station. The passengers usually don't know the exact arrival time of train and eventually reaches their destination late or miss the train. Therefore, this project has been developed to track and predict the actual arrival time of trains by using sensors and developing a mobile application. The actual time of the train is stored in a database and then retrieve by the mobile application to get the exact up to date time of the train. Furthermore, the trains arrival time is displayed in each station according the train path and time.

PROJECT CHAPTERS

CHAPTER 1

INTRODUCTION

People use train as a mean of transport in many countries as it is a convenient and safe. However, Rail transportation in some countries has significant long delays (Somkiat Kosolsombat, 25 June 2017). One of the major problems of these systems are the prediction of arriving train time to intermediary or final station. [1] Train timetabling is a complex and time-consuming problem. [2]

Tracking the arrival time and display the actual arrival time of trains can be used as solution to tackle the delay problem.

The project Railway Tracking and Arrival Time is about developing a system that will automatically record the arrival time of trains and will save the details without any admin intervention. It will also control the flow of trains in the station.

The system will be developed with the low-cost technologies available and with an innovated algorithm that will be implemented throughout the implementation of the system.

The system will be able to record the train's ID, station name and train's arrival time and save the details automatically. The data will be stored on the cloud as there will be more than one system running simultaneously in each station.

1.1 Problem Statement

The implementation of railway tracking and arrival time prediction will solve the problem related to time and traffic. According to research, the train delay causes British rail passengers to lose four million hours in just one year (independent.co.uk, 2018). Sometimes people don't know exactly at what time their train is meant to leave the platform. Moreover, if a person is late, or in a hurry, it's not easy to track the train. Furthermore, different trains travel at different speed.

1.2 Aim and Objectives

The aim of this project is to develop a railway tracking and arrival time prediction system. This will provide reliable and accurate time for trains in the different railway stations. Hence, the passengers will be able to verify the up to date time of their trains.

The objectives of the project are as follows:

1. Reliable and accurate (up-to-date) time of the train to arrive at a specific station.
2. Automatically take the details of the train and save it in the database.
3. Monitor the movement of trains in or out of the station.
4. The average time a passenger has to wait to board a train (if delay).
5. People can query time for a specific train.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter describes the study about the implementation of railway tracking and arrival time projects that can inform the passengers the accurate time for trains. During the research on it was noticed that in Thailand the State Railway, not all train had a delay problem. About 23.66% were delayed for less than 10 minutes, almost half of the total 186 trains, 47.85% had average delays of less than 20 minutes and about 175 trains or 94.09% had average delays of less than one hour. There were 11 trains or 5.91% having average delays of more than one hour, where 7 of these trains or 3.76% had average delays of more than 90 minutes (Suporn Pongnumkul).

An informative linkage mechanism for train reception and departure system by fusing intelligent video processing techniques was developed to get the accurate arrival time of trains. Surveillance cameras was used to deploy in track turnout region, thus real-time image for monitoring the status of train operation was obtained. These images were used for further analysis for acquiring the position, arrival time, type and velocity of the train. Cameras are installed in track turnout region for monitoring the operation of trains. The dispatch center is responsible for the monitoring of relevant area. As the region has a certain distance to the station platform, they can inform the passenger service system arrival and departure time when the train passes through the detection area. This could not recognize the train number. [3]

RT-Train: Real-time Public Train Tracking System is a system for the train user to check the train schedule, track the upcoming train's location in real time, and receiving announcement via push notification if train is late. All the data is added manually by the admin and driver. When the train is late driver will send a notification. [4]

A Real time railway indicator was developed using android studio and MySQL. This application displays the passenger information and display message information on mobile

application in real time. There is a server that hold the database and API server. This data, which is uploaded to database, will be available to call by the application via API until the next update and this process will keep repeating. The mobile application has a list of stations buttons, which when pressed show the information of the indicators. [5]

Real Time GPS for railway automation system was develop. Global Position system (GPS) was used to track every train and a proposed system by which every train are personally monitored and passing necessary messages to the individual trains. The train with the GPS Module and GSM module on board. This enable passenger to query the location of a train via SMS from his mobile phone but also provide real- time tracking system. This system was not secure to use as it can be attack by malicious agents who might try to disrupt the function of the system. [6]

2.1.2 Problem with actual system

According to the research we could observe that most of time some trains are late and the main problem is to predict the actual time of a train.

2.2 Proposed Solution

A mobile application will be developed to track the actual arrival time of trains and sensors also will be used. Each train will have a sensor and whenever it arrives to a station this will be read by a sensor reader to detect the train Id and its actual arrival time. These data will be stored in the database.

2.2.1 Microcontrollers

A microcontroller board has a microcontroller installed on it with all its pins already defined and ready for use.

The Arduino series microcontroller broads use the Atmel Atmega microcontroller series and are commonly known for automation and robotics projects. They are preferred choice for standalone projects as it is inexpensive and easy to work with the help of its IDE, Integrated Development Environment software freely available from its website.

Figure 1 Microcontroller Mega 2560

2.2.2 Integrated Development Environment (IDE)

The microcontroller board need to be programmed before it starts to work. Arduino provides a freely software with IDE to program the microcontroller board using a series of syntax code such as C++ which are known as sketch and those code are written on the IDE software working area. Then the IDE compile the sketch and load it onto the memory for microcontroller to execute. The IDE software has several examples and libraries that can help to develop the sketch.

Figure 2 Arduino IDE

The IDE has a Serial Monitor to monitor the output values as all outputs are displayed on this window. The Serial Monitor is used to check if the system is working properly by outputting the required values.

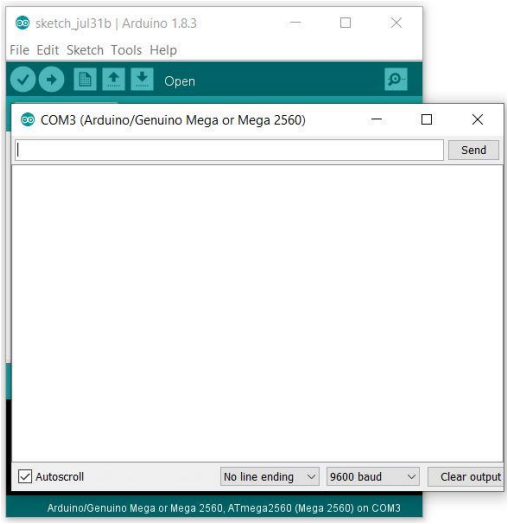


Figure 3 Serial Monitor

2.2.3 Breadboard

A breadboard is a rectangular plastic board with bunch of tiny holes in it. It is used to connect other electronic device and then multiple connection can be made from that point without being soldered. There is an arrangement of metal strips inside the breadboard in the horizontal and vertical direction. Electronic components can easily be connected to the holes. The breadboard is connected to the microcontroller pins 5v and GND.

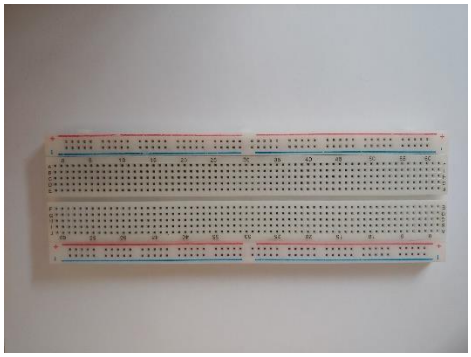


Figure 4 Breadboard

2.2.4 F-M Dupont Wires

It is an electrical wire with a pin or a connector at each end. They are used to connect components to the breadboard and the microcontroller. They can easily be plugged and unplugged.

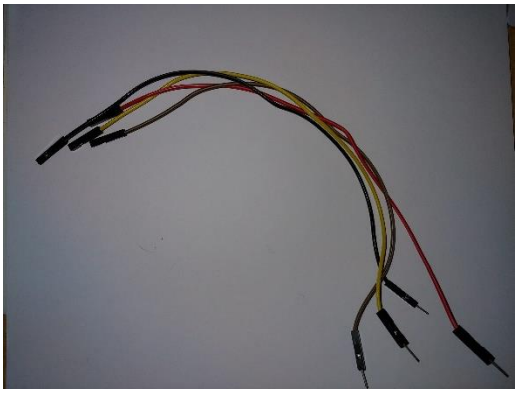


Figure 5 F-m Dupont Wires

2.2.5 Jumper Wire

It is an electrical wire with a pin at each end. They are used to connect the breadboard and the microcontroller.

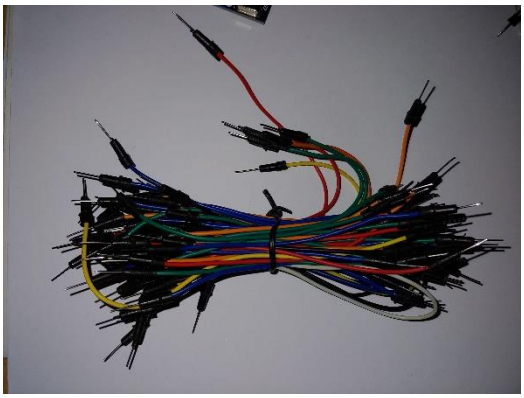


Figure 6 Jumper Wire

2.2.6 RTC Module

RTC stands for Real Time Clock. It is a time and date remembering system which keeps the module running and up to date even after the system has been shut down. It has a battery

setup inside which keep it running when the system is turned off. When the RTC module is connected to the microcontroller the accurate Time and Date is read.

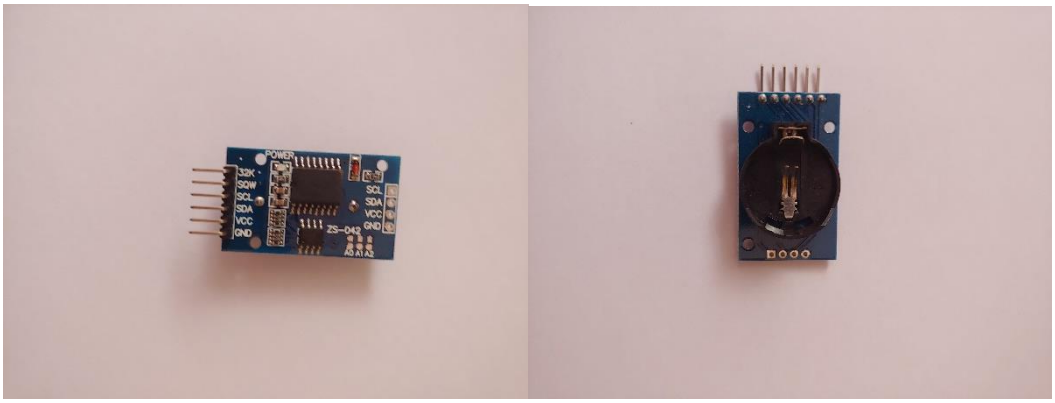


Figure 7 RTC Module

2.2.7 LCD Module

An LCD is an electronic displace module which used liquid crystal library to produce a visible image. The 16x2 LCD display will be used. The LCD will display the train number and the accrual arrival time of the train.

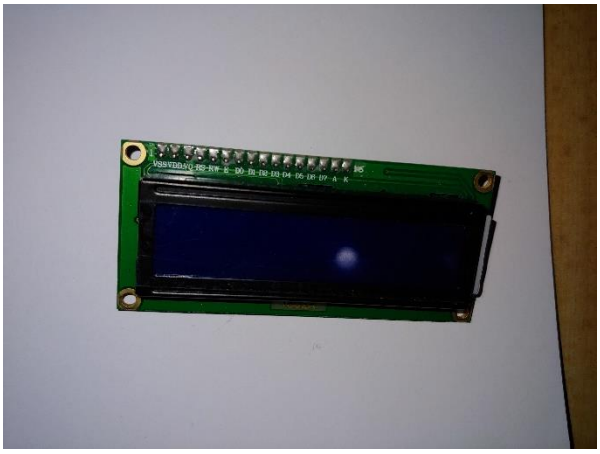


Figure 8 LCD Module

2.2.8 WIFI ESP 8266 Module

ESP 8266 is a Wi-Fi microchip which connect the microcontroller board to the internet. ESP 8266 libraries use TCP and UDP to communicate over Wi-Fi.

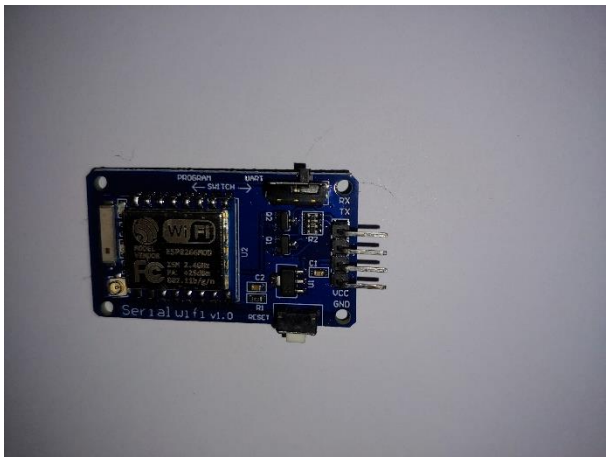


Figure 9 Wi-Fi ESP 8266 Module

2.3 Evaluation of Methods

2.3.1 Manual Input Method

This system consists of manually entering data with the help of an admin. Each time a train arrives at the station the admin will have to insert the arrival time in the database.

- Advantage of Manual Input Method
    i. Only a PC will be required to insert the data.
- Disadvantages of Manual Input Method
    I. The admin can make a mistake while entering data.
    II. The arrival time won't be accurate as a person is doing the work.
    III. One person should always be present in the office to add the time.

2.3.2 Motion Sensor (PIR) Method

The passive infrared motion sensor is used to detect any warm moving objects. It measures infrared light radiating from objects and generates energy based on observed heat surrounding it by default. Therefore, when any warm objects come nearby, it receives more thermal energy compared to normal conditions, thus determining the presence of an object or person. It will easier to detect any train.

- Advantages of Motion Sensor Method
    1. Easy to detect any movement.

2. It will help to detect the exact arrival time of time.
3. It is not expensive.
4. It is easy to install a motion sensor.

- Disadvantages of Motion Sensor Method
  1. It is insensitive to the very slow motion of the object.
  2. The PIR sensor can be triggered by any kind of moving object.

2.3.3 RFID Method

Radio Frequency Identification is an automatic id system which is a fast and accurate recognition. It uses an RFID tag or card which has a unique ID that can be read by an RFID reader. These RFID tags can be placed on each train for the identification of different trains.

The RFID tag has a red coil attached to it that sends a magnetic wave which induces a current in the coils of the RFID tag is enough to power the chip inside, to send those unique set number wirelessly through radio waves to the RFID reader to read.

Figure 10 RFID reader, tag and card

- Advantages of RFID Method
  I. Accurate recognition of train and arrival time.
  II. Easy to register train for identification.
  III. It is more secure no one can modify or tamper the information.
- Disadvantage of RFID Method
  i. RFID is more expensive than a motion sensor,

ii. RFID can cover a limited range of about 3 meters.

2.3.4 Chosen Recognition Method

After going through all these methods, it was found that the best method was to used RFID Radio Frequency method to identify the trains. RFID is fast and accurate. RFID is the best to identify multiple trains in a short period of time. An RFID card will be included in the train so that when it passes an RFID reader it can be identified.

2.4 Chosen Components

Here is the list of components for this project.

- An Arduino Mega 2560 board as a standalone main system to control all the modules and components.
- ESP 8266 module to connect the system to the network.
- An RFID reader module to read RDID cards of the trains.
- An LCD display to output the information of the system such as arrival time.
- RTC module to give the accurate time and date.

2.5 Selection of appropriate Software and Language

Android studio will be used to implement the project. The project consists of four part that should be developed an android application for the admin, client, driver and station. The android application will be developed on Android studio using Java as programming language. Furthermore, for the sensor part Arduino IDE will be used which used C++ and C as programming language.

2.5.1 Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains'IntelliJ IDEA software and designed specifically for Android development. The latter require a minimum 4 GB of RAM and minimum 2 GB of available disk space for Android SDK and emulator system image. An android emulator simulates an android device on your computer so that you can run the application created on it instead of testing it on a real android phone. Furthermore, the

application can be tested on a variety of devices. Android studio use Java and Kotlin as programming language. In this project Java has been used as programming language.

2.5.2 Java

Java is a widely used programming language and is designed for the distributed environment of internet. It is a general-purpose programming language that is concurrent, class-based, and object-oriented. Java follows the principles of Object-Oriented Programming (OOP). The OOP consist of Objects, Classes, Inheritance, Encapsulation, Abstraction and polymorphism.

Advantages	Disadvantages
Java is simple and straightforward to use.	Java is memory-consuming and significantly slower and poor performance.
Use object-oriented language.	
It is multithreaded.	
It is platform independent.	

Table 1 Advantages and disadvantages of Java

2.5.3 Firebase

Firebase is a mobile and web application development platform developed by Firebase, in 2011, then acquired by Google in 2014. It is a unified backend-as-a-service (BaaS) platform for mobile developers. It has many services such as

- Firebase Cloud Messaging
- Firebase Authentication
- Firebase Realtime Database
- Cloud Firestore
- Cloud Functions
- Cloud Storage
- Firebase Hosting

- ML Kit

For this project Realtime Database and Firebase Authentication have been used to store the data. Both the mobile application and the Arduino equipment's are connected to the firebase.

Here are some details about Realtime Database and Authentication

Realtime Database

Realtime Database is Firebase's original database. An API is provided to the application developers that allows the application data to be synchronized across clients and stored on Firebase's cloud. It's an efficient, low-latency solution for mobile apps. It stores all the data for the mobile and can easily access and retrieve data from another mobile using the same database. The database is accessible through a REST API.

The REST API uses the Server-Sent Events protocol, which is an API for creating HTTP connections for receiving push notification from a server. The data in Realtime database can be secured by using the server-side-enforced security rules.

Advantage of using online database

- Less storage will be used on local machine.
- It can be used by multiple user at the same time
- It can be accessed from a web browser or mobile application from anywhere in the world.
- No installation of any software is required.

Disadvantages of using online database

- If there is no internet connection, user won't get access to the database.

Authentication

It manages the users in a simple and secure way. It can authenticate users using only client-side code. Moreover, it includes a user management system where it enables user

authentication with email and password login stored with Firebase after registration through the mobile application.

Connection with Android Studio

Furthermore, Android studio has been connected to the firebase. Android studio has a tool to connect directly with the firebase. The following steps have been done to achieve this process.

1. Click on Tools and then select Firebase

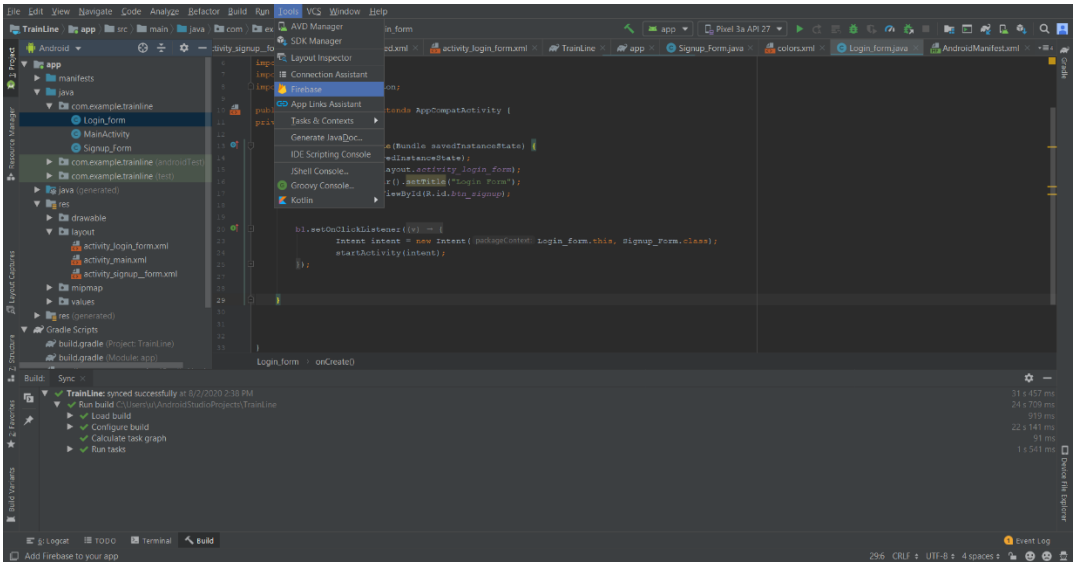


Figure 11 Firebase Tool

2. The options for firebase is appeared, click on the Realtime Database.

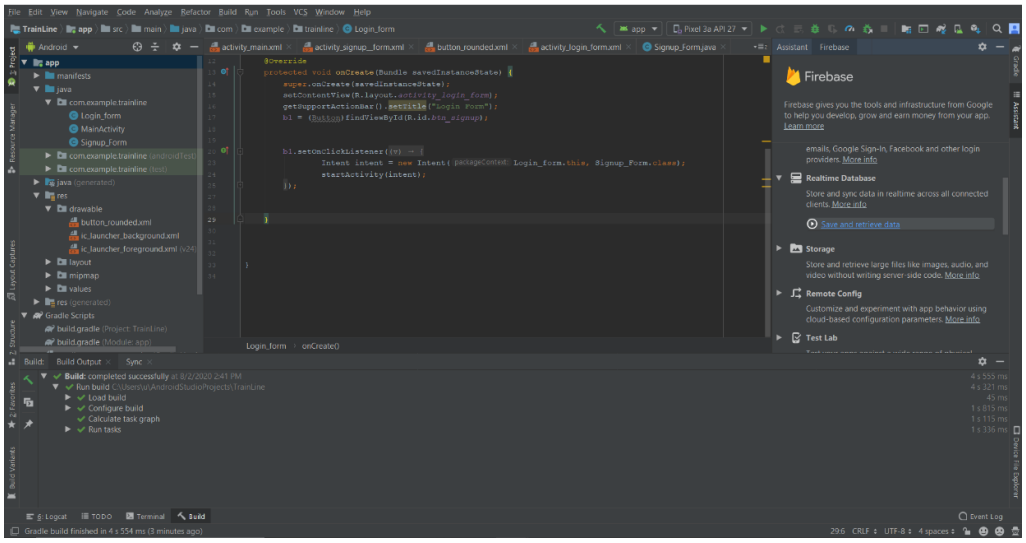


Figure 12 Firebase Realtime

3. After the step 2 click on Connect to firebase.

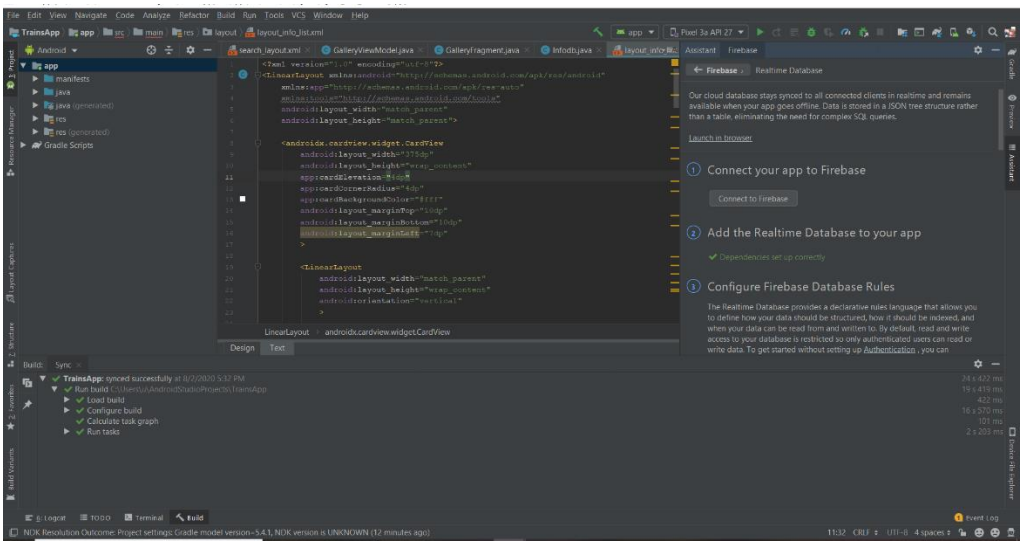


Figure 13 Connect to Firebase

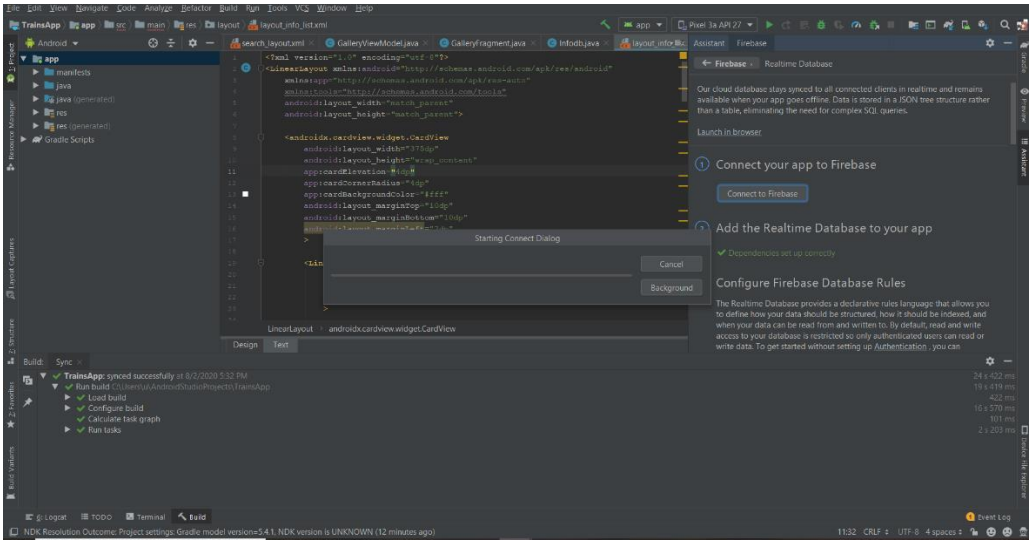


Figure 14 Starting Firebase Connection

4. Then click on Add the Realtime Database to your app and Accept Changes.

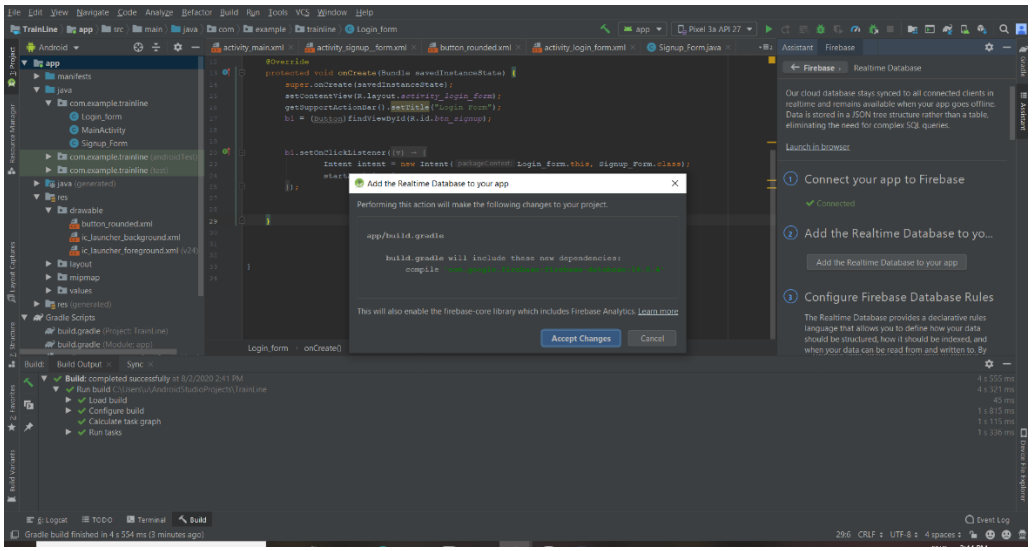


Figure 15 Accept Changes for Realtime Database

5. After that click on Authentication.

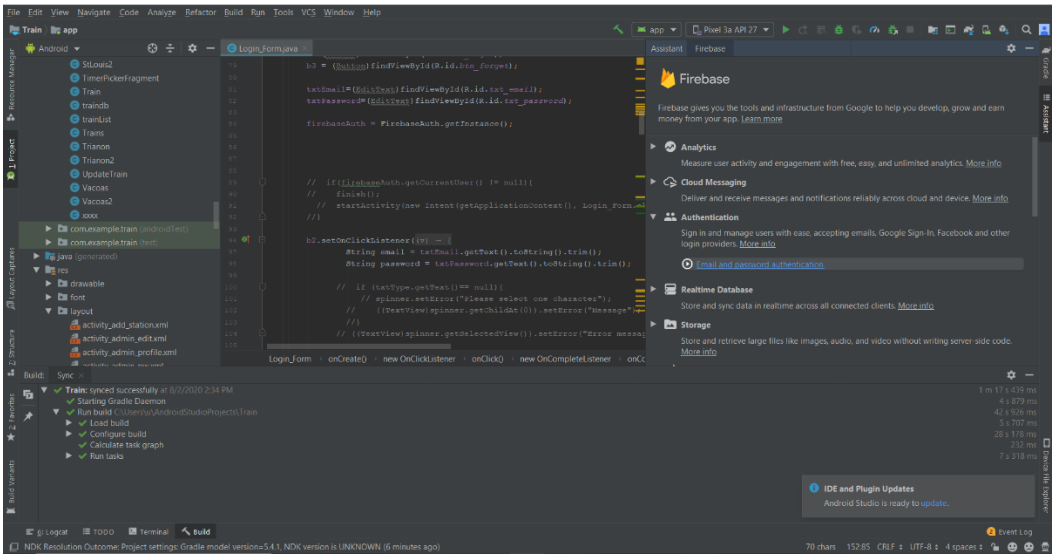


Figure 16 Add Authentication

2.6 Components Testing

The components are tested individually to see how it works according to their datasheets and code obtained from examples. These initials tests are done to see how it works and also to check if they are working properly with the microcontroller board using bread board, jumper wires and other components.

2.6.1 Test of the Arduino IDE and microcontroller MEGA 2560 board

The Arduino IDE includes a large collection of libraries and sketches, as examples, that can be loaded to the memory on the board for the microcontroller MEGA 2560 to execute it. The microcontroller MEGA 2560 board is connected to a laptop by a USB cable. One end of the USB cable is plugged into the board and the other end is connected to the laptop, this is how a sketch can be loaded.

After the connection the blink sketch is loaded to see if it executes accordingly, for example blink LED sketch.

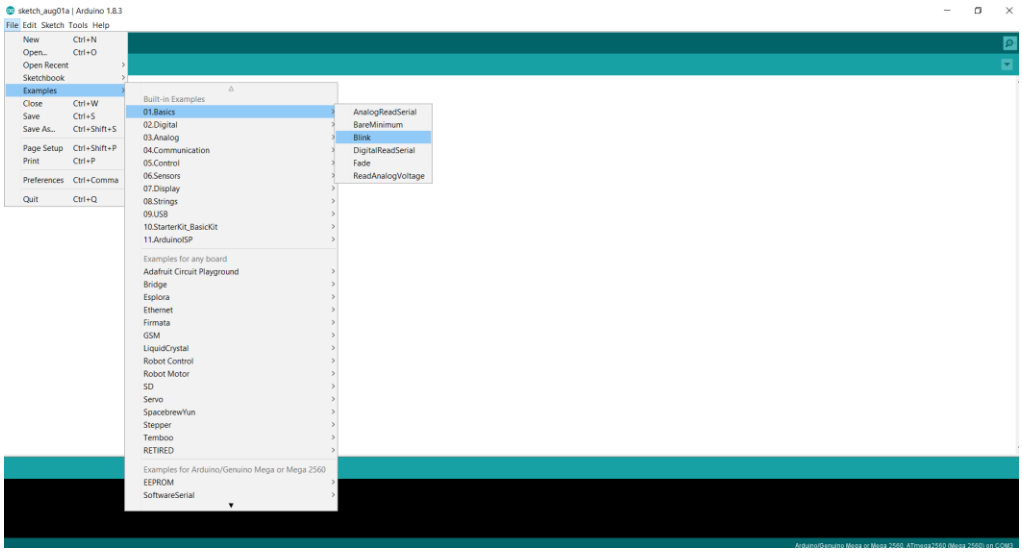


Figure 17 Uploading libraries

The sketch is compiled and uploaded on the board to execute, soon after the sketch has been uploaded, the system starts running and a green LED on the board stars blinking.

Figure 18 Testing Mega 2560 board

2.6.2 Test of the LCD

The LCD has been connected with the MEGA 2560 board on pin 5V, GND, SDA (20) and SCL (21). The LCD works the library Liquid Crystal l2C for sketch. The library has been

added to the software by adding the LiquidCrystall2C.zip file. After the sketch has been added the example to print “Hello World” has been uploaded.

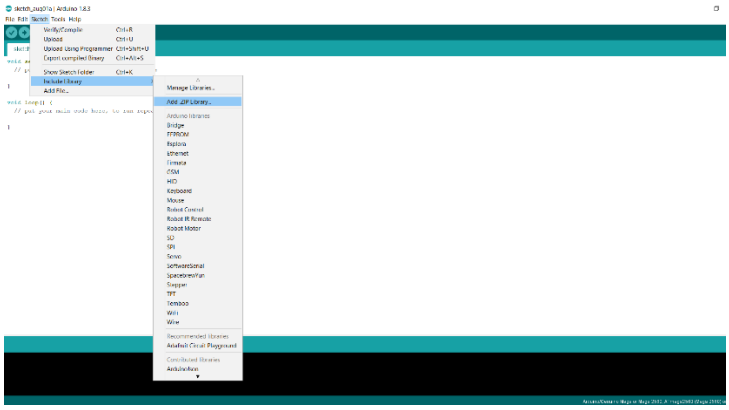


Figure 19 Uploading library

Figure 20 Display "Hello World"

2.6.3 Test of the RTC Module

For the RTC Module the MEGA 2560 board on pin 5V, pin GND, SDA and SCL. The library and sketch Real Time Clock is uploaded to get the accurate time.

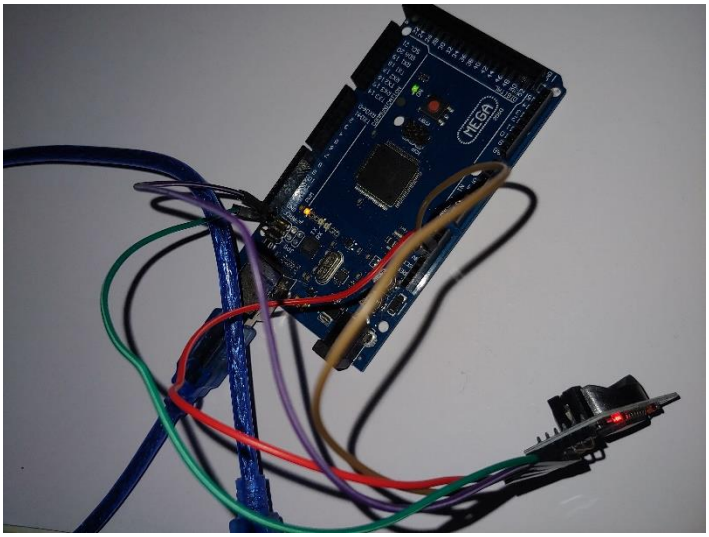


Figure 21 RTC Module connected to Mega 2560 board

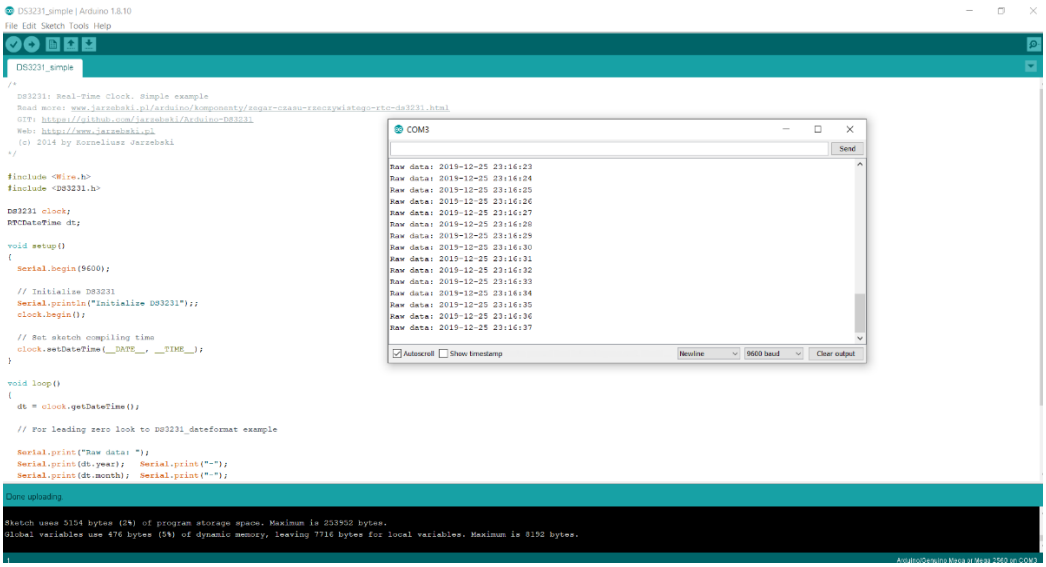


Figure 22 Display current date and time

2.6.4 Test of the LCD connected with the RTC Module

Both the LCD and the RTC Module is connected to the breadboard as both used the 5V and GDN pins. In this both the LiquidCrystall2C and Real Time Clock libraries has been used and uploaded to get the result.

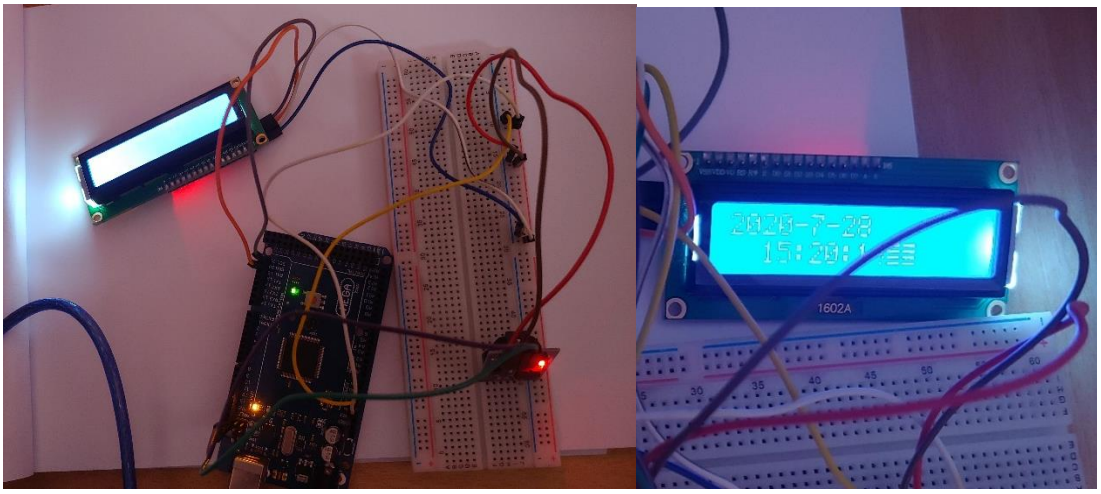


Figure 23 Display Date and Time on LCD

2.6.5 Test of the RFID

The RC522 RFID Reader Module library is uploaded on the MEGA 2560. The RC522 RFID is connected on the following pins 3V, GND, pin 49, pin 50, pin 51, pin 52 and pin 53 on the board.

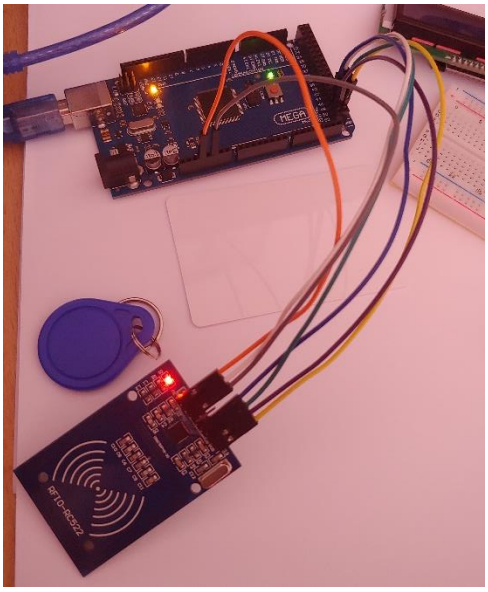


Figure 24 RFID Reader connected to Mega 2560 board

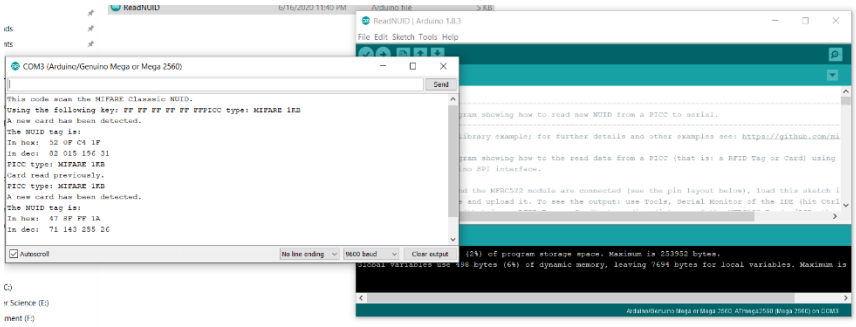


Figure 25 Uid of the tag and card

2.7 Scope

A railway tracking and time arrival prediction system will be developed. This system will have RFID sensors implemented at a different station which will detect the actual arrival time of the train. The train ID and arrival time will be recorded in the database once the train passes through the sensor. The database will keep track of train ID, train number, arrival time, delay time and station's name. Consequently, each station platform will retrieve and display this information from the database. The database will be on a cloud where each station will be able to access it. Moreover, if the train time delay is 2 minutes, the RFID sensor will read the actual time and then add + 2 min next to the arrival tie, thus informing the passengers to the next station about the delay. It will also indicate if a train is early by displaying -2 min next to the arrival time. Furthermore, passengers will be able to query for the train, as a mobile application will be created to help them. The passenger will be able to view the different train time table for a specific day and date.

2.8 Project Management

Project management is the process of working to achieve goals and meet all the criteria on a specific time.

2.8.2 Gantt Chart

A Gantt is really a simple bar chart with tasks detailed on the Y axis and time represented on the X axis as shown below.

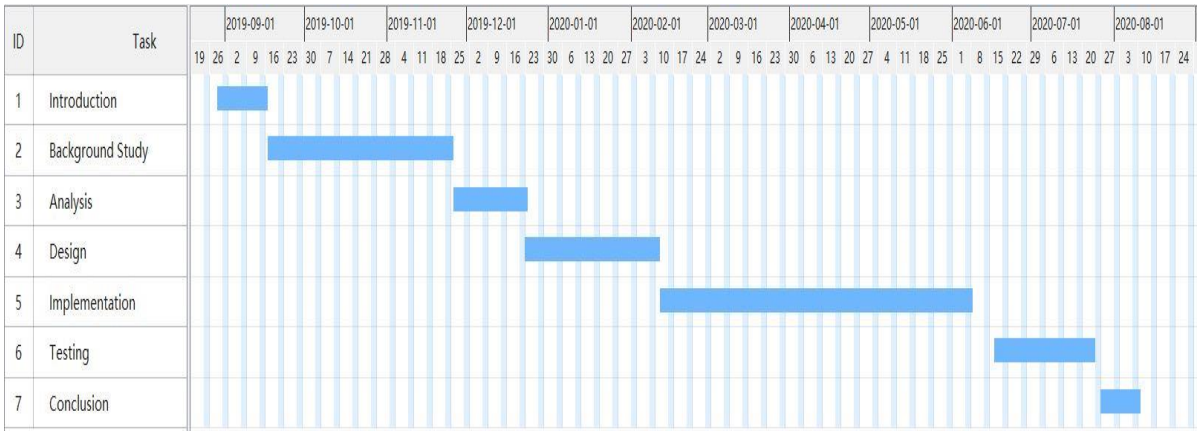


Figure 26 Gantt Chart

CHAPTER 3

FEASIBILITY STUDY

3.1 Technical Feasibility

Technical Feasibility is an essential step in the design process that ensures the proposal solution is feasible across the range of device and platform.

In this project, the aim is to developed a mobile application for admin, user and driver. Furthermore, to create an online database that will allow synchronization of data between the application.

Mobile application

Mobile application may be created with a variety of development technologies including hybrid tooling such as PhoneGap, Xamarin and native tooling such as Android studio.

Native Mobile Application

- They are downloaded and installed by customers.
- Android, iOS, Window Mobile are required as supporting platform for execution.
- Excellent performance

Hybrid Mobile Application

- They are developed by using web technologies but runs as native on different platforms.
- HTML5, CSS3, JavaScript and jQuery are used to develop the app.
- Medium performance.

Moreover, a native mobile application will be created in this project after analysing the different type. It is easy to download an install by user and has an excellent performance.

3.2 Operational Feasibility

Operational Feasibility focuses on the degree to which the proposed system fits in the organisation and whether the user will use the system.

Does the proposed system solve the problems mention above and how do user benefit from it?

- It allows user to track the nearest station from their actual location.
- It gives the up to date arrival time of trains on each station.
- The user interface of the application is kept simple for common user to understand.
- The user will know whenever the train will be late.

3.3 Economic Feasibility

Economic Feasibility analysed the cost and benefits of the proposed system.

Project Cost

Quantity	Description	Total Cost
1	Android device	Rs40,000
1	Microcontroller Mega 2560	Rs4000
1	RFID Reader	
2	Sensor	
1	breadboard	
1	LCD Screen	
1	RTC Module	
Total		Rs44,000

Table 2 Project Cost

CHAPTER 4

METHODOLOGY

For this project, the Waterfall Development Model was chosen as methodology. It is the less iterative and flexible approaches as the progress flow only in one direction. Below is the diagram of Waterfall Development Model.

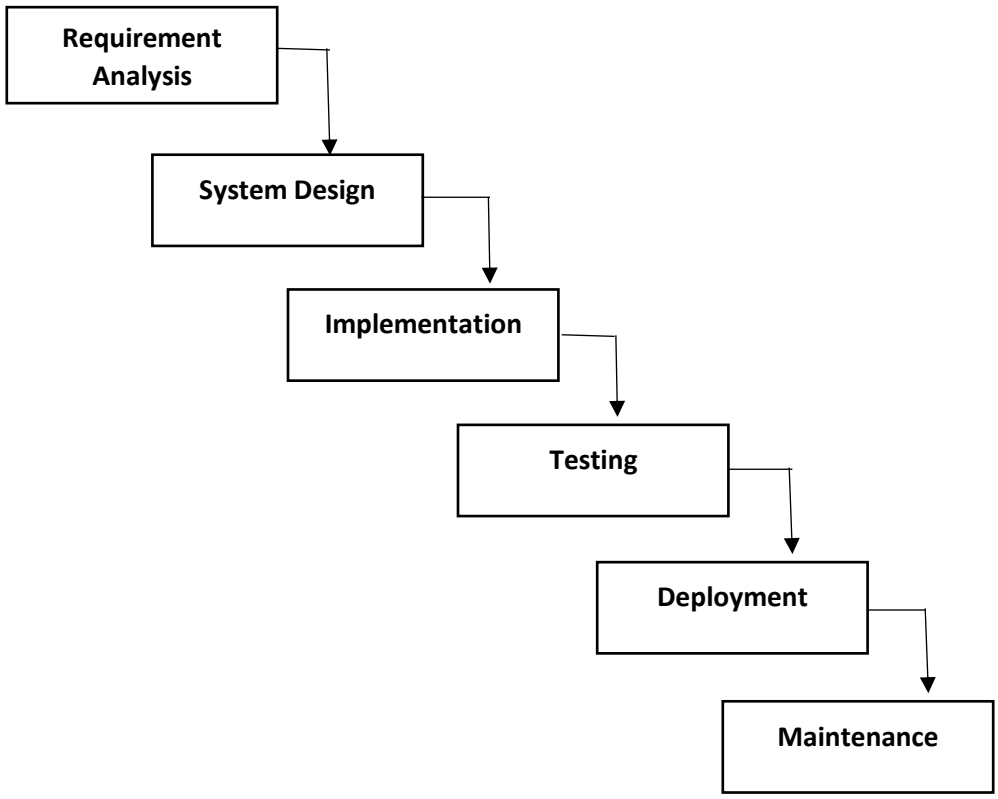


Figure 27 Waterfall Development Model

The waterfall model sequential phases are:

- **Requirement Analysis** – The system requirements that should be developed are gathered in this phase and documented.
- **System Design** – The system design is done after analysing the requirements mention in phase one. In this phase the hardware and the system requirements are specified. The architecture of the system is prepared in this phase.

- **Implementation** – In this phase the system is developed into small programs called units and unit testing is done as each unit is developed and tested its functionality.
- **Integration and Testing** – All the units developed in phase three is integrated into a system. After integration the entire system is tested to see there is any failures.
- **Maintenance** – In this phase the issues which come from the client environment is fixed and a better version is released.

Advantages of waterfall model

- It is easy and simple to understand.
- Easy to manage for a small project.
- Easy to arrange tasks.
- Each phase is processed and completed one at a time.

Disadvantage of waterfall model

- Difficult to go back and change something once arrived on the testing stage.
- Not suitable for long and ongoing projects.

CHAPTER 5

REQUIREMENTS ANALYSIS

Requirement analysis is critical to the success of a system or software project. The project requirements have two part functional and non-function (technical) requirements. The functional and non-functional list will identify the requirements of the software.

5.1 Functional Requirements

A functional requirement is the requirement that will allow the user to perform some kind of function. In this the operational features of the projects are specified and are used to design the system's interface.

Admin

The whole system is controlled by an admin who login in the system by giving his authentication details such as username and password. After login into the system, he will be able to view modify and add the trains' details which are available to the passengers. The admin will also add time table of the upcoming trains.

Sensor

The sensor will read the actual time of the arrival train. If any delay is reveals in the arrival time at any station the sensor will modify the arrival time automatically in the database.

User

An android application will be created where the user will be able to search the available trains by their requirements such as departure time, destination and journey date. The user will be provided with a list of available trains with accurate arrival time. The user will also know if the train is delayed and indicating by how many minutes late.

Driver

The driver also will have an application in which he will be able to inform the admin about any delay and by how much time and at which station he will be late.

Here is the list of function requirement:

- The system will contain a graphical user interface.
- The user shall be able to input the station name, train number, date and the arrival time of trains.
- The system shall display the train information on stations.
- The system shall display the up to date of the train, whether its “on time” or “late”.
- The user shall be able to view the timetable of the trains.

5.2.1 Security

Security is an important factor to consider while developing the application.

- Authentication

  Since the application allows different users to register and login for example the admin and the client. Each user has their own interfaces and cannot access the interface of others.

  Session and account type will be used. When a user will register he/she will have to select user type and he/she will be directed to the user interface.
- Verification of email address

  Each time a new user is registered, the system will verify if that email already exist. If it’s already exist, the user won’t be able to register again. If the user does not exist a verification email will be send to the user to confirm the registered email. After confirming the email, the user will get accessed to login with his/her email else the latter won’t be able to login.
- Verify User changes

  The system will send a user email whenever he/she changes his person detail especially his/her email address. A mail will be send informing about the changes made and stating the new email address to verify if the user has made these changes.

5.2 Non- Functional (Technical) Requirements

Non-Functional requirements define systems properties and constraints. The following Non-Functional requirements would be considered.

- The system shall be able to allow the user to interact with the system.
- The interface shall be user friendly.
- The system shall have a main menu where the user can navigate.
- The admin shall should be able to create new train and schedule.
- The admin shall be able to edit and delete the information which has been created.
- The system shall allow the user to view the save schedule and details.
- The system will have a map where user can just send their queries to the admin.
- The shall have a registration and login page.
- The user shall be able to edit their personal information.
- The user shall be able to delete their account.
- The client shall be able to calculate the fees of their journey.
- The system shall be reliable and ensure that the data is secure.

5.3 Propose Solution

An android application will be developed to overcome the problems faces by passengers a mention in the literature review. The android application will be divided into four parts, the admin, the client, the driver and the station part. The admin will be allowed to login in the application and made changes about any time schedule. The latter will be able to create schedule for different trains for different date and time to prevent traffic and predict the arrival time of trains. The client will be able to view the timetable of trains and search for a specific train by filtering which station, platform and train number they want. He will also be able to send query to the admin about any station and train. The admin will be able to reply back to the client query. In the driver part, the driver will be able to send notification to the admin if ever the train will be late by how much time and on which station and reason for being late. After getting the notification the admin will be able to change the time for this particular train. This modification will enable the client to know

that their train will be late. For the station part, the detail of trains and time will be display in each station according to the train destination station. The station will display the information for trains for the current date and starting from the actual time.

Furthermore, to be able predict the actual time of each train, sensors and RFID reader will be used. The sensors will be place at each station and each train will have a different unique ID number for identification. Whenever the train will pass through the sensor, the RFID reader will read and detect the train ID number and the actual time. Then these data will be send to the database. If the actual arrival time match with the schedule time it will display "on time", if its early it will display "- min" and if its late it will display "+ min" so that the passenger waiting on the next station will know the actual time of the train and if its late.

5.4 Hardware and Software Requirements

The development includes the hardware and software on which the system will run. These have already been described in the literature review part.

Here is the list of hardware that will be used.

- Mobile Phone and tablet
- Microcontrollers Mega 2560
- Bread board
- RTC Module
- ESP 8266 WIFI
- LCD Screen
- RFID- RC522
- Sensors
- Wires and USB wire

Here is the list of software that will be used.

- Android Studio
- Arduino IDE

- StarUML – for UML design

5.5 Use Case Diagram

A use case is a set of events that occurs when an actor uses a system to complete a process. An actor represents a role played by an outside object. One object may plan several roles and, therefore is represented by several actors.

In this project there will be three actors the admin, the client and the driver. The admin will be interacting with the whole software. The client will be able to view and send queries about the schedule for trains. Furthermore, the driver will be able to send notification.

Admin Use Case Diagram

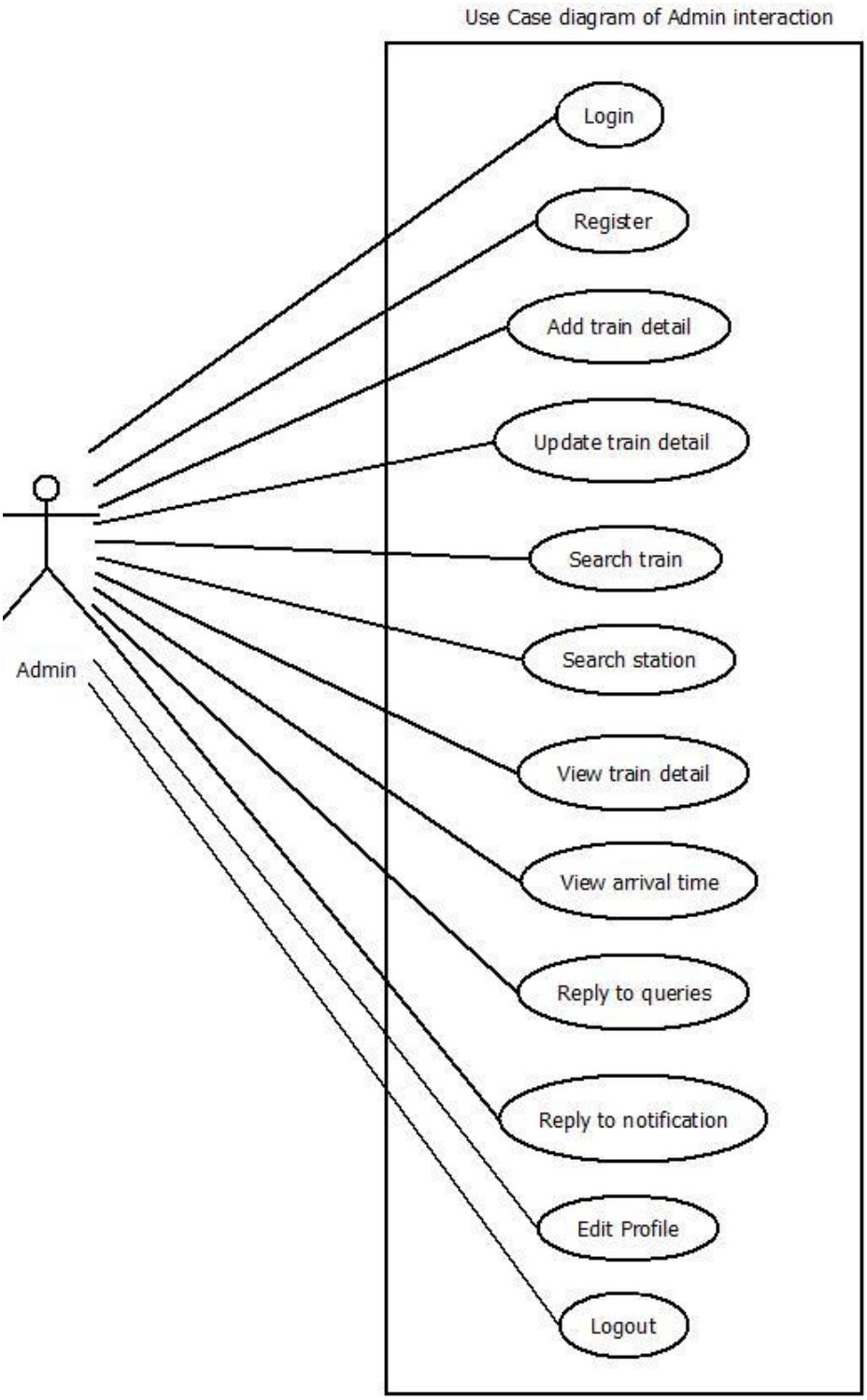


Figure 28 Use Case Diagram of Admin interaction

Client Use Case Diagram

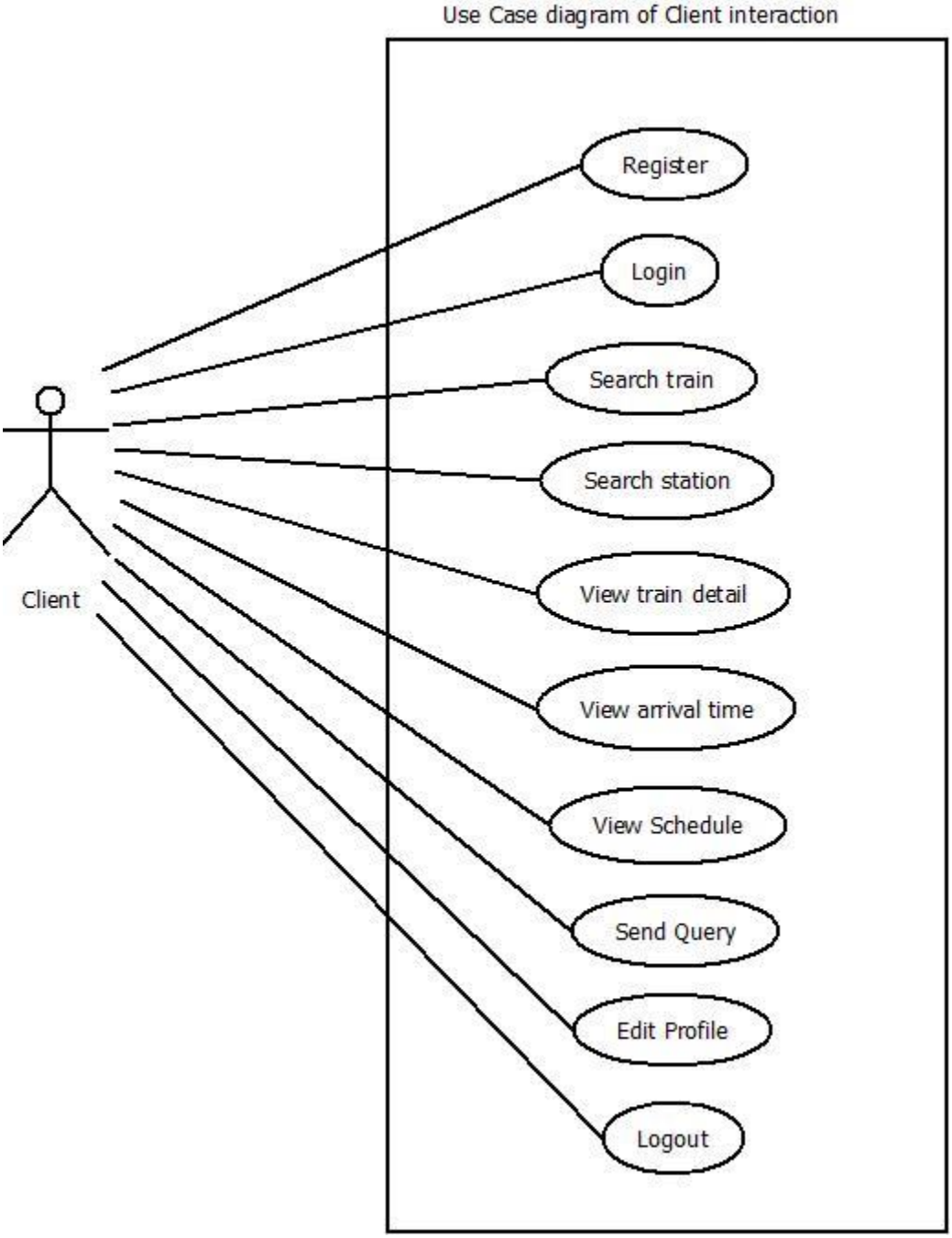


Figure 29 Use Case of Client interaction

Driver Use Case Diagram

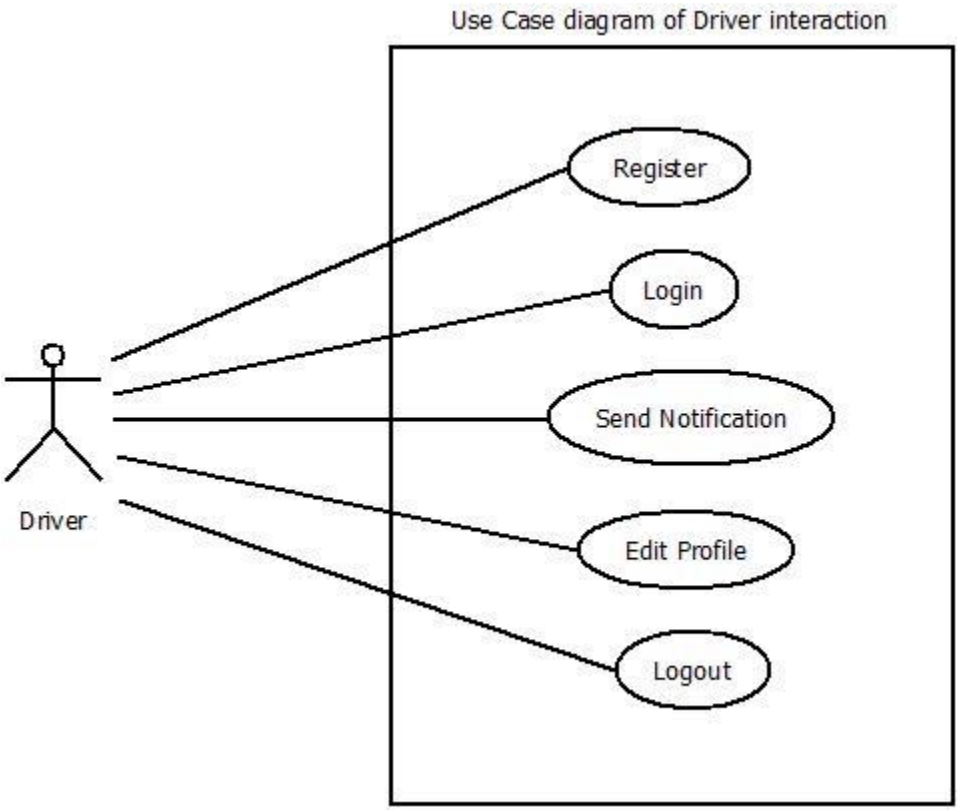


Figure 30 Use Case of Driver interaction

5.6 Written Use Case

Written use case is a written description by step-by-step of a user will perform tasks on the application. It helps to explain how the system should behave during the process and give a brief view of what can go wrong during the process. It provides a list of goals that should be achieved. It consists of the following elements:

- **Title** – descriptive name in the use case diagram.
- **Level** – describe the degree of detail in the use case.
- **Primary actor** – A user who interact with the system to achieve a goal.
- **Stakeholder –** someone with an interest in the behavior of the use case.
- **Precondition –** the conditions that must be satisfied in order to execute the use case.

- **Minimal guarantee –** the output that can be expected when the system request failed.
- **Success guarantee –** the outputs that are expected when the system request succeeds.
- **Trigger** – An action that initiates the use case.
- **Main success scenario** – describe the interaction between actor and the use case during the execution process.
- **Extensions** – describe about error handling if something go wrong.

5.6.1 Written use case for Admin

Use Case Title	**Use Case of admin interaction with the system**
Primary Actor	Admin
Level	Kite (Summary)
Stakeholder	Admin
Precondition	Admin accesses the system application
Minimal Guarantee	Roll back of any uncompleted transaction
Success Guarantee	Admin gets the train detail
Trigger	Admin accesses application page
Main Success Scenario	1. Admin fill registration form. 2. Admin login in application. 3. Admin add train details. 4. Admin update train details. 5. Admin searches for a specific train. 6. Admin searches for station. 7. Admin view train detail. 8. Admin view arrival time. 9. Admin replies to client queries. 10. Admin replies to driver notifications. 11. Admin logout application.

Extension	1a. Admin could not register. 1a1. Admin quits application. 1a2. Admin takes action to regain access. 2a. There is a problem with the application. 2a1. Admin refresh page. 2a2. Admin quits site. 3a. Admin cannot add trains detail. 3a1. Retry to add the details. 3a2. Refresh the page. 4a. Admin cannot search for a train. 4a1. Retry to search. 4b2. Reload the page. 4b3. Quit the page.

5.6.2 Written use case for client

Use Case Title	**Use Case of client interaction with the system**
Primary Actor	Client
Level	Kite (Summary)
Stakeholder	Client
Precondition	Client accesses the system application
Minimal Guarantee	Roll back of any uncompleted transaction
Success Guarantee	Client gets the train detail
Trigger	Client accesses application page
Main Success Scenario	1. Client fill registration form. 2. Client login in application. 3. Client searches for a specific train. 4. Client searches for station. 5. Client view train detail.

	6. Client view arrival time. 7. Client sends queries to admin. 8. Client logout application.
Extension	1a. Client could not register. 1a1 Client quits application. 1a2. Client takes action to regain access. 2a. There is a problem with the application. 2a1. Client refresh page. 2a2. Client quits site. 3a. Client cannot search for a train. 3a1. Retry to search. 3b2. Reload the page. 3b3. Quit the page.

5.6.3 Written use case for driver

Use Case Title	**Use Case of driver interaction with the system**
Primary Actor	Driver
Level	Kite (Summary)
Stakeholder	Driver
Precondition	Driver accesses the system application
Minimal Guarantee	Roll back of any uncompleted transaction
Success Guarantee	Driver gets the train detail
Trigger	Driver accesses application page
Main Success Scenario	1. Driver fill registration form. 2. Driver login in application. 3. Driver sends notification about latency to admin. 4. Driver logout application.

Extension	1a. Driver could not register. 1a1. Driver quits application. 1a2. Driver t takes action to regain access. 2a. There is a problem with the application. 2a1. Driver refresh page. 2a2. Driver quits site. 3a. Driver could not send notification to admin. 3b1. Reload the page. 3b2. Quit the page.

5.7 Sequence Diagram

It shows the object interactions and the exchange message between objects needed to execute the functionality of the scenario.

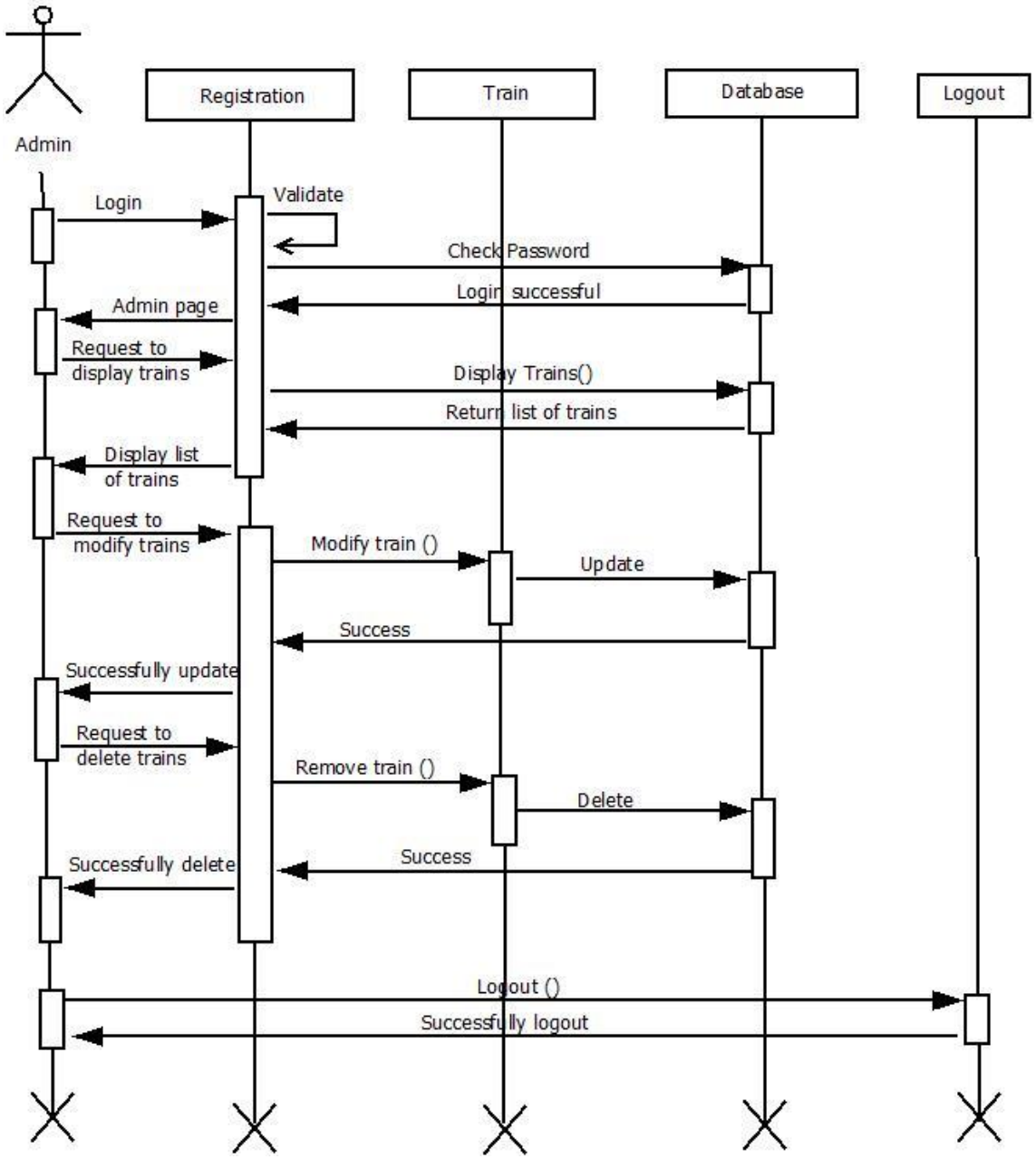


Figure 31 Sequence Diagram

5.8 Class Diagram

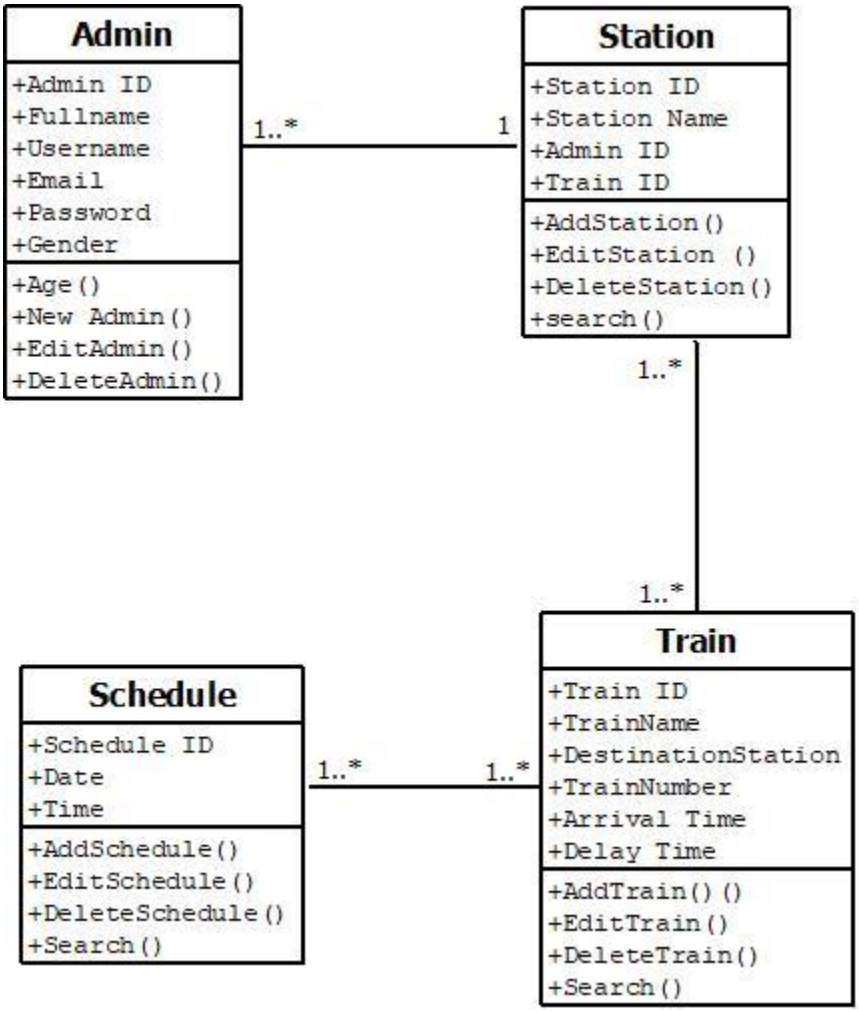


Figure 32 Class Diagram

CHAPTER 6

DESIGN

6.1 User Interface layout

1) Pre-launching

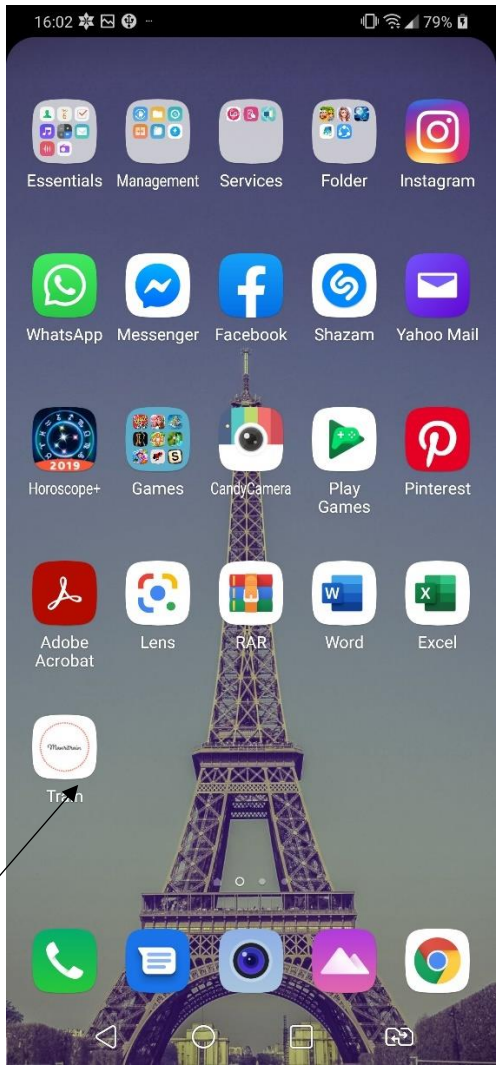


Figure 33 Application icon

The application icon

2) Admin Menu Screen

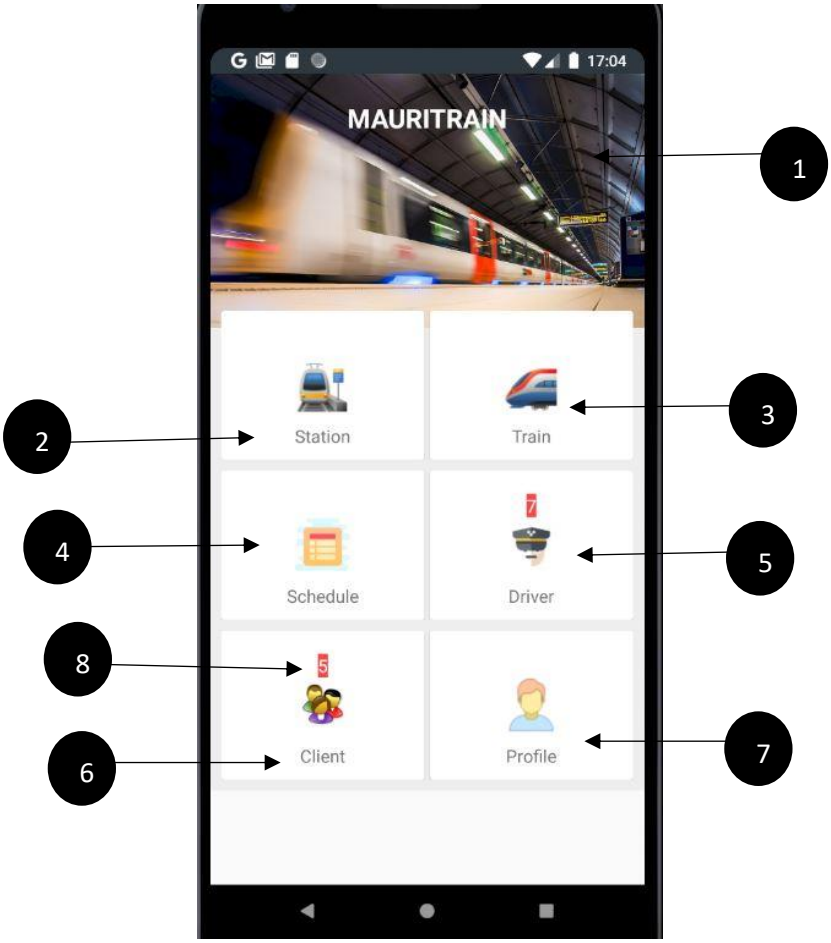


Figure 34 Layout of driver main menu

1. Background banner with the title.
2. Station card view menu icon links to Station screen.
3. Train card view menu icon links to Train screen.
4. Schedule card view menu icon links to Schedule screen.
5. Driver card view menu icon links to driver notification screen.
6. Client card view menu icon links to client queries screen.
7. Profile card view menu icon links to View Profile screen.
8. Show the number of queries and notifications.

2) Reply to Queries and Notification

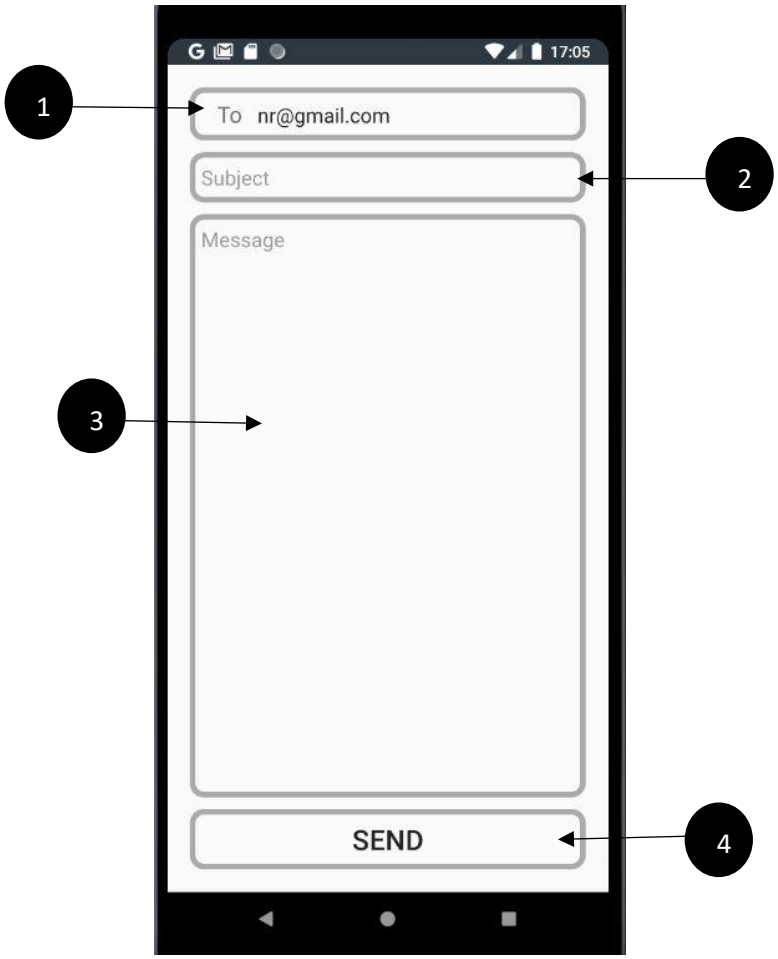


Figure 35 Layout to reply query and notification

1. Edit Text to get the client/driver email.
2. Edit Text to write the subject concerning to the query ask.
3. Edit Text to write the answer for the queries asked.
4. Button to send an email to the client/driver.

3)Login Interface

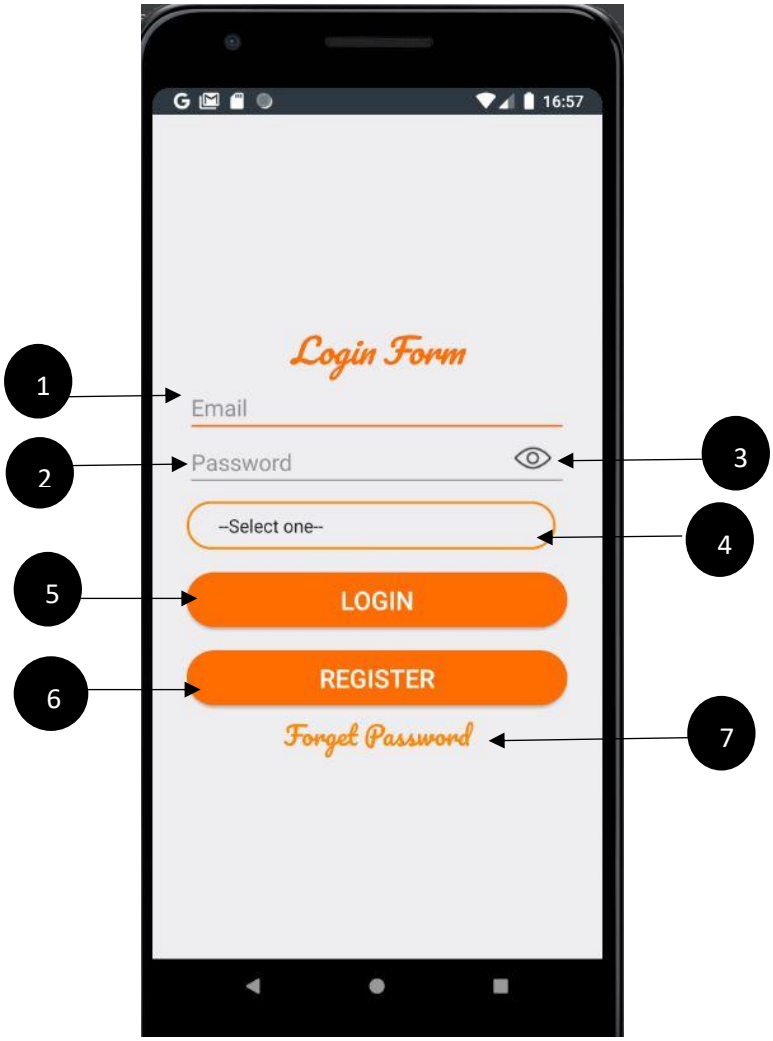


Figure 36 Layout of login form

1. Edit Text to insert existing email address.
2. Edit Text to insert password.
3. Button to view password and hide it.
4. Spinner to select if admin, client or driver.
5. Button to authenticate user to main menu after login.
6. Button to register for new user.
7. Button to reset password if forgotten.

4) Registration Interface

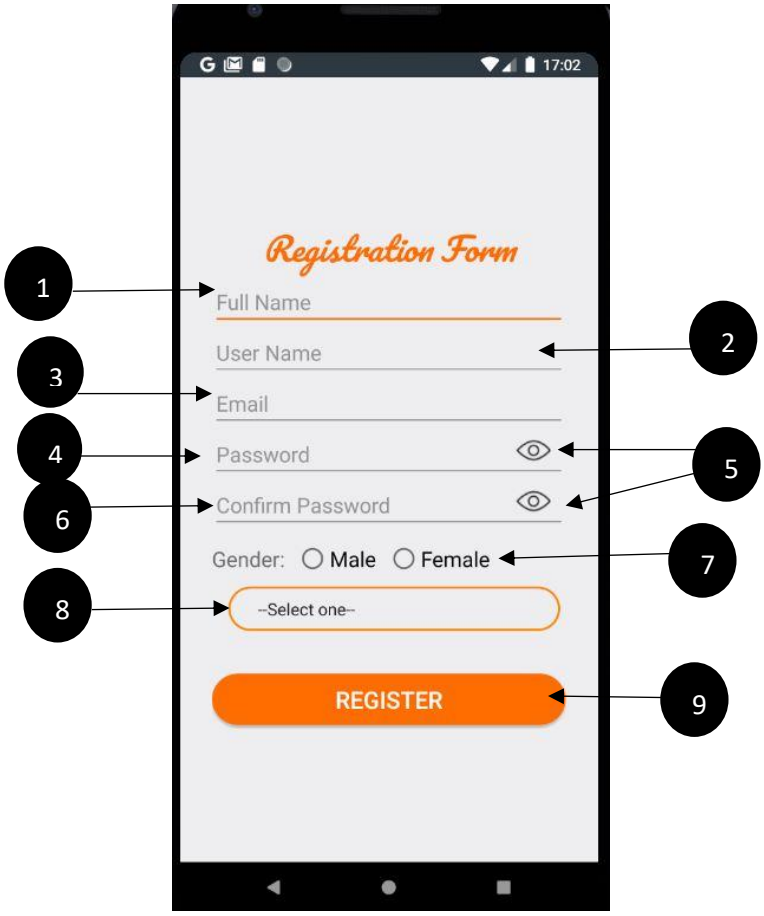


Figure 37 Layout of registration form

1. Edit Text to insert Full name.
2. Edit Text to insert User name.
3. Edit Text to insert existing email address.
4. Edit Text to insert password.
5. Button to view password and hide it.
6. Edit Text to insert same password to confirm password.
7. Radio Button to select a gender.
8. Spinner to select admin, client or driver.
9. Button to save the information.

5) Forget Password Interface

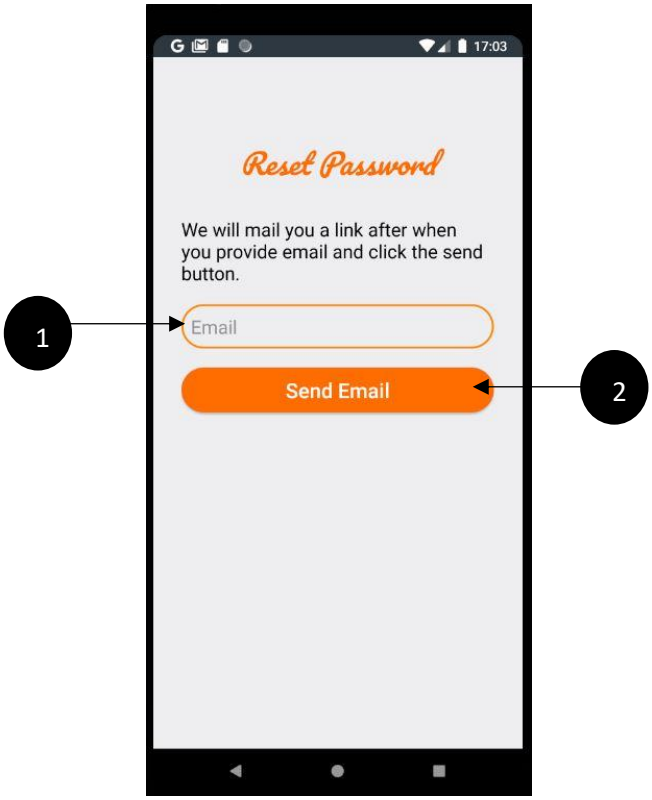


Figure 38 Reset password layout

1. Edit Text to enter user email.
2. Button to send reset password email to user.

6) User Menu Interface

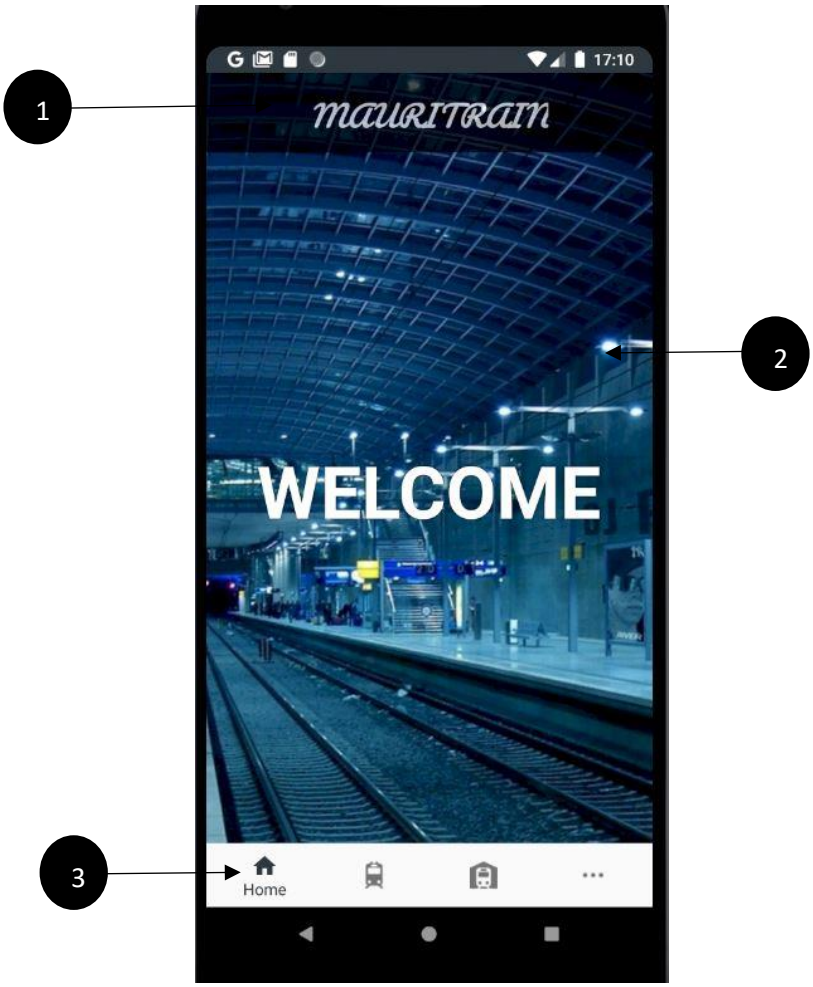


Figure 39 User Main Menu layout

1. Tittle layer
2. Background image slide show
3. Bottom navigation menu

7) Client Train display layout

Figure 40 Layout for train display

1. Edit Text enter test for search
2. Button to search the text in the edit Text.
3. List View to display the train list.

8) Client Station display layout

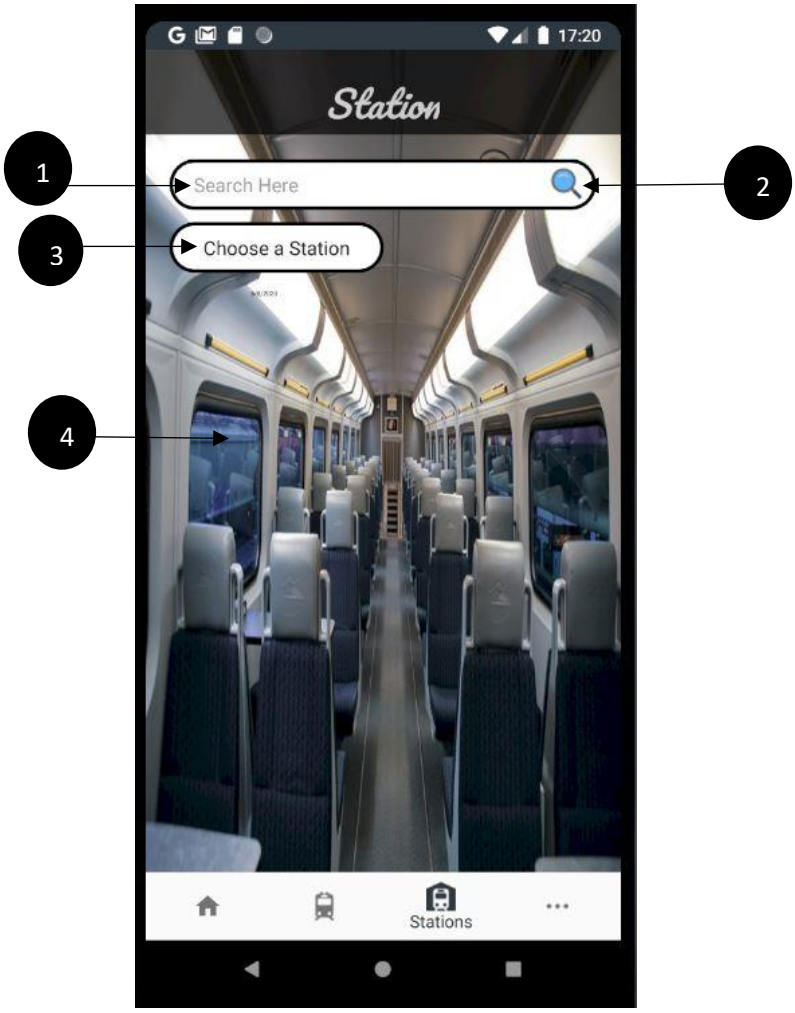


Figure 41 Layout for station list

1. Edit Text to enter text for search.
2. Button to search for the test in edit Text.
3. Spinner to search for a specific station.
4. List view to display list of stations.

9) Profile of User

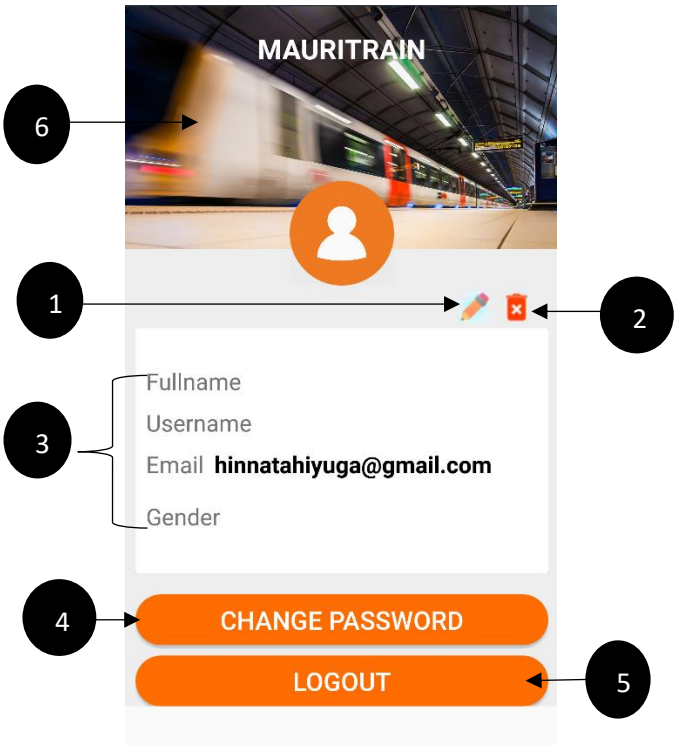


Figure 42 Layout of user profile

1. Button to edit personal detail.
2. Button to delete account.
3. Text view
4. Button to change password.
5. Button to logout.
6. Background banner with title.

10) More menu layout

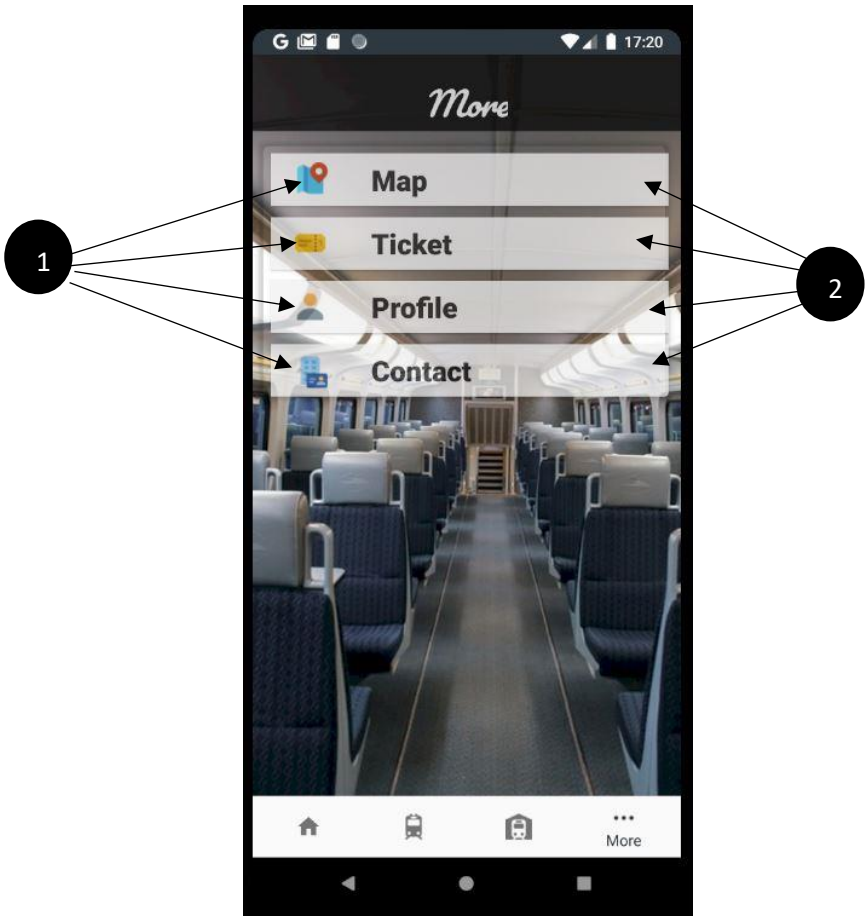


Figure 43 Layout of page more

1. Image icons.
2. Buttons to navigate to map, ticket, profile and contact page.

11) Driver Menu

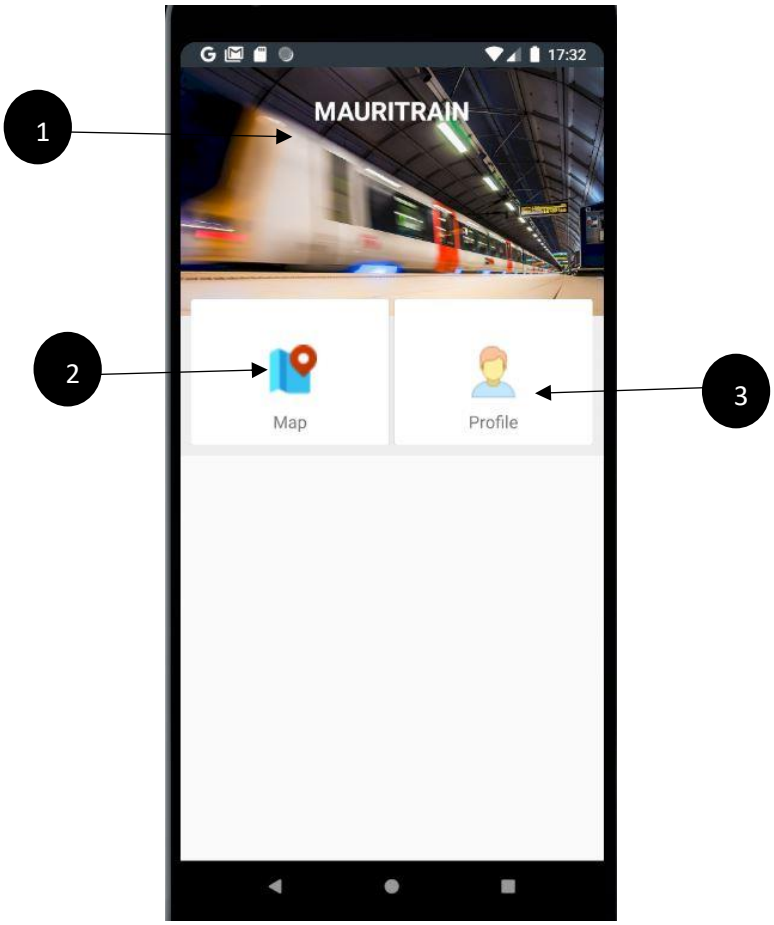


Figure 44 Layout of driver Main menu

1. Background banner with title.
2. Map card view menu icon link to Map screen.
3. Profile card view icon link to Profile Screen.

12) Station display layout

Figure 45 Layout for station display

1. Text view for station name text.
2. Text view for actual date.
3. Text view for actual time.
4. List view to display train list for actual date and time.

CHAPTER 7

IMPLEMENTATION

Implementation is the execution of the project.

7.1 Tools Selection

Android studio version 3.5 was downloaded which is free and Arduino version 1.8.3 was also downloaded.

7.1.2 Development Environment

Both Android studio and Arduino IDE were connected to the firebase.

Android studio was connected directly to the firebase by clicking tools and then Firebase.

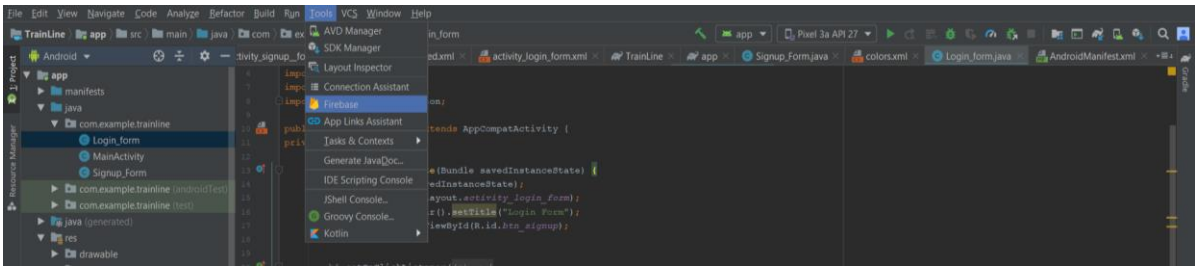


Figure 46 Android Studio Firebase

Arduino IDE was connected to the firebase by uploading the library firebase and adding the link of the firebase.

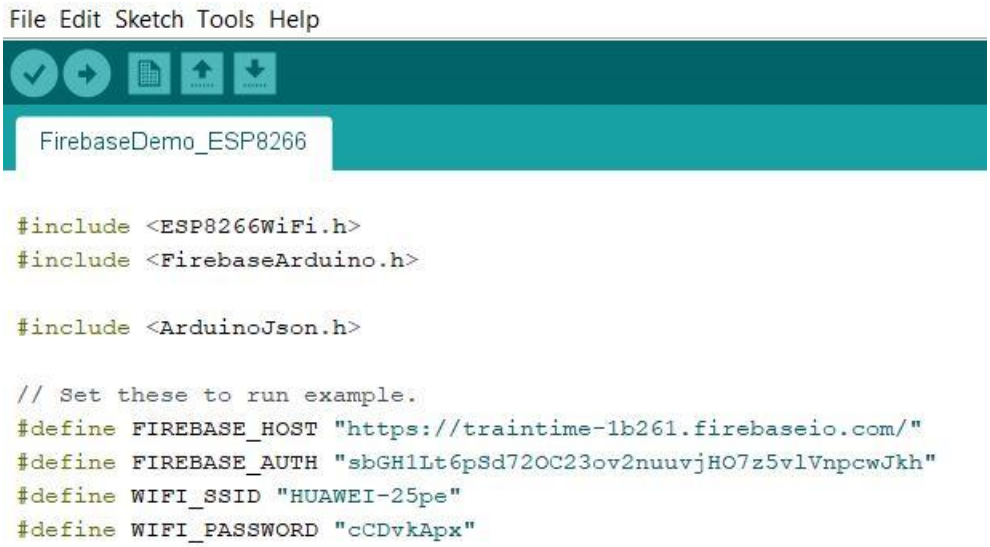


Figure 47 Arduino Firebase setting

7.1.2 Software Specifications

Software	**Description**
Operating System	Window 10 64 bits
Development Environment	Android Studio version 3.5 Arduino IDE version 1.8.3
Libraries	Firebase Liquid crystal ESP8266 RFID RC522

Table 3 Software Specifications

7.1.3 Logo Implementation

Tailor Brands Studio was used to design the logo of the application. Tailor Brands Studio is an online application to personalize and design logo. The application is named Mauritrain and the logo was designed as shown below.

Figure 48 Application Logo

7.2 Activity-lifecycle concept for Android

An android application is made up of different types of components.

1. Activities

   An activity is a screen shown to the user when he interacts with the application. An application consists of multiple activities but only one activity runs at a time. The Main Activity is executed when the application is being launched for the first time. In this project, the first main launcher is the MainActivity.java which display the home menu of admin. The MainActivity.java is connect to its MainActivity.xml which display the user interface of the home menu.

2. Lifecycle

   The activity provides different states of activity lifecycle to navigate and invokes callbacks methods.

   - onCreate() – the activity is called only once during the life to create the user interface.
   - onResume() – called when the user re-interacts with the activity.
   - onPause() – called when the activity is no longer foreground and called before the activity is not visible anymore.
   - onStop() – called when the activity is no longer visible to the user.
   - onStart() – called when the activity become interactive.

7.3 System Implementation

7.3.1 Registration

All the field has validation to prevent the user from entering anything. There is error message display at each field to guide the user. Regular expression has been used to implement the validation of each field.

7.3.1.1 Regular Expression

A regular expression is a sequence of different characters to make a pattern.

```
public class Signup_Form extends AppCompatActivity {
    private static final Pattern PASSWORD_PATTERN =
            Pattern.compile("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@@#$%^&*_=+-]).{6,15}$");
    private static final Pattern FULLNAME_PATTERN =
            Pattern.compile("^(?=.*[a-zA-Z]).{2,60}$");
    private static final Pattern USERNAME_PATTERN =
            Pattern.compile("^(?=.*[a-zA-Z]).{2,35}$");
```

Figure 49 Regular expression

Password Pattern

The password should contain at least one upper case letter, one lower case letter, one digit, one special character and should contain at least 6 characters.

```
("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@@#$%^&*_=+-]).{6,15}$");
```

(?=.*[a-z]) – at least 1 lower case letter

(?=.*[A-Z]) – at least 1 upper case letter

(?=.*[0-9]) – at least 1 digit

(?=.*[!@@#$%^&*_=+-]) – at least 1 special character

.{6,15} – at least 6 characters and maximum 15 characters

Full Name Pattern

The user name should contain letters only. It can be upper case and lower-case letter.

```
("^(?=.*[a-zA-Z]).{2,60}$");
```

(?=.*[a-zA-Z]) – any letter

.{2,60} – at least 2 characters and maximum 60 characters

User Name Pattern

The user name should contain letters only. It can be upper case and lower-case letter.

```
("^(?=.*[a-zA-Z]).{2,35}$");
```

(?=.*[a-zA-Z]) – any letter

.{2,35} – at least 2 characters and maximum 35 characters

Email Pattern

The pattern checks if the email address enters follow the real form of email. For example, the form abc@example.com. If the email address is not written this form the user won't be able to move forward.

```
public static final Pattern EMAIL_ADDRESS
    = Pattern.compile(
        "[a-zA-Z0-9\\+\\.\\_\\%\\-\\+]{1,256}" +
        "\\@" +
        "[a-zA-Z0-9][a-zA-Z0-9\\-]{0,64}" +
        "(" +
            "\\." +
            "[a-zA-Z0-9][a-zA-Z0-9\\-]{0,25}" +
        ")+"
    );
```

Figure 50 Email Regular expression

```
"[a-zA-Z0-9\\+\\.\\_\\%\\-\\+]{1,256}"
```

Any letter or digit and at least 1 character and maximum 256 characters

```
"\\@"
```

@ character should enter

```
"[a-zA-Z0-9][a-zA-Z0-9\\-]{0,64}"
```

Any letter or digit and maximum 64 characters

```
"\\."
```

. character

```
"[a-zA-Z0-9][a-zA-Z0-9\\-]{0,25}"
```

Any letter or digit and maximum 25 characters

7.3.1.2 Error Message

Full Name error message

```
((TextView)spinner.getSelectedView()).setError("Error message");
if(TextUtils.isEmpty(fname)){
   // Toast.makeText(Signup_Form.this, "Please enter Email", Toast.LENGTH_LONG).show();
    ed1.setError("Please enter Fullname");
    return;
}else if (!FULLNAME_PATTERN.matcher(fname).matches()) {
    ed1.setError("Fullname should contains characters only");
    return;
} else {
    ed1.setError(null);
}
```

Figure 51 Error message for full name

User Name error message

```
if(TextUtils.isEmpty(uname)){
    // Toast.makeText(Signup_Form.this, "Please enter Email", Toast.LENGTH_LONG).show();
    ed2.setError("Please enter Username");
    return;
}else if (!USERNAME_PATTERN.matcher(fname).matches()) {
    ed2.setError("Please enter a valid username");
    return;
} else {
    ed2.setError(null);
}
```

Figure 52 Error message for user name

Email error message

```
if(TextUtils.isEmpty(emai)){
    // Toast.makeText(Signup_Form.this, "Please enter Email", Toast.LENGTH_LONG).show();
     ed3.setError("Please enter Email");
     return;
}else if (!Patterns.EMAIL_ADDRESS.matcher(emai).matches()) {
     ed3.setError("Please enter a valid email address");
     return;
} else {
     ed3.setError(null);
}
```

Figure 53 Error message for email

Password and confirm password error message

```
if(TextUtils.isEmpty(pass)){
    // Toast.makeText(Signup_Form.this, "Please enter Password", Toast.LENGTH_LONG).show();
     ed4.setError("Please enter Password");
     return;
}else if (!PASSWORD_PATTERN.matcher(pass).matches()){
     ed4.setError("Password should contains Capitals, small letters, number and special character");
     return;
} else {
     ed4.setError(null);
}

if(TextUtils.isEmpty(cpass)){
    // Toast.makeText(Signup_Form.this, "Please enter Confirm Password", Toast.LENGTH_LONG).show();
     ed5.setError("Please enter Confirm Password");
     return;
}else if (!PASSWORD_PATTERN.matcher(cpass).matches()){
     ed5.setError("Confirm password should match password above");
     return;
} else {
     ed4.setError(null);
}
```

Figure 54 Error message for password

Radio button to select a gender error message

```
if(radMale.isChecked()){
    gender="Male";
}
if (radFemale.isChecked()){
    gender="Female";
}

if (!radMale.isChecked() && !radFemale.isChecked()){
    radFemale.setError("Select Item");
}else
{
    radFemale.setError(null);
}
```

Figure 55 Error message for radio button

7.3.1.3 Hide and Show password button

This particular code is used to show and hide the password when click on the button. On click if the password is hide the image on the button will be an eye and if the password is shown, a cross eye image will display.

```
btnshow.setOnClickListener((v) → {
        //txtPassword.transformationMethod = HideReturnsTransformationMethod.getInstance();
        // txtPassword.setTransformationMethod(HideReturnsTransformationMethod.getInstance());

        if(ed4.getTransformationMethod().equals(PasswordTransformationMethod.getInstance()))

        //  if (txtPassword.getInputType() != InputType.TYPE_TEXT_VARIATION_PASSWORD)
        {
            btnshow.setImageResource(R.drawable.hide);
            ed4.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
        }
        // else if (txtPassword.getInputType() != InputType.TYPE_TEXT_VARIATION_PASSWORD && btnshow.isPressed()){
        else{
            btnshow.setImageResource(R.drawable.eye);
            ed4.setTransformationMethod(PasswordTransformationMethod.getInstance());
        }
});
```

Figure 56 Hide and show password button

7.3.1.4 Authentication and save in database

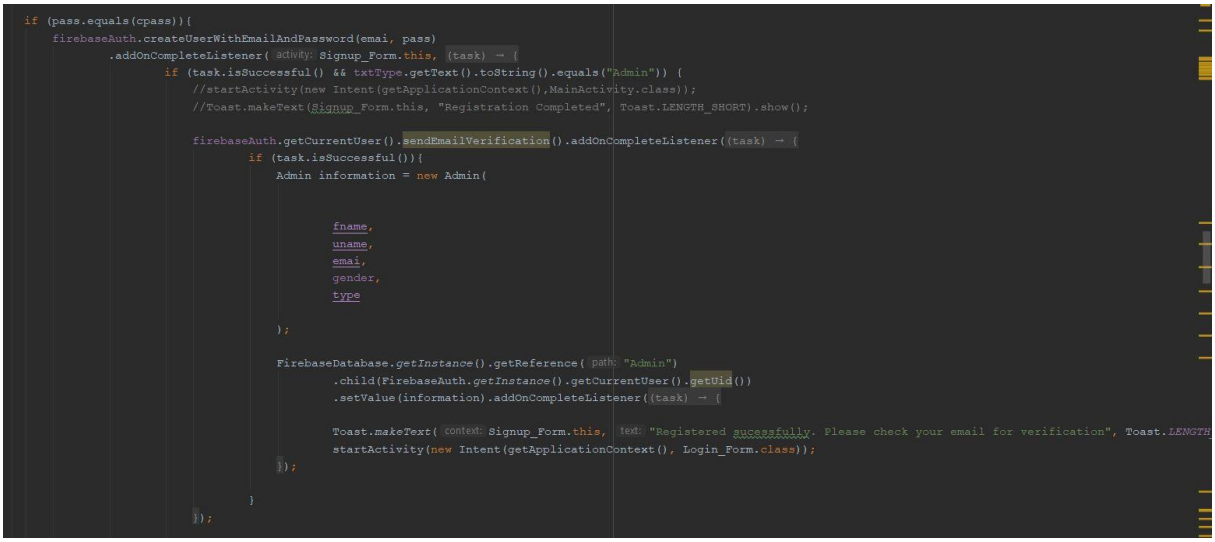


Figure 57 Authentication

The user will be send to the login page after the information has been saved. The email address is send for verification. The user gets a mail to the email enter to confirm his email address. When the user accepts the mail, his email is validated and he can login with the email address.

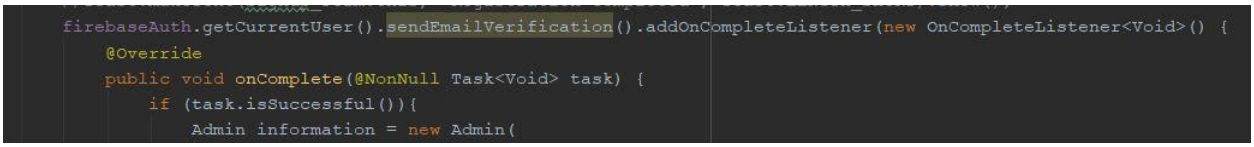


Figure 58 Verification of email

7.3.2 Login

When click on login, the email and password will be verified in the database. If the email and password match it will check whether the user is an admin, a client or a driver and then it will authenticate the user to the correct home page. The admin will be authenticated to the admin home page, the client to the client home page and the driver to the driver home page.

```
firebaseAuth.signInWithEmailAndPassword(email, password)


        .addOnCompleteListener( activity: Login_Form.this, (task) → {
                if (task.isSuccessful() && txtType.getText().toString().equals("Admin")) {
                    if(firebaseAuth.getCurrentUser().isEmailVerified()){
                        startActivity(new Intent(getApplicationContext(),MainActivity.class));
                    }else{
                        Toast.makeText( context: Login_Form.this,  text: "Please verify your email address", Toast.LENGTH_LONG).show();
                    }
```

Figure 59 Login after email has been verified

7.3.3 Admin Part

7.3.3.1 Add New Train

7.3.3.1.2 Populating a spinner

To retrieve the information from the database and populate the spinner selector, the following is used.

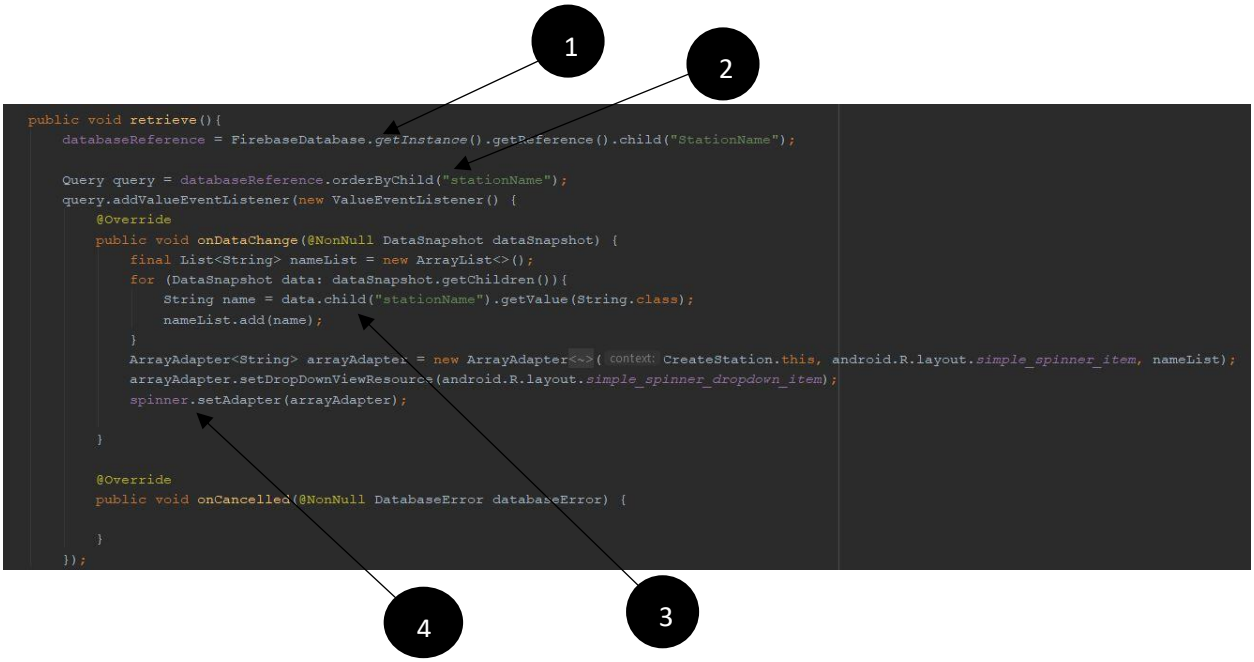


Figure 60 Populating a spinner

1. Get data from table “StationName” from database.
2. Get data from column “stationName” in table “StationName”.
3. Get the values of “stationName”.
4. Populate the spinner with the values of “stationName”.

Furthermore, when a station name is selected from the spinner the platform number and destination is displayed in another spinner accordingly to the position where the station name is found. To achieve this the following codes have been used and a method has been created.

Figure 61 List of platform number

1. A list of platform number and destination station has been created for each station.

Figure 62 Spinner position

2. The number 2 is showing the position number and list name. using this example, when the station name at position 0 is selected, the “BeauBassin” list will be populated in spinner2.

The above method is called onCreate when the application runs.

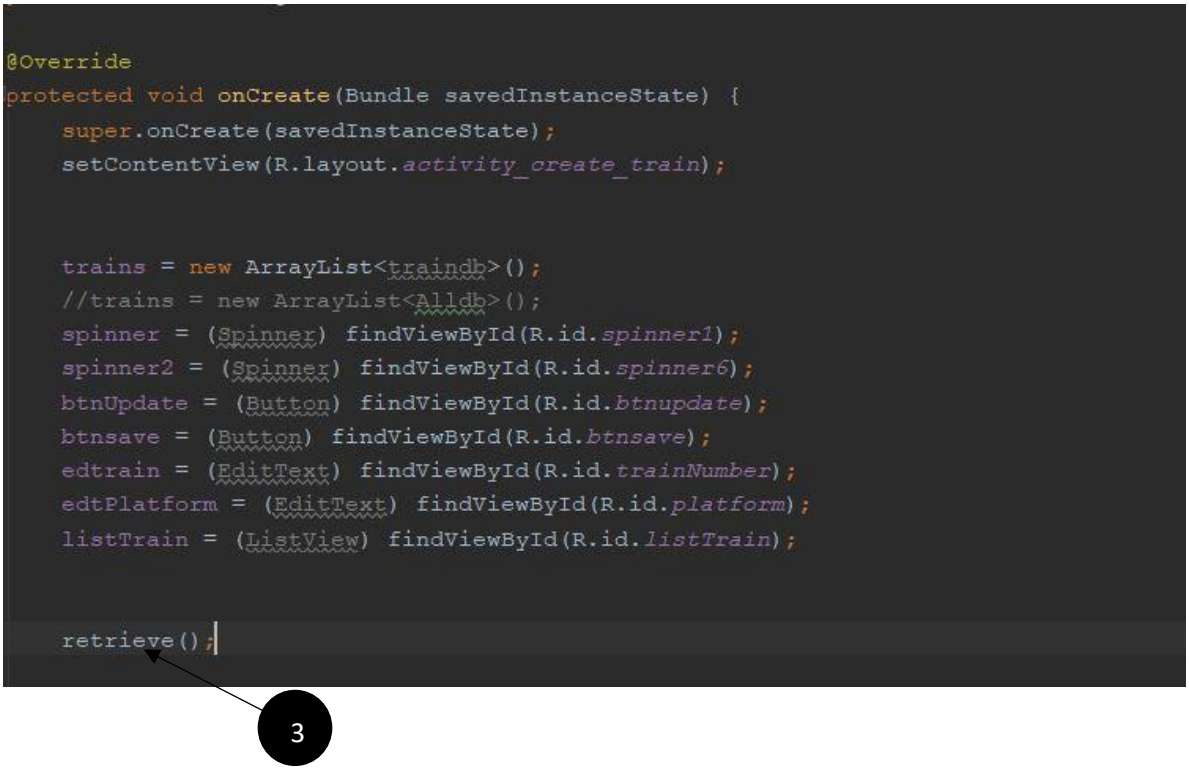


Figure 63 Calling a method

3. Calling the method retrieve.

7.3.3.1.2 Saving the data in database.

A class is created to set and get the data in database.

```
public class traindb {
    private String id;
    private String StationName;
    private String trainNumber;
    private String platformNumber;

    public traindb() {
    }

    public traindb(String id, String stationName, String trainNumber, String platformNumber) {
        this.id = id;
        StationName = stationName;
        this.trainNumber = trainNumber;
        this.platformNumber = platformNumber;
    }

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public String getStationName() { return StationName; }

    public void setStationName(String stationName) { StationName = stationName; }

    public String getTrainNumber() { return trainNumber; }

    public void setTrainNumber(String trainNumber) { this.trainNumber = trainNumber; }

    public String getPlatformNumber() { return platformNumber; }

    public void setPlatformNumber(String platformNumber) { this.platformNumber = platformNumber; }
}
```

Figure 64 Class to get and set data

Before saving the data in the database, all data are converted into string and then save. If the data are successfully inserted in the database a message will be shown “Successfully created” and revert back to the train list page to view.

```
btnsave.setOnClickListener((v) -> {
        databaseReference = FirebaseDatabase.getInstance().getReference( path: "Train");
        //databaseReference = FirebaseDatabase.getInstance().getReference("All");
        String trainNumber = edtrain.getText().toString();
        //String platformNumber = edtPlatform.getText().toString();
        String stationName = spinner.getSelectedItem().toString();
        String platformNumber = spinner2.getSelectedItem().toString();

        if (TextUtils.isEmpty(trainId)) {
            //save
            String id = databaseReference.push().getKey();
            traindb train = new traindb(id, stationName, trainNumber, platformNumber);
            databaseReference.child(id).setValue(train);

            Toast.makeText( context: createTrain.this,  text: "Successfully created", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(), Train.class));
        }
```

Figure 65 Save data in database

7.3.3.2 Create List View to display train list

The layout structure of list view is complex therefore, inheritance has been used to customise the Array Adapter. A new class was created name “trainList”, and it extends the “traindb” class. The “traindb” has no default constructor, so constructors have been defined to match the constructors in the superclass.

```
public class trainList extends ArrayAdapter<traindb> {

    private Activity context;
    private List<traindb> trains;
    //private List<Alldb> trains;
    DatabaseReference databaseReference;
    EditText edtrain, edtPlatform;
    Spinner spinner, spinner1;

    public trainList(Activity context, List<traindb> trains, DatabaseReference databaseReference, EditText edtrain, Spinner spinner, Spinner spinner1) {
        super(context, R.layout.layout_train_list, trains);
        this.context=context;
        this.trains = trains;
        this.databaseReference = databaseReference;
        this.edtrain = edtrain;
        this.spinner = spinner;
        this.spinner1 = spinner1;
    }
```

Figure 66 Create list view

After that the part of overriding getView() method is done. A view is inflated from layout resource in the getView() method. In this prat we have used “layout_train_list” as layout. The view is filled with data once it has been inflated, the getItem(position) is used to access the data and get the values. Finally, these values are set in the layout and return the view.

```
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    LayoutInflater inflater = context.getLayoutInflater();

    View listViewItem = inflater.inflate(R.layout.layout_train_list, root: null, attachToRoot: true);

    TextView textViewName = (TextView) listViewItem.findViewById(R.id.txtTname);
    TextView textViewNumber = (TextView) listViewItem.findViewById(R.id.txtTrain);
    TextView textViewPlatform = (TextView) listViewItem.findViewById(R.id.txtplatformNumber);
    Button btnDelete = (Button) listViewItem.findViewById(R.id.btnDelete2);
    Button btnEdit = (Button) listViewItem.findViewById(R.id.btnEdit2);

    //final Alldb train = trains.get(position);
    final traindb train = trains.get(position);
    textViewName.setText(train.getStationName());
    textViewNumber.setText(train.getTrainNumber());
    textViewPlatform.setText(train.getPlatformNumber());
```

Figure 67 Get values of data

The following code is used to display the values in the list view.

```
protected void onStart() {
    super.onStart();
    //databaseReference = FirebaseDatabase.getInstance().getReference("All");
    databaseReference = FirebaseDatabase.getInstance().getReference( path: "Train");
    Query query = databaseReference.orderByChild("Train").limitToFirst(15);
    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            trains.clear();
            List<String> keys = new ArrayList<>();
            for (DataSnapshot ds : dataSnapshot.getChildren()) {
                keys.add(ds.getKey());
                traindb train = ds.getValue(traindb.class);
                // Alldb train = ds.getValue(Alldb.class);
                trains.add(train);


            }

            trainList trainAdapter = new trainList( context: Train.this, trains, databaseReference, edtrain, spinner, spinner1);
            listTrain.setAdapter(trainAdapter);
```

Figure 68 Display data in list view

7.3.3.3 Edit and Delete data

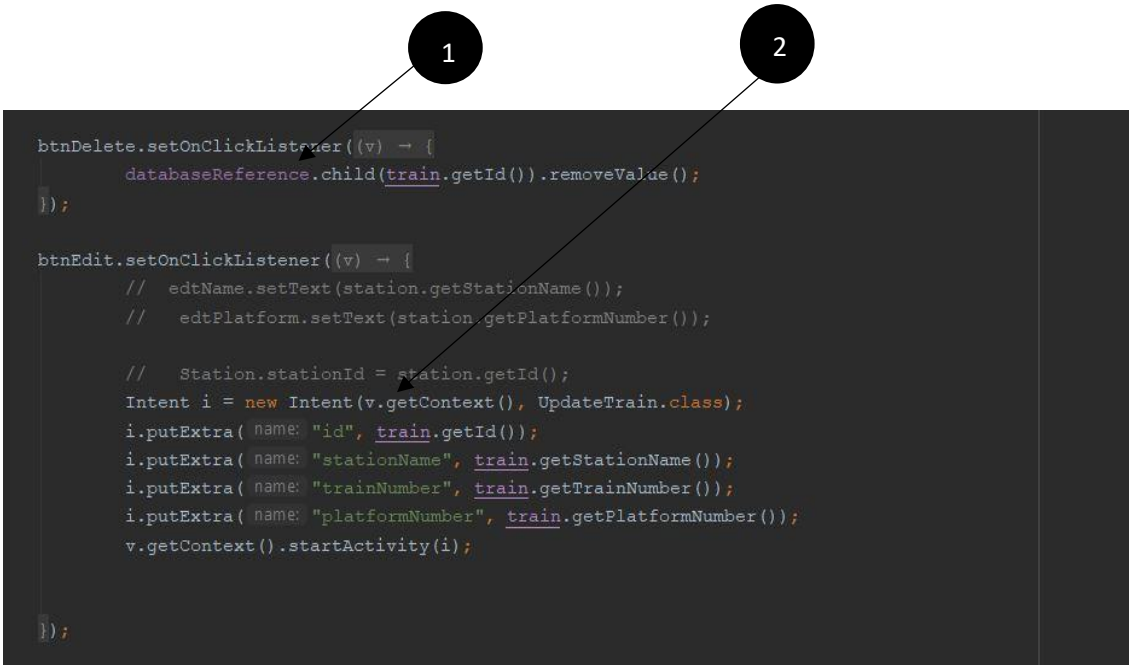


Figure 69 Delete and edit data

1. The data is removed from the database according to its Id.
2. On click on Edit button the current information is push to the edit page according to the current Id.

7.3.3.4 Update Current Data

On the update page, the we get the data we want to change. Hence this is done by getting the values from the specific Id we want to change from the database.

After the data has been changed it is saved and updated in the database.

```
Intent intent = getIntent();
id = intent.getStringExtra( name: "id");
stationName = intent.getStringExtra( name: "stationName");
platformNumber = intent.getStringExtra( name: "platformNumber");
trainNumber = intent.getStringExtra( name: "trainNumber");
//  edtname.setText(stationName);
//edtplatform.setText(platformNumber);


edtrain.setText(trainNumber);

btnsave.setOnClickListener((v) → {
        databaseReference = FirebaseDatabase.getInstance().getReference( path: "Train");
        //databaseReference = FirebaseDatabase.getInstance().getReference("All");
        String trainNumber = edtrain.getText().toString();
        //String platformNumber = edtPlatform.getText().toString();
        String stationName = spinner.getSelectedItem().toString();
        String platformNumber = spinner2.getSelectedItem().toString();

        if (TextUtils.isEmpty(trainId)) {
            //save
            String id = databaseReference.push().getKey();
            traindb train = new traindb(id, stationName, trainNumber, platformNumber);
            databaseReference.child(id).setValue(train);

            Toast.makeText( context: UpdateTrain.this,  text: "Successfully updated", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(), Train.class));
        }
```

Figure 70 Update current data

7.3.3.5 Set Time Clock

Figure 71 Set time clock

1. If the hour is less than 10 then add 0 in front the number. For example, if it's 9 o'clock then it is written as 09:00.
2. If the minute is less than 10 then add 0 in front the number. For example, if it's 2 minutes then it is written as 00:02.
3. Both the hour and minute is written together to display the time in correct form like this "09:02".

7.3.3.6 Search in Text

Search from the database where the data matches the text enter.

```
private void firebaseSearch4(String searchText2){
    databaseReference = FirebaseDatabase.getInstance().getReference( path: "Info");
    Query query = databaseReference.orderByChild("date").startAt(searchText2).endAt(searchText2 + "").limitToFirst(15);


    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {


            infos.clear();
            for (DataSnapshot data: dataSnapshot.getChildren()){
                Infodb info = data.getValue(Infodb.class);
                infos.add(info);
            }

            InfoList infoAdapter = new InfoList( context: Trains.this, infos, databaseReference, edtDate, edtTime, spinner, spinner1, spinner2, txttime, s
            list.setAdapter(infoAdapter);

        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
```

Figure 72 Search box

7.3.3.7 Search by filtering

a) There are three spinners display available in the interface, the user has to choose which station name, platform number, destination and train number he wishes to get. The list of train will be display according to this information enter.

```
public void T2(){
    databaseReference = FirebaseDatabase.getInstance().getReference().child("Train");

    // Query query = databaseReference.orderByChild("trainNumber").startAt("0003").endAt("0004");
    Query query = databaseReference.orderByChild("platformNumber").equalTo("2(Quatre-Bornes)");
    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            final List<String> nameList = new ArrayList<>();

            for (DataSnapshot data : dataSnapshot.getChildren()) {
                String name = data.child("trainNumber").getValue(String.class);
                nameList.add(name);
            }
            ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>( context: Trains.this, android.R.layout.simple_spinner_item, nameList);
            arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
            spinner2.setAdapter(arrayAdapter);
            spinner2.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
                @Override
                public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                    String itemSelect = nameList.get(position);


                    dbtrain.setText(spinner2.getSelectedItem().toString());
                   // firebaseSearch2();
                    if (mSearch.getText().toString().trim().length() == 0){
                        firebaseSearch2();
                    }else if (mSearch.getText().toString().trim().length() >0) {
                        firebaseSearch3();
                    }
                }

                @Override
                public void onNothingSelected(AdapterView<?> parent) {
```

Figure 73 Search by filtering part 1

```
private void firebaseSearch2(){
    dbtrain.setText(spinner2.getSelectedItem().toString());
    mSearch.getText().toString();
    final String trainNumber = dbtrain.getText().toString().trim();
    databaseReference = FirebaseDatabase.getInstance().getReference().child("Info");
    //Query query = databaseReference.orderByChild("trainNumber").equalTo("0001");

    Query query = databaseReference.orderByChild("trainNumber");


    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {


            infos.clear();

            for (DataSnapshot ds : dataSnapshot.getChildren()) {
                // String train1 = dataSnapshot.child("trainNumber").getValue().toString();
                String train1 = ds.child("trainNumber").getValue(String.class);
                if (trainNumber.equals(train1)) {
                    Infodb train = ds.getValue(Infodb.class);

                    infos.add(train);

                }

            }

            InfoList infoAdapter = new InfoList( context: Trains.this, infos, databaseReference, edtDate, edtTime, spinner, spinner1, s
            list.setAdapter(infoAdapter);
```

Figure 74 Search by filtering part 2

b) The user can filter by using the three spinners and by entering date also. He will get the list of trains for that specific date and according to the train number, station name and platform number.

```
private void firebaseSearch3(){
    dbtrain.setText(spinner2.getSelectedItem().toString());
    final String search = mSearch.getText().toString();
    final String trainNumber = dbtrain.getText().toString().trim();
    databaseReference = FirebaseDatabase.getInstance().getReference().child("Info");
    //Query query = databaseReference.orderByChild("trainNumber").equalTo("0001");

    Query query = databaseReference.orderByChild("trainNumber");


    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {


            infos.clear();

            for (DataSnapshot ds : dataSnapshot.getChildren()) {
                // String train1 = dataSnapshot.child("trainNumber").getValue().toString();
                String train1 = ds.child("trainNumber").getValue(String.class);
                if (trainNumber.equals(train1)) {
                    Infodb train = ds.getValue(Infodb.class);
                    if (train.getDate().equals(search)) {

                        infos.add(train);
                    }
                }

            }

            InfoList infoAdapter = new InfoList( context: Trains.this, infos, databaseReference, edtDate, edtTime, spinner, spinner1, spinner2, txttime, s
            list.setAdapter(infoAdapter);
```

Figure 75 Search filtering part 3

7.3.3.8 Send Email to client

In this part the admin is able to answer the client queries via email. The Id and the email address of the client is retrieve and get the values.

Figure 76 Send email to client

1. Mail to: Email of the client.
2. Subject: The query he is replying to.
3. Message: Write the answer for the query ask.

7.3.3.9 Profile Detail

To get the current details of the user, we need to get the current user and user Id.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
if (user != null) {
    // Name, email address, and profile photo Url
    // String name = user.getDisplayName()
    String email = user.getEmail();


    // Check if user's email is verified
    boolean emailVerified = user.isEmailVerified();

    // FirebaseUser.getIdToken() instead.
    String uid = user.getUid();

    //txtname.setText(name);
    txtemail.setText(email);


}
```

Figure 77 Get current user id

After getting the current user Id and email, we search in the database table Admin where the Id and email matches and then display all the required details of the user.

```
firebaseDatabase = FirebaseDatabase.getInstance();
databaseReference = firebaseDatabase.getReference().child("Admin");


Query query = databaseReference.orderByChild("Email").equalTo(user.getEmail());

query.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        for (DataSnapshot ds: dataSnapshot.getChildren()){
            // String name = ds.child("Email").getValue(String.class);
            // if (ds.child("Email").getValue().equals(txtemail)) {
            // txtname.setText(ds.child("Username").getValue(String.class));
            String Username = "" + ds.child("Username").getValue();
            String Gender = "" + ds.child("Gender").getValue();
            String fullName = "" + ds.child("fullName").getValue();
            //  String id = "" + ds.child("id").getValue();
            String Admin = "" + ds.child("Admin").getValue();
            //String email = "" + ds.child("Email").getValue();
            // String id = databaseReference.push().getKey();
            txtname.setText(Username);
            txtgender.setText(Gender);
            txtfull.setText(fullName);
            //  txtid.setText(id);
            txtAdmin.setText(Admin);
            // txtemail.setText(Email);

            // }

        }

    }
```

Figure 78 Display user details

7.3.3.10 Logout and Delete account

```
btnLogout.setOnClickListener((v) → {
        FirebaseAuth.getInstance().signOut();
        finish();
        startActivity(new Intent(getApplicationContext(),Login_Form.class));
});
```

2

1

Figure 79 Logout

1. The user session is destroyed.
2. The current page activity is finished.

When the user will delete his account, a message will be displayed as shown below.

```
btndelete.setOnClickListener((v) → {
        final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        if (user != null) {
            user.delete().addOnCompleteListener((task) → {
                    if (task.isSuccessful()) {
                        Toast.makeText( context: AdminProfile.this,  text: "Your profile is deleted:( Create a account now!", Toast.LENGTH_SHORT).show();
                        databaseReference.child(user.getUid()).removeValue();
                        startActivity(new Intent(getApplicationContext(),Login_Form.class));
                    } else {
                        Toast.makeText( context: AdminProfile.this,  text: "Failed to delete your account!", Toast.LENGTH_SHORT).show();
                    }
            });

        }
});
```

Figure 80 Delete Account

7.3.3.11 Edit Password

The password field has the same validation and error message as mention above in the registration part. The button to hide and show the password is the same as above.

```
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

    user.updatePassword(txtnew.getText().toString().trim())
            .addOnCompleteListener((task) → {
                    if (task.isSuccessful()) {

                        Toast.makeText( context: AdminPW.this,  text: "Password updated", Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(getApplicationContext(),AdminProfile.class));
                    }
            });
```

Figure 81 Edit password

7.3.3.12 Edit Personal Detail

Get current user Id and his personal details on an edit text field. The field has the same validation and error messages as mention above in the registration part.

To update the email address the following codes are used.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

user.updateEmail(txtemail.getText().toString().trim())
        .addOnCompleteListener((task) → {
                if (task.isSuccessful()) {

                    Toast.makeText( context: AdminEdit.this,
                            text: "Email address updated",
                            Toast.LENGTH_SHORT).show();
                }
        });
```

Figure 82 Update Email address

To the save the information after editing the following codes are used.

```
DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference( path: "Admin").child(user.getUid());
String Admin = txtAdmin.getText().toString();
// String key = databaseReference.push().getKey();
String Username = txtname.getText().toString();
String Email = txtemail.getText().toString();
String fullName = txtfull.getText().toString();
String Gender = txtgender.getText().toString();


Admin info = new Admin( fullName, Username, Email, Gender, Admin);
// databaseReference.child(key).setValue(info);
databaseReference.setValue(info);
Toast.makeText( context: AdminEdit.this, text: "Successfully update", Toast.LENGTH_SHORT).show();
startActivity(new Intent(getApplicationContext(),AdminProfile.class));
```

Figure 83 Save the editing information

7.3.4 Forget Password

If a user has forgotten his password he just need to click on forget password and add his email address and send it. He will get a mail afterward asking to reset his password, he needs to click on it and reset his password.

```
mAuth = FirebaseAuth.getInstance();
btnEmail.setOnClickListener((v) -> {

        String email = txtEmail.getText().toString().trim();

        if (TextUtils.isEmpty(email)) {
           // Toast.makeText(getApplication(), "Enter your registered email id", Toast.LENGTH_SHORT).show();
            txtEmail.setError("Please enter Email");
            return;
        }else if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            txtEmail.setError("Please enter a valid email address");
            return;
        } else {
            txtEmail.setError(null);
        }

        mAuth.sendPasswordResetEmail(email).addOnCompleteListener((task) -> {
                if (task.isSuccessful()) {
                    Toast.makeText( context: ForgetPassword.this,  text: "We have sent you instructions to reset your password!", Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent( packageContext: ForgetPassword.this, Login_Form.class);
                } else {
                    Toast.makeText( context: ForgetPassword.this,  text: "Failed to send reset email!", Toast.LENGTH_SHORT).show();
                }
        });


});
```

Figure 84 Forget Password

7.3.5 Client Side

7.3.5.1 Home page Image Slide show

An anim_slide.xml was created in drawable to add the images.

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:duration="4000"
        android:drawable="@drawable/back1"/>

    <item
        android:duration="4000"
        android:drawable="@drawable/back2"/>

    <item
        android:duration="4000"
        android:drawable="@drawable/back3"/>

    <item
        android:duration="4000"
        android:drawable="@drawable/back4"/>

</animation-list>
```

Figure 85 Image slide show

Then Animation Drawable effect was added to the images folder when called.

```
ImageView iv_background = root.findViewById(R.id.iv_background);

AnimationDrawable animationDrawable = (AnimationDrawable) iv_background.getDrawable();
animationDrawable.start();
```

Figure 86 Add animation drawable effect

7.3.5.2 Search on Edit Text

If the text enters matched and equal to any data in the database it will be displayed on the list view.

```
private void firebaseSearch(String searchText){
    final String trainNumber = mSearch.getText().toString().trim();
    databaseReference = FirebaseDatabase.getInstance().getReference( path: "Info");
    //Query query = databaseReference.orderByChild("platformNumber").startAt(searchText).endAt(searchText + "").limitToFirst(15);
    Query query = databaseReference.orderByChild("Info");

    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {


            infos.clear();
            for (DataSnapshot data: dataSnapshot.getChildren()){
                //String train1 = data.child("trainNumber").getValue(String.class);
               // if (trainNumber.equals(train1)) {
                 //   Infodb info = data.getValue(Infodb.class);
                   // infos.add(info);

                //}
                Infodb info = data.getValue(Infodb.class);
                if (info.getTrainNumber().equals(trainNumber)) {

                    infos.add(info);
                }
                else if (info.getPlatformNumber().equals(trainNumber)){
                    infos.add(info);
                }
                //infos.add(info);
            }

            client infoAdapter = new client(getActivity(), infos, databaseReference, edtDate, edtTime, spinner, spinner1, spinner2, txttime, spinner10);
            list.setAdapter(infoAdapter);
```

Figure 87 Search part 1

7.3.5.3 Search by Filtering

The user can filter his search by specifying which station name, platform number and train number he is search for.


```
public void T1(){
    databaseReference = FirebaseDatabase.getInstance().getReference().child("Train");
    //  Query query = databaseReference.orderByChild("trainNumber").startAt("0001").endAt("0002");
    Query query = databaseReference.orderByChild("platformNumber").equalTo("1(Trianon)");
    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            final List<String> nameList = new ArrayList<>();
            nameList.add( index: 0,  element: "Choose a Train");
            for (DataSnapshot data : dataSnapshot.getChildren()) {
                String name = data.child("trainNumber").getValue(String.class);
                nameList.add(name);
            }
            ArrayAdapter<String> arrayAdapter = new ArrayAdapter<~>(getActivity(), android.R.layout.simple_spinner_item, nameList);
            arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
            spinner2.setAdapter(arrayAdapter);

            spinner2.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
                @Override
                public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                    // String itemSelect = nameList.get(position);
                    // if (parent.getItemAtPosition(position).equals("Choose a Station")){

                    // firebaseSearch1();
                    //   }

                    //if (position == 0){
                    // dbtrain.setText("0001");
                    dbtrain.setText(spinner2.getSelectedItem().toString());
                    if (mSearch.getText().toString().trim().length() == 0){
                        firebaseSearch2();
                    }else if (mSearch.getText().toString().trim().length() >0) {
                        firebaseSearch3();
                    }
```

Figure 88 Search part 2

1. This method is used when the Edit text is null.
2. This method is used when the Edit text contains a text.

```
private void firebaseSearch2(){
    dbtrain.setText(spinner2.getSelectedItem().toString());
    mSearch.getText().toString();
    final String trainNumber = dbtrain.getText().toString().trim();
    databaseReference = FirebaseDatabase.getInstance().getReference().child("Info");
    //Query query = databaseReference.orderByChild("trainNumber").equalTo("0001");

    Query query = databaseReference.orderByChild("trainNumber");

    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            infos.clear();

            for (DataSnapshot ds : dataSnapshot.getChildren()) {
                // String train1 = dataSnapshot.child("trainNumber").getValue().toString();
                String train1 = ds.child("trainNumber").getValue(String.class);
                if (trainNumber.equals(train1)) {
                    Infodb train = ds.getValue(Infodb.class);

                    infos.add(train);

                }

            }
            clientStation_List infoAdapter = new clientStation_List(getActivity(), infos, databaseReference, edtDate, edtTime, sp
            list.setAdapter(infoAdapter);

        }
```

Figure 89 Search part 3

The user can use the second method to do his filtering by specifying which date on the edit text and the station name, platform number and train number on the spinners.

```
private void firebaseSearch3(){
    dbtrain.setText(spinner2.getSelectedItem().toString());
    final String search = mSearch.getText().toString();
    final String trainNumber = dbtrain.getText().toString().trim();
    databaseReference = FirebaseDatabase.getInstance().getReference().child("Info");
    //Query query = databaseReference.orderByChild("trainNumber").equalTo("0001");

    Query query = databaseReference.orderByChild("trainNumber");


    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {


            infos.clear();

            for (DataSnapshot ds : dataSnapshot.getChildren()) {
                // String train1 = dataSnapshot.child("trainNumber").getValue().toString();
                String train1 = ds.child("trainNumber").getValue(String.class);
                if (trainNumber.equals(train1)) {
                    Infodb train = ds.getValue(Infodb.class);
                    if (train.getDate().equals(search)) {

                        infos.add(train);
                    }
                }

            }


            clientStation_List infoAdapter = new clientStation_List(getActivity(), infos, databaseReference, edtDate, edtTime, spinne
            list.setAdapter(infoAdapter);
```

Figure 90 Search part 4

7.3.5.4 Client Map

An image map has been added with exact position of different station location to facilitate user. The user can see the nearest station near to their location. The user just need to click on any station and then send their queries to the admin.

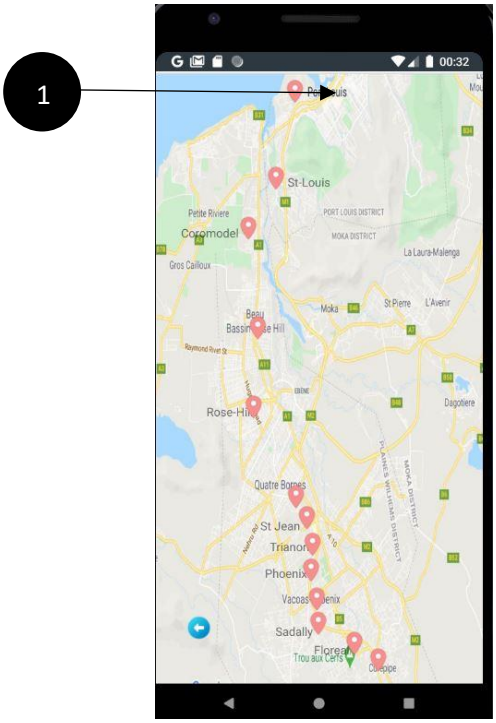


Figure 91 Map

1. Station Location

On click on each station location icon. For each station, there is a page which will display according to the chosen station name where user can send their queries.

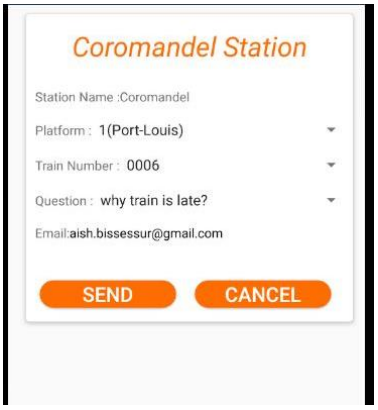


Figure 92 Query

The position of the station was coordinated.

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/map">

    <ImageButton
        android:id="@+id/back"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginRight="311dp"
        android:layout_marginBottom="37dp"
        android:background="@android:color/transparent"
        android:src="@drawable/iconback" />

    <ImageButton
        android:id="@+id/portlouis"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginRight="210dp"
        android:layout_marginBottom="696dp"
        android:background="@android:color/transparent"
        android:src="@drawable/marker1" />

    <ImageButton
        android:id="@+id/curepipe"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_alignParentRight="true"
```

Figure 93 Station location coordinates

The following code was used to implement the above interface.

```
final String question[] = {"why the train is late?", "Last Train time", "First Train time"};

ArrayAdapter<String> arrayAdapter = new ArrayAdapter<~>( context: Coromandel.this, android.R.layout.simple_spinner_item, question);
arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner2.setAdapter(arrayAdapter);

btnsend.setOnClickListener((v) -> {
        databaseReference = FirebaseDatabase.getInstance().getReference( path: "Question");
        String trainNumber = spinner1.getSelectedItem().toString();
        String stationName = txtstation.getText().toString();
        String question = spinner2.getSelectedItem().toString();
        String platformNumber = spinner.getSelectedItem().toString();
        String Email = txtemail.getText().toString();


        if (TextUtils.isEmpty(infoId)){
            //save
            String id = databaseReference.push().getKey();
            Questiondb info = new Questiondb(id, trainNumber, stationName, platformNumber, question, Email);
            databaseReference.child(id).setValue(info);

            Toast.makeText( context: Coromandel.this, text: "Successfully created", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(), ClientMap.class));
        }

});

btncancel.setOnClickListener((v) -> {
        startActivity(new Intent(getApplicationContext(), Map.class));
});

retrieve1();
```

Figure 94 Send query to admin

7.3.6 Driver Part

7.3.6.1 Driver Map

The driver can notify the admin if the train will be late at which station and reason for being late. After the admin received the notification he will be able to adjust the time so that the user knows about the lateness.

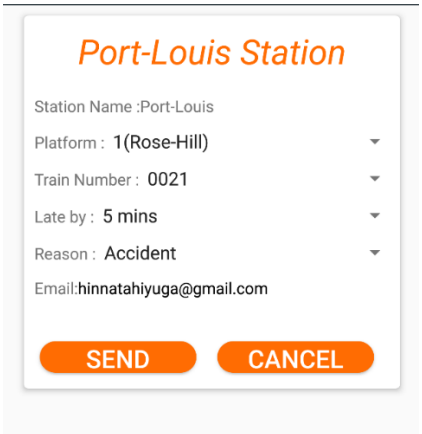
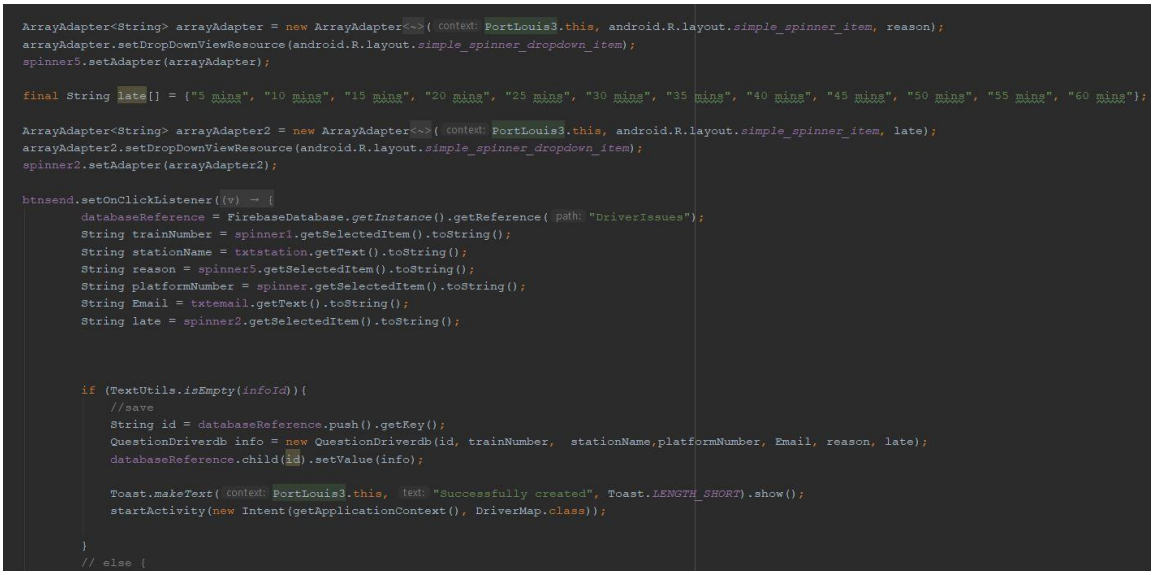


Figure 95 Notification about lateness



```
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<~>( context: PortLouis3.this, android.R.layout.simple_spinner_item, reason);
arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner5.setAdapter(arrayAdapter);

final String late[] = {"5 mins", "10 mins", "15 mins", "20 mins", "25 mins", "30 mins", "35 mins", "40 mins", "45 mins", "50 mins", "55 mins", "60 mins"};

ArrayAdapter<String> arrayAdapter2 = new ArrayAdapter<~>( context: PortLouis3.this, android.R.layout.simple_spinner_item, late);
arrayAdapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner2.setAdapter(arrayAdapter2);

btnsend.setOnClickListener((v) -> {
        databaseReference = FirebaseDatabase.getInstance().getReference( path: "DriverIssues");
        String trainNumber = spinner1.getSelectedItem().toString();
        String stationName = txtstation.getText().toString();
        String reason = spinner5.getSelectedItem().toString();
        String platformNumber = spinner.getSelectedItem().toString();
        String Email = txtemail.getText().toString();
        String late = spinner2.getSelectedItem().toString();



        if (TextUtils.isEmpty(infoId)){
            //save
            String id = databaseReference.push().getKey();
            QuestionDriverdb info = new QuestionDriverdb(id, trainNumber,  stationName,platformNumber, Email, reason, late);
            databaseReference.child(id).setValue(info);

            Toast.makeText( context: PortLouis3.this,  text: "Successfully created", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(), DriverMap.class));

        }
        // else {
```

Figure 96 Send notification

7.3.7 Station Display

In this part the train list is display according to each station and current date and time.

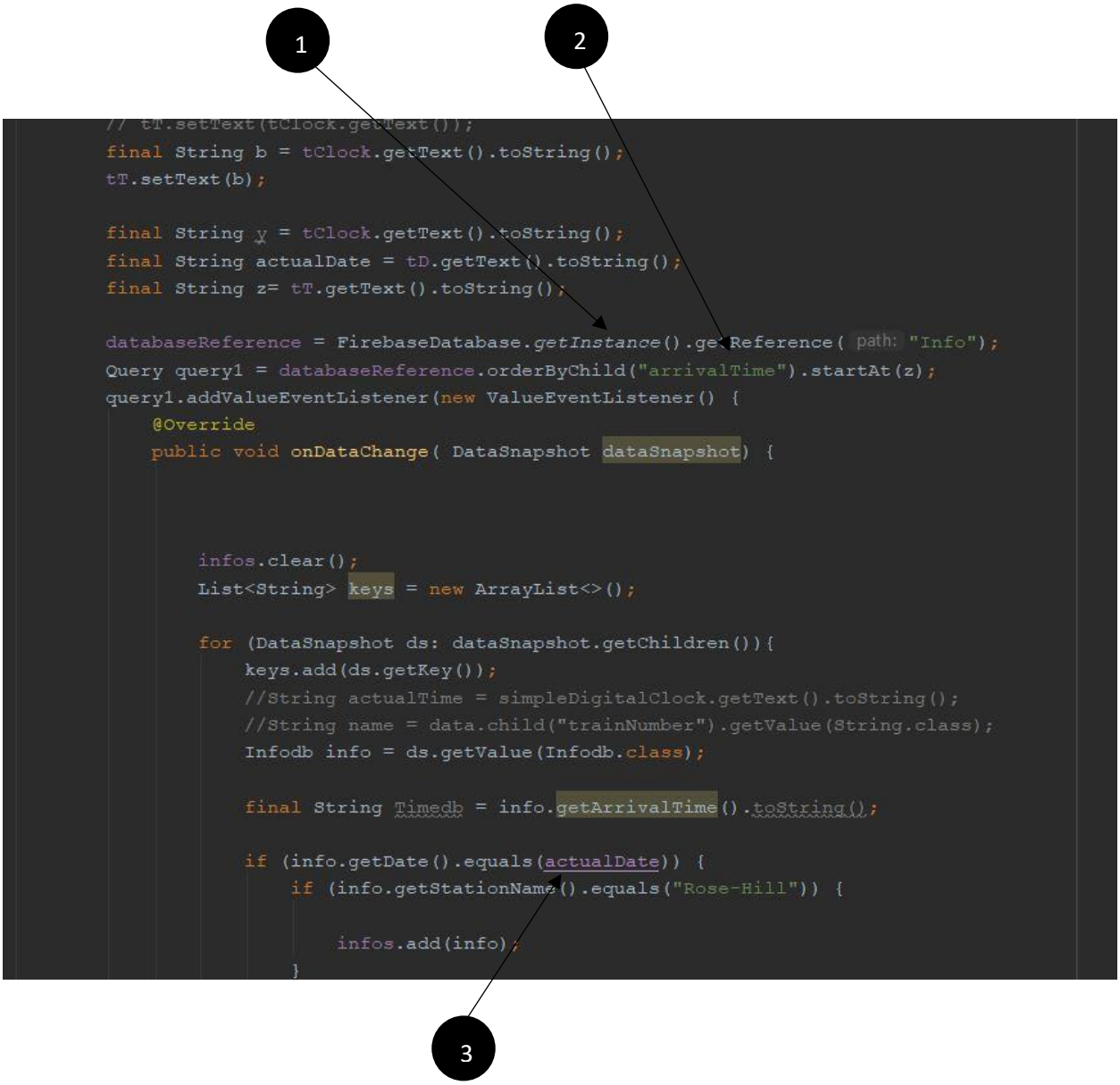


Figure 97 Station Display

1. From the database table Info.
2. Search from the column arrival Time data which start from the actual time.
3. Get the date from the database which is equal to the actual date.

Figure 98 Refresh page each minute

4. The method is reloaded each minute to change the list accordingly to the actual time.

7.4 Arduino Implementation

In this part all the selected part of the Arduino is mounted together and is connected to the firebase to record the train number and the actual arrival time.

7.4.1 Check connection

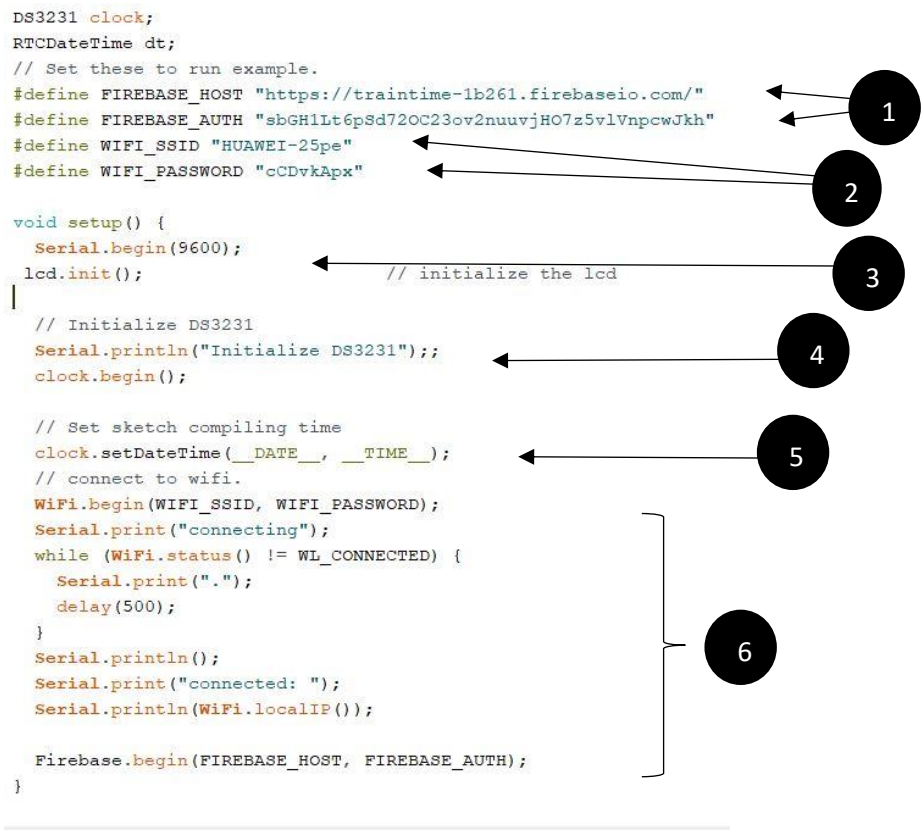


Figure 99 Arduino check connection

1. Connect to firebase.
2. Connect to Wi-Fi.
3. Initialise the LCD screen.
4. Initialise the Real Time Clock.
5. Set the actual date and time.
6. Check the Wi-Fi connection.

```
void loop() {
  // set value
  Firebase.setFloat("number", 42.0);
  // handle error
  if (Firebase.failed()) {
      Serial.print("setting /number failed:");
      Serial.println(Firebase.error());
      return;
  }
  delay(1000);

dt = clock.getDateTime();

  // For leading zero look to DS3231_dateformat example

   lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print(dt.year);   lcd.print("-");
  lcd.print(dt.month);  lcd.print("-");
  lcd.print(dt.day);    lcd.print(" ");
    lcd.print(dt.hour);    lcd.print(":");
   lcd.print(dt.minute);
```

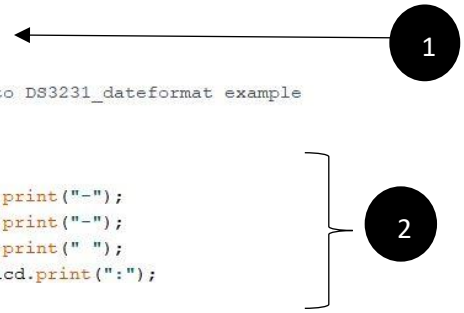


Figure 100 RTC module get the actual date and time

1. Get the actual date and time.
2. Print the date and time on the LCD Screen.

7.4.2 Read the sensor

```
// set string value
// Reset the loop if no new card present on the sensor/reader. This saves the e
if ( ! mfrc522.PICC_IsNewCardPresent()) {
  return;
}

// Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial()) {
  return;
}

// Dump debug info about the card; PICC_HaltA() is automatically called
mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
if (mfrc522.uid(52 0F 04 1F)){
 lcd.setCursor(2,1);
 lcd.print("Train"); lcd.print(" "); lcd.print("0004");
  lcd.println(clock.getDateTime());

 Firebase.setString("Time", "Train");
}
 if (mfrc522.uid(47 8F FF 04 1F)){
 lcd.setCursor(2,1);
 lcd.print("Train"); lcd.print(" "); lcd.print("0003");
  Firebase.setString("Time", "dt");
  lcd.println(clock.getDateTime());
Firebase.setString("Time", "Train");
}
// handle error
if (Firebase.failed()) {
    Serial.print("setting /message failed:");
    Serial.println(Firebase.error());
    return;
```

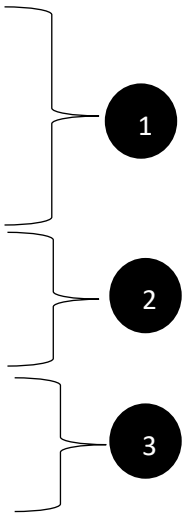


Figure 101 Read the uid of sensor

1. The RFID reader read the uid of the sensor.
2. If the uid detect is equals to the train number 0004, set the train number and the actual time in the firebase.
3. If the uid detect is equals to the train number 0004, set the train number and the actual time in the firebase.

7.4.3 Connect the result with mobile application

When the actual time is read and recorded, the mobile application will retrieve the data from the firebase and will make the difference to see if the train is early, on time or late. If the train is early a minus (-) sign will be placed in front of the difference and if the train is late a plus (+) sign will be placed in front.

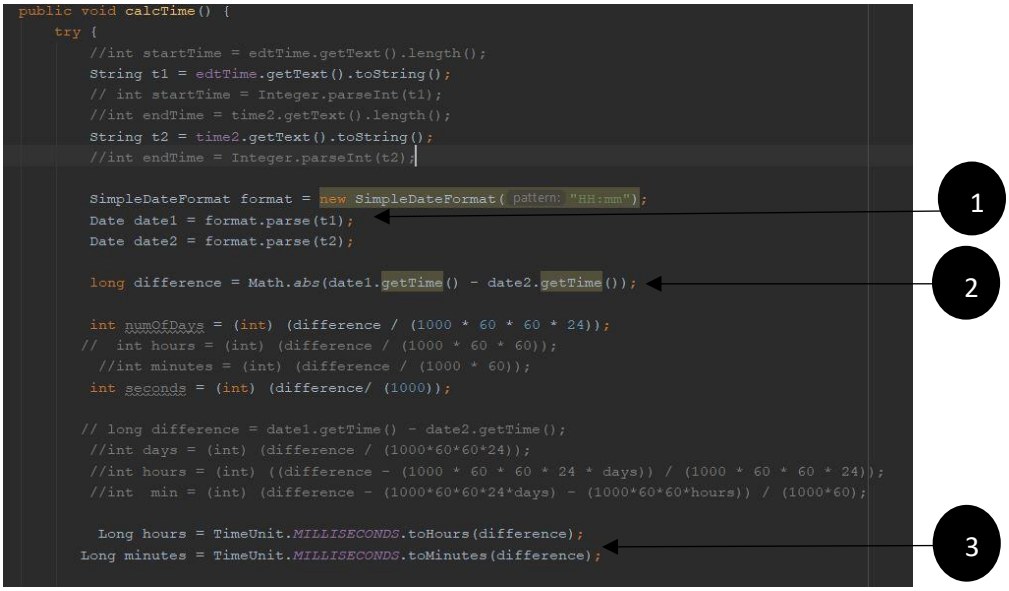


Figure 102 Calculate the difference

1. Convert String into date format
2. Formula to calculate the difference.
3. Convert the difference.

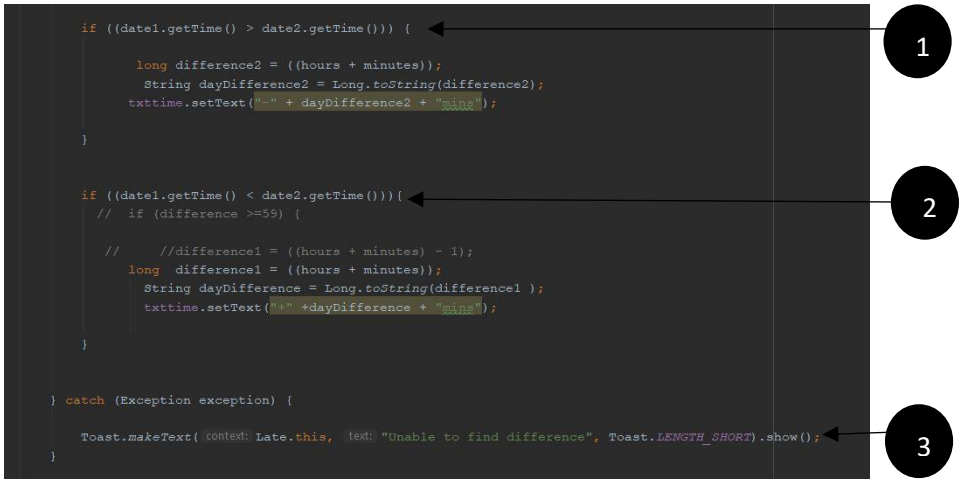


Figure 103 Compare Time

1. Compare both time if schedule time is greater than actual time.
2. Compare both time if schedule time is less than actual time.
3. If the calculation does not work then display the following message.

CHAPTER 8

TESTING

Testing is a vital investigation conducted use to validate and verify the system. Testing is done on each stage of the software development to identify the bugs in the systems and correct them. The objective of the testing phase is shown below:

- To spot bugs during implementation of the system.
- To check if the systems meets up all the functional requirements.

8.1 Types of Testing Used

Unit Testing

In unit testing, individual units of the software are tested to see if that part performed and function as designed. At each stage of the implementation unit test is done to see if its working as predicted.

Advantage

- Easy to detect bugs and correct them.
- Debugging is easy.

Incremental Integration Testing

In Incremental testing, small units are integrated together and continuous testing is being done as new functionality is added in the application.

Advantage

- Easy to detect the bugs in a smaller assembly

Integration Testing

The combine parts of the application are tested if they interact correctly together and to expose faults during the interaction between integrated units.

Regression Testing

It is the re-testing of the software after any modification or fixes to ensure it still performs correctly. Many re-testing has been carried out every time changes were made to the system.

8.2 Test Cases

User registration is one of the main proceeding step in this system. If the user is not registered, the latter won't be allowed to search for a train and send queries to admin and the admin won't be able to add new train schedule. The user should provide a valid email address and a password to register in the system.

Test No.	Test Title	Test Description	Expected Results	Pass/Fail
1.	Registration	a) Check for empty field	To display a message to user that the field is empty	Pass (Working as expected)
		b) Check for required validation	To display a message to user guiding the latter what the field should contains	Pass (Working as expected)
		c) Check for duplicate entry in the system	To display a message, it already exists	Pass (Working as expected)
2.	Confirm Email	Check if email is valid and exist	Send an email for confirmation to	Pass (Working as expected)

			be able to login with the email	
3.	Forget Password	Enter email to reset password	A mail is obtained to reset the password	Pass (Working as expected)
4.	Login	Invalid Credential	Prevent user from accessing the home page	Pass (Working as expected)
5.	Add Train	Create new train number (new train is bought)	Create new train using proper detail	Pass (Working as expected)
6.	Edit Train	Admin can change existing train	Change description about the train	Pass (Working as expected)
7.	Delete Train	Admin can delete a train	A message is displayed to inform that the train is deleted	Pass (Working as expected)
8.	Search Train	Admin can search for a specific train number	The train detail is displayed	Pass (Working as expected)
9.	Add Schedule	Create new schedule	Create new schedule adding proper date, time, station name, platform number and train number	Pass (Working as expected)
10.	Edit Schedule	Admin can edit existing schedule	Change the time of the train	Pass (Working as expected)

11.	Delete Schedule	Admin can delete a train	A message is displayed to inform that the train is deleted	Pass (Working as expected)
12.	Search by date	User can search a train by date	The train detail is displayed according to date	Pass (Working as expected)
13.	Search by Station Name	User can search a train by filtering Station Name	The train detail is displayed according to the selected Station Name	Pass (Working as expected)
14.	Search by Platform Number	User can search a train by filtering Platform Number	The train detail is displayed according to the selected Platform Number	Pass (Working as expected)
15.	Search by Train Number	User can search a train by filtering train number	The train detail is displayed according to the selected Train Number	Pass (Working as expected)
16.	Reply client queries	Admin can reply to client queries by send them a mail	The client will get a mail	Pass (Working as expected)
17.	Reply to driver notification	Admin can reply to driver notification by send them a mail	The driver will get a mail	Pass (Working as expected)

18.	Edit Profile	User can edit person detail and email	The user will get a mail confirming that the latter has changed his email	Pass (Working as expected)
19.	Edit Password	User can edit password	The password will be changed, the user won't be able to login with the old password	Pass (Working as expected)
20.	Logout	User can logout	The session will end up when logout	Pass (Working as expected)
21.	Delete Account	User can delete his account	The account is deleted	Pass (Working as expected)
22.	Send query to admin	Client can send query to admin	Admin get the query	Pass (Working as expected)
23.	Journey price	User can check journey cost	The price will be displayed	Pass (Working as expected)
24.	Send notification to admin	Driver can send notification to driver informing about train latency and reason	The admin will get a notification	Pass (Working as expected)
25.	Station display	The arrival time of train for the current date and time	Train list will be displayed according to the actual date and time	Pass (Working as expected)

Table 4 Test Cases

8.2.1 Test Case Registration

a) During registration the application will check if a field is empty and then an error message will be display telling the user to fill in the fields.

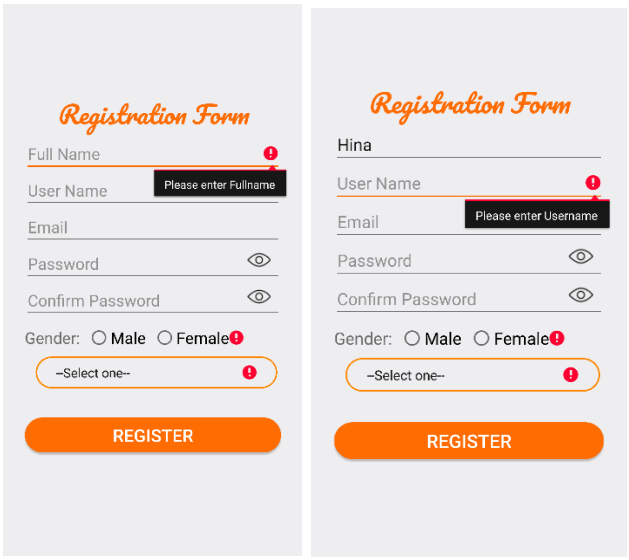


Figure 104 Test to very empty field

b) Each field has validation, if the user has wrongly entered something in a field an error message will be display guiding the user to fill the field correctly.

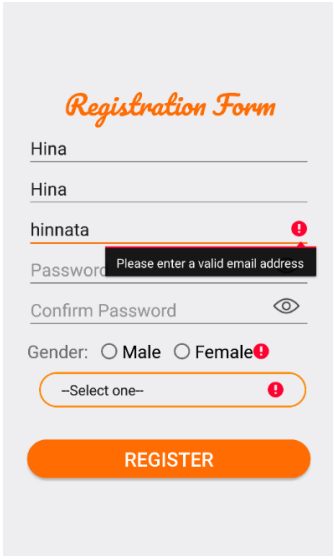


Figure 105 Test to check validation

c) The system will check if there is another account using the same email, if the email has already been registered a message will be display “Already exist”.

8.2.2 Test Case for confirm Email

A mail is send to the user to verify the email enter during registration and accept the mail to be able to login with the email. This a kind of security to confirm if it’s really the user who has register in the system.

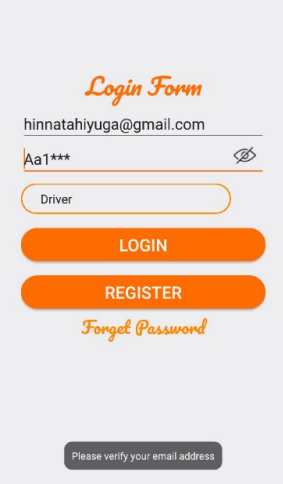


Figure 106 Test to verify email address

Figure 107 Verification email

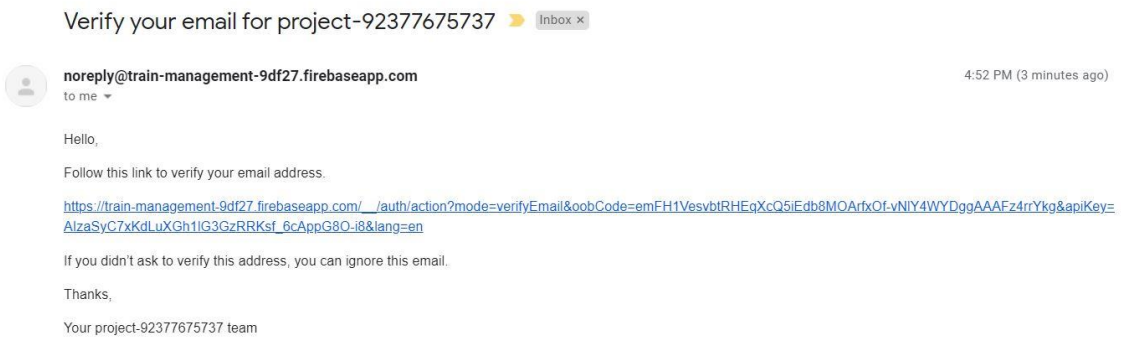


Figure 108 Link to verify email

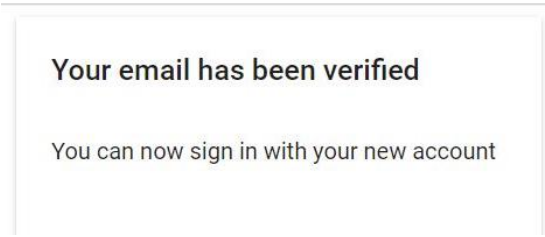


Figure 109 Email has been verified

8.2.3 Test Case for Forget Password

When a user forgets her/his password, the user can click on forget password and enter his/her registered email address. Hence a mail will be send to the user after that, the user has to click on the link and add a new password to reset his/her password. The user has a time interval of 2 days to reset his/her password else the mail token will be expired.

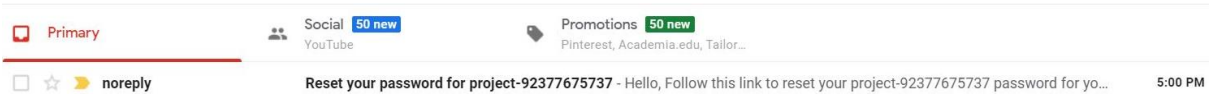


Figure 110 Email to reset password

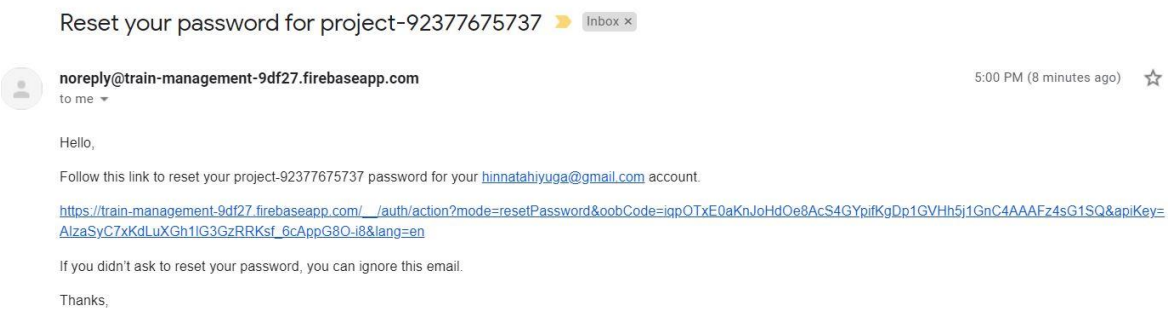


Figure 111 Link to reset password

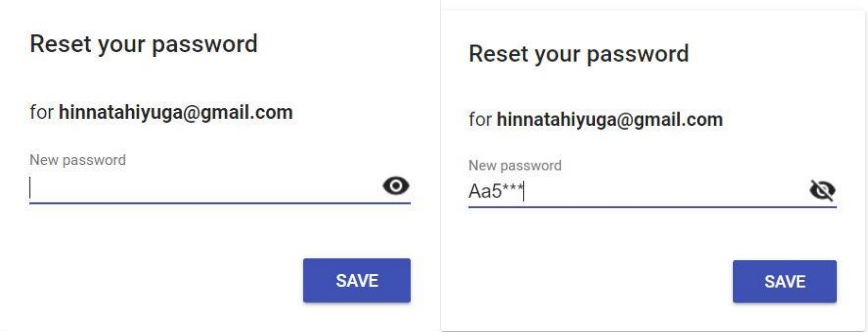


Figure 112 Reset Password

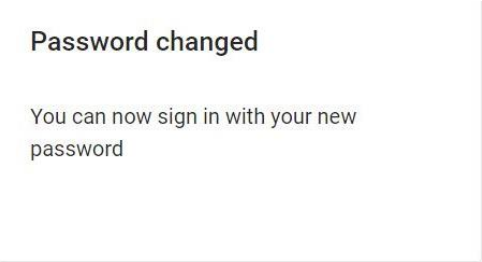


Figure 113 Password has been reset

8.2.4 Test Case for Login

If the user has wrongly entered his/her credential, a message will display saying “login fail”.

8.2.5 Test Case to Add New Train

New Train are entered in the database whenever a station got a new train. The train is assigned with a train number and platform number and at which station it will operate. When a new train is successfully created and stored in database a message is displayed “Successfully created”.

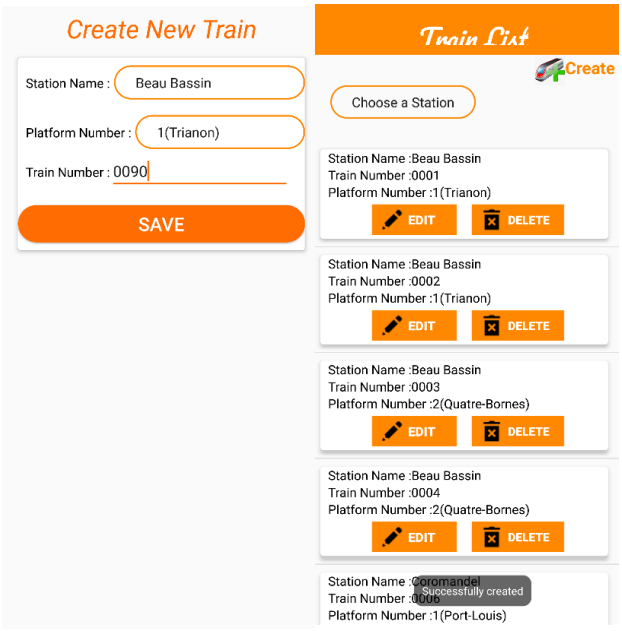


Figure 114 Create new train

8.2.6 Test Case to Edit a train

The admin need to select a train which he/she want to edit. The Id for that train and its details will be send to the editing page. After the details has been successfully updated a message will be displayed “Successfully update”.

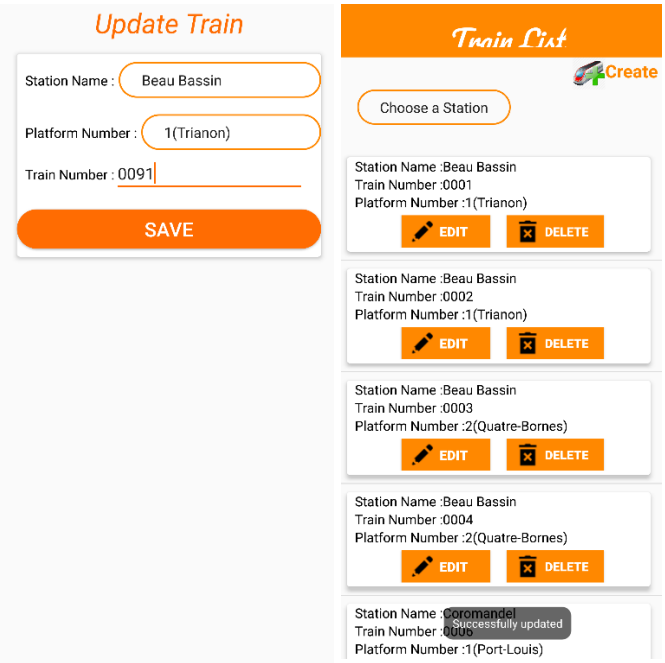


Figure 115 Update train

8.2.7 Test Case to Search a Train

An admin can check if a train has successfully entered or to edit a train by searching the train number. The latter will enter the train number in the edit text and click on search to search for that specific train.

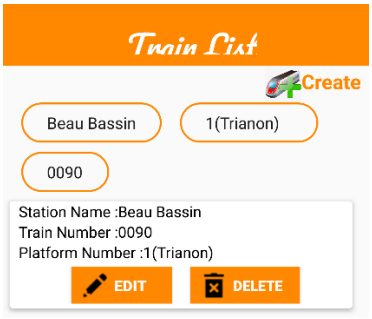


Figure 116 Search for a train

8.2.8 Test Case to Delete a Train

A train can be deleted by the admin if the train is no longer operational. After deleting the train, a message will be displayed "Successfully Delete".

8.2.9 Test Case to Add Schedule

This part is the main purpose of the project as according to the adding schedule the client will be able to track their train. In this part the admin adds the departure station, destination station, train number, platform number, date and arrival time the train will reach to its destination. When the admin has already entered all the information, a message will be displayed saying "Successfully Created".

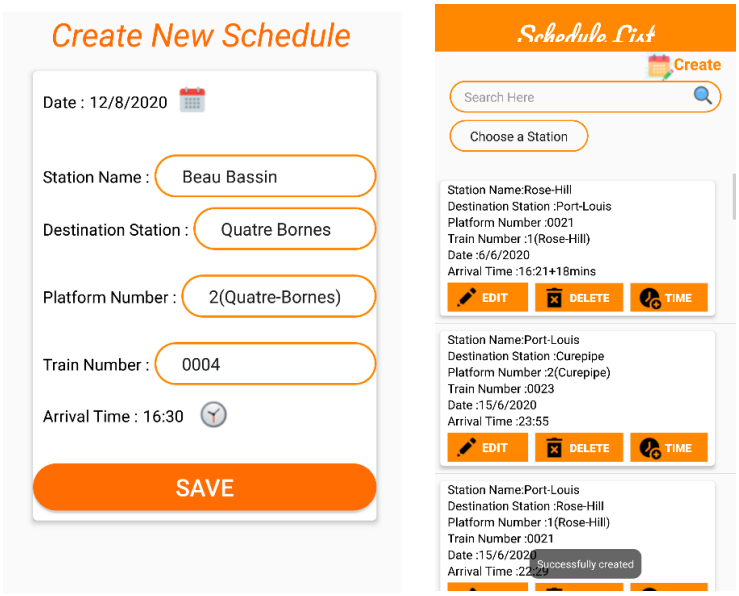


Figure 117 Create new schedule

8.2.10 Test Case to Edit Schedule

The admin can edit the schedule time if the arrival time has changed due to some reasons. After editing the information will automatically refresh and update in the client application. A message is displayed “Successfully update”, each time the admin updates a schedule.

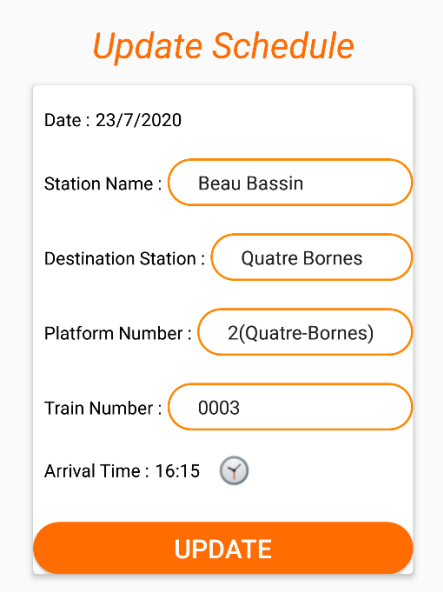


Figure 118 Update Reference

8.2.11 Test Case to Delete Schedule

The schedule can be deleted by the admin.

8.2.12 Test Case to Search by date

Both in the client and admin the user can search a train by its date. The user has to enter the date he/she want and click on search button. The list of train for that specific date will be displayed.

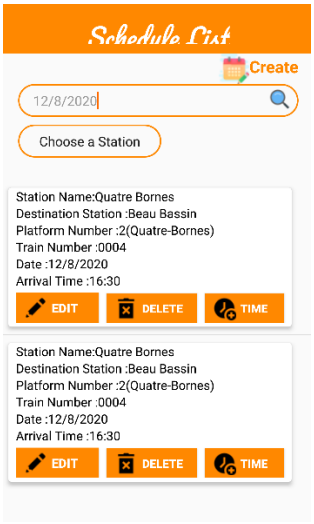


Figure 119 Search by date

8.2.13 Test Case to Search by Station Name, Platform Number and Train Number

In this part the both the admin and client are able to search for a train according to the station name, platform number and train number. The user will be able to filter for a specific train. At first there will be one spinner telling them to choose one Station Name, after selecting the station name the user want the second spinner will be visible to the user indicating them to choose a platform number. The platform number will be display according to the station name selected. After selecting a platform number, the available train number for that platform will be displayed in the third spinner. Hence on selecting the train number, the train list following these specific data will be display from the database.

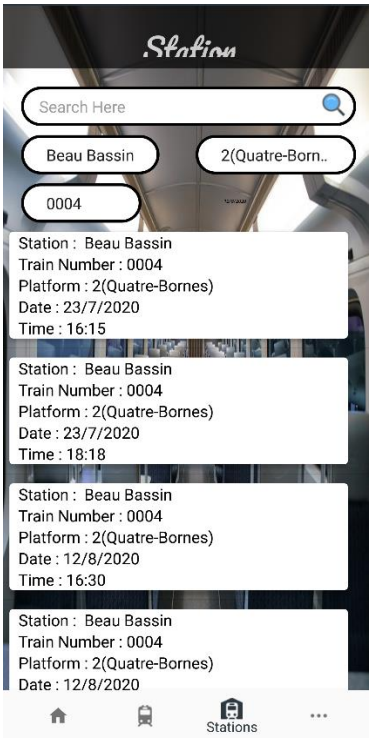


Figure 120 Search by station name, platform number and train number

8.2.14 Test Case to Search by Date, Station Name, Platform Number and Train Number

Both the admin and client are able to search for a train according to date, the station name, platform number and train number. The user will be able to filter for a specific train. The user will have to inset the date in the edit text and will have to choose one Station Name as mention above, after selecting the station name the user will have to choose a platform number. The platform number will be display according to the station name selected and train number will be displayed depending on the platform number selected. Furthermore, after selecting train number the train list will be displayed according to the date and the information selected.

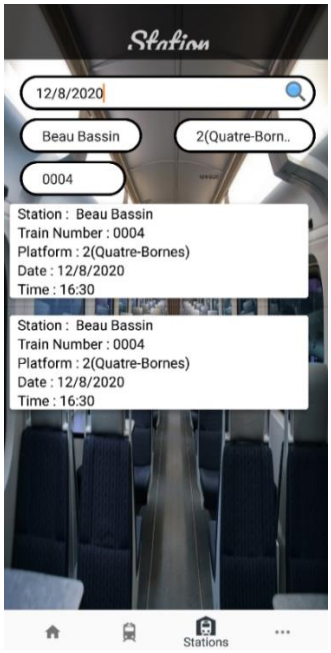


Figure 121 Search by date, station name, platform number and train number

8.2.15 Test Case Reply to Client Queries

The admin is able to answer the client queries by sending them an email. After clicking on the button send on the form, the client received a mail from the admin.

Figure 122 Reply to queries

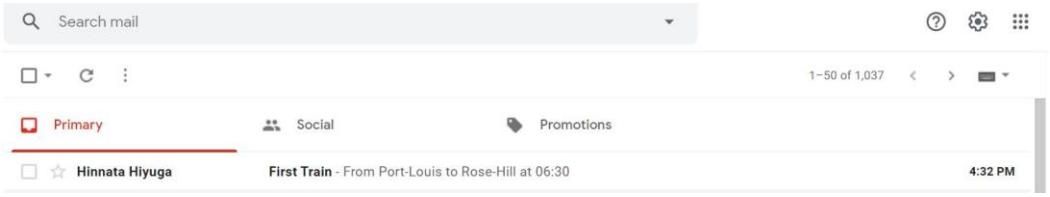


Figure 123 Mail send successfully

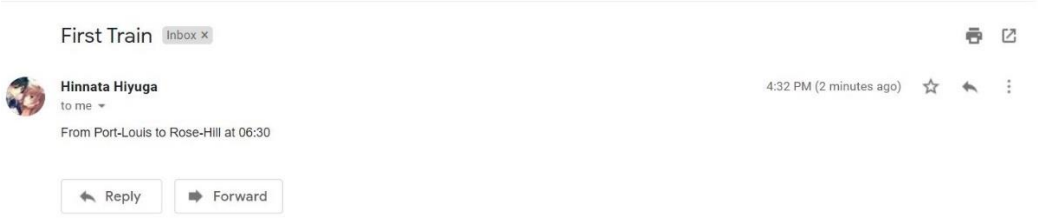


Figure 124 Query mail

8.2.16 Test Case Reply to Driver Notification

The admin is able to answer to the driver notification to change any time issues in the schedule. The procedure works the same as mention above for client.

8.2.17 Test Case to Edit Profile

In this part the admin, client and driver can edit their profile. After editing the email address, the user will get an email to his/her registered email informing and asking if they had change the email address and stating the new email address.

Figure 125 Email informing about changes made

8.2.18 Test Case to Edit Password

The user can edit his/her password also. The use just need to click on change password. After that the user need to add his/her old password, then enter the new password. Hence the user won't be able to login with his/her old password a message will be displayed "Login Fail". The latter has to enter the new password to be able to login.

8.2.19 Test Case to Logout

Moreover, the user can logout from the application after he/she had used it. The user session will be destroyed he/she won't be able to go back if they clicked the back button, they will have to login to re-use the application.

8.2.20 Test Case to Delete Account

The user is able to delete his/her account. The account will be deleted, if ever the latter login with his deleted account email and password he/she won't get any access, he/she will have to register for a new account.

8.2.21 Test Case Send Query to Admin

A map is available to the client where there is all the location of the station, the user just need to click on any station. A query page will be displayed according to the station selected, the user will be able to select the platform number, train number and their queries from spinners. After clicking on send, the queries will be stored in the database and then it will be displayed on the admin application where the latter would be able to view the query.

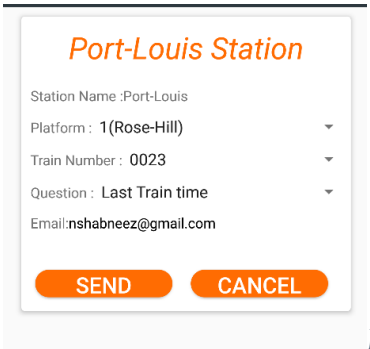


Figure 126 Query

8.2.22 Test Case for Journey Price

Furthermore, the user will be able to calculate his journey price. The price is categories in three type the adult, child and student. The prices differ for each category and journey. An adult can view the price for his/her child.

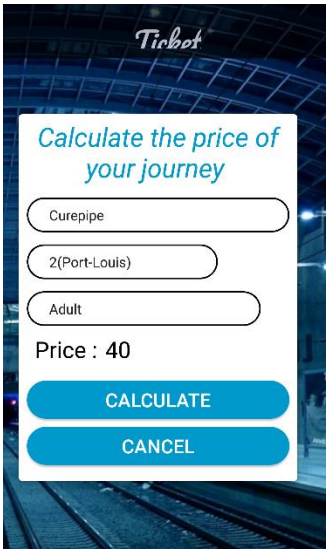


Figure 127 Journey Price

8.2.23 Test Case Send Notification to Admin.

This process is the same as for the client query. A map is available where the driver has to clicked on any station and inform about any latency by how many minutes and the reason, so that the admin will be able to change the arrival time of the train accordingly to inform the client.

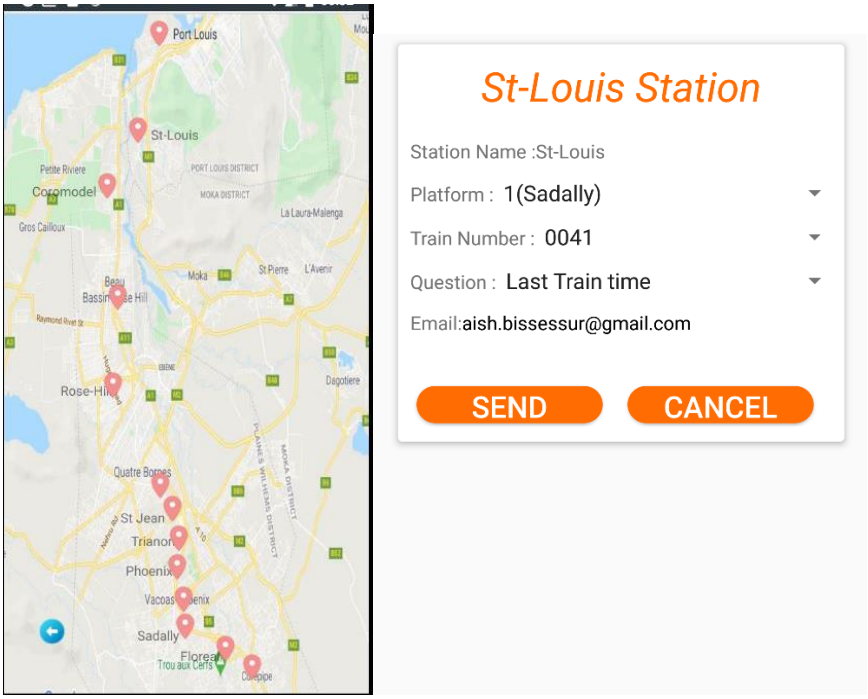


Figure 128 Test notification

8.2.24 Test Case to Display Arrival Train on Station

Each station consists of a screen, where the passenger is able to view the arrival time of trains for the current date and time for a particular station. The page is refresh each one minute to keep the information display up-to-date.

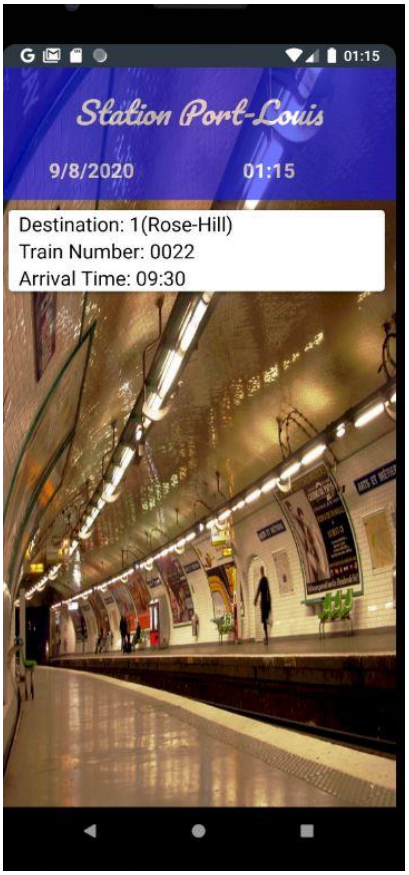


Figure 129 Display Station

8.2.25 Test Case for Sensors

When the RFID reader read the sensor, it displays the train number assign to this tag/card on the LCD screen. Thus, the actual arrival time and the train number is save in the database.

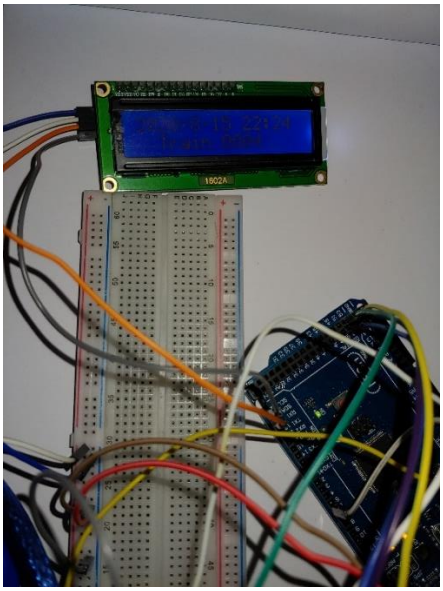


Figure 130 Train number and actual time

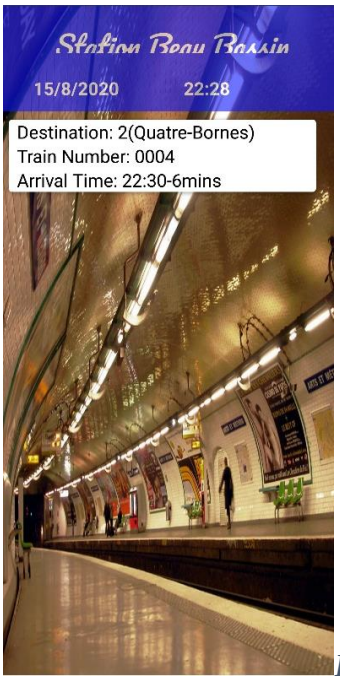


Figure 131 Displaying the -6 mins to show the train is early

CHAPTER 9

CONCLUSION AND FUTURE WORKS

9.1 Evaluation of Application Requirements

Req. No	Requirements	Completed ☑	Not Completed ☒
1.	The system should allow the user to register.	✓	
2.	The system shall send a verification email to confirm the email register to be able login.	✓	
3.	The system shall send an email to reset password.	✓	
4.	The user shall be able to login after verification.	✓	
5.	The system should allow the user to add new train.	✓	
6.	The system should allow the user to edit a train detail.	✓	
7.	The system should allow the user to delete a train.	✓	
8.	The system should allow the user to search a train.	✓	
9.	The system should allow the user to create new schedule.	✓	
10.	The system should allow the user to edit the time of the schedule.	✓	
11.	The system should allow the user to delete a schedule.	✓	

12.	The user shall be able to search by date.	✓	
13.	The user shall be able to search by station name.	✓	
14.	The user shall be able to search by platform number.	✓	
15.	The user shall be able to search by train number.	✓	
16.	The user shall be able to filter by date, station name, platform and train number.	✓	
17.	The user shall be able to send queries to admin.	✓	
18.	The admin shall be able to replies to the user queries.	✓	
19.	A map shall be displayed to allow user to find the nearest station to their location.	✓	
20.	The user shall be able to edit personal detail.	✓	
21.	The system shall send an email to verify if the user edited his email address.	✓	
22.	The user shall be able to edit his/her password.	✓	
23.	The user shall delete his/her account.	✓	
24.	The user shall be able to see the price for each journey.	✓	

25.	The system should be able to display the list of train according to the actual date and time by refreshing each minute.	✓	
26.	The mobile application was designed for all screen size.	✓	
27.	The system shall get all the information of all trains but will automatically select the data that refers to particular station and shows that information on screen.	✓	

Table 5 Completed Requirement

9.2 Evaluation of Arduino (Sensor) requirements

Req. No	Requirements	Completed ☑	Not Completed ☒
1.	Should read the actual date and time.	✓	
2.	RFID reader shall read the sensor.	✓	
3.	The system should record the actual date and time when the sensor is read.	✓	
4.	The system shall display the actual time and date the sensor has been read on a LCD.	✓	
5.	The system shall be able to differentiate between the time the sensor has been read and the time predicted in the schedule. If the actual time is less than the predicted time it shall display "- min". And if the actual time is greater than the	✓	

	predicted time it shall display "+ min".		

Table 6 Arduino requirements completed

9.3 Limitation

- The application was created for android user only.

9.4 Conclusion

In this thesis, we addressed the issues of passengers are experiencing with the arrival time of train. One of the main contribution of this project was to re-engineer the existing system in order to improve and track the actual arrival time of train. A mobile application has been created to help the passenger to track the arrival time of their trains whether it will be early, on time or late. The application is connected with a sensor, the RFID reader read the train ID whenever the train passed through it and the actual date and time is recorded. Furthermore, it helps the passengers to see the arrival time of train through their application.

Moreover, this project has taught me the importance of time management to be able to achieve our objective. Object-oriented programming concepts has been used to develop this application. I have acquired more skill in object-oriented while developing this project. It was a great experience to apply all the acknowledge and skills that I have acquired during the 3 years course of BSc Computer Science with Network Security.

9.5 Difficulties

The main difficulty with this project would be learning and connecting the Arduino parts togethers. The mounting where to connect the wires and how to use the libraries in Arduino IDE. Moreover, queries for firebase also was a bit difficult to implement search function.

9.6 Future Work

- Add booking system.
- Add a live chat system.
- Sign up with Facebook or Instagram account.

REFERENCE

[1]. Hierarchical Railway Traffic Model for Information Systems.

[2]. Train routing and timetabling via a genetic algorithm.

[3]. Research on Train Reception and Departure System Based on Intelligent Video Analysis (2016).

[4]. RT-Train: A Real-Time Tracking System for Public Train in Malaysia.

[5]. REAL TIME RAILWAY INDICATOR (2017.

[6]. Real Time GPS for Railway Automation System (2016).

[7]. https://www.independent.co.uk/travel/news-and-advice/train-delays-2018-cancelled-uk-rail-passengers-time-lost-waiting-a8921736.html [30 August 2019]

[8]. https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-RFID.html [20 September 2019]

[9].https://www.swann.com/blog/motion-security-sensors-explained/ [21September 2019]

[10]. https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-motion-Sensor.html [21 September 2019]

[11]. https://data-flair.training/blogs/java-tutorial/[21 September 2019]

[12]. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm[23 September 2019]

[13]. https://en.wikipedia.org/wiki/Firebase[10 October 2019]

[14]. https://firebase.google.com/products[10 October 2019]

[15].https://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard#:~:text=A%20breadboard%20is%20a%20rectangular,(light%2Demitting%20diode). [20 October 2019]

[16].http://softwaretestingfundamentals.com/unittesting/#:~:text=UNIT%20TESTING%20is%20a%20level,and%20usually%20a%20single%20output. [25 October 2019]

APPENDIX

APPENDIX A

ADMIN MANNUAL

Purpose

The purpose of this manual is to act as a guide to a user who will use the system.

Application Requirements

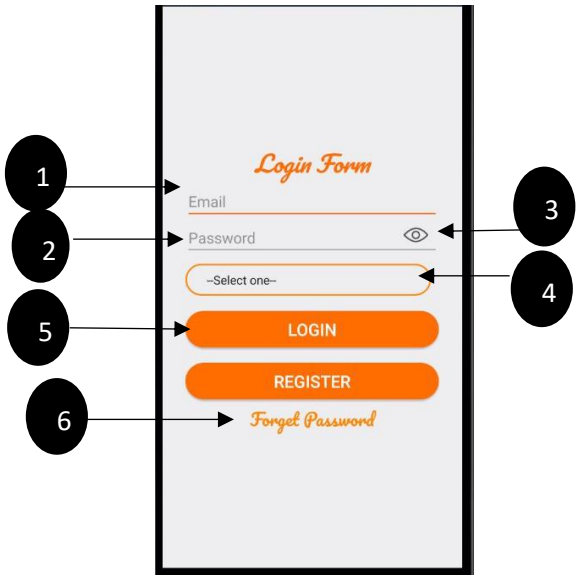
1. Android Mobile phone
2. Internet Connection

Getting Started

After installing the apk application on the mobile phone, clicked on the icon to open it.

Login

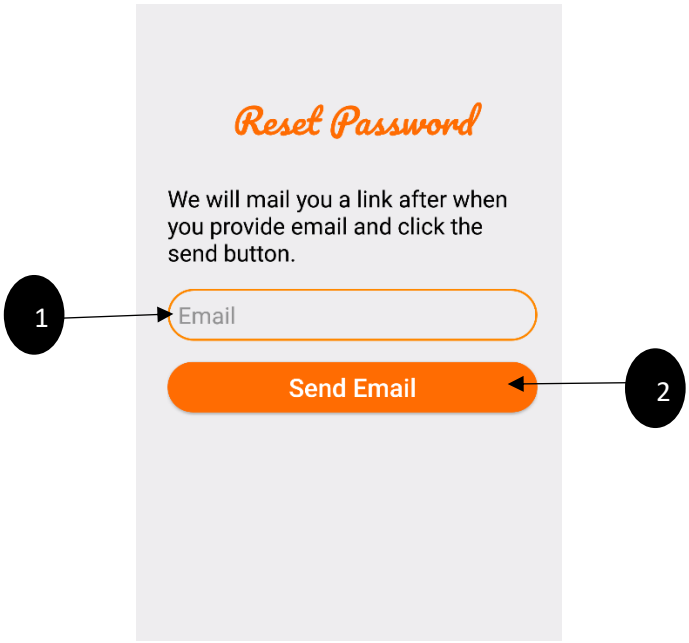
The admin has to login with his/her email and password.



1. Enter email address.
2. Enter password.
3. Click on button to show and hide password.
4. Select a user type “Admin”.
5. Click on login to get access to the home page.

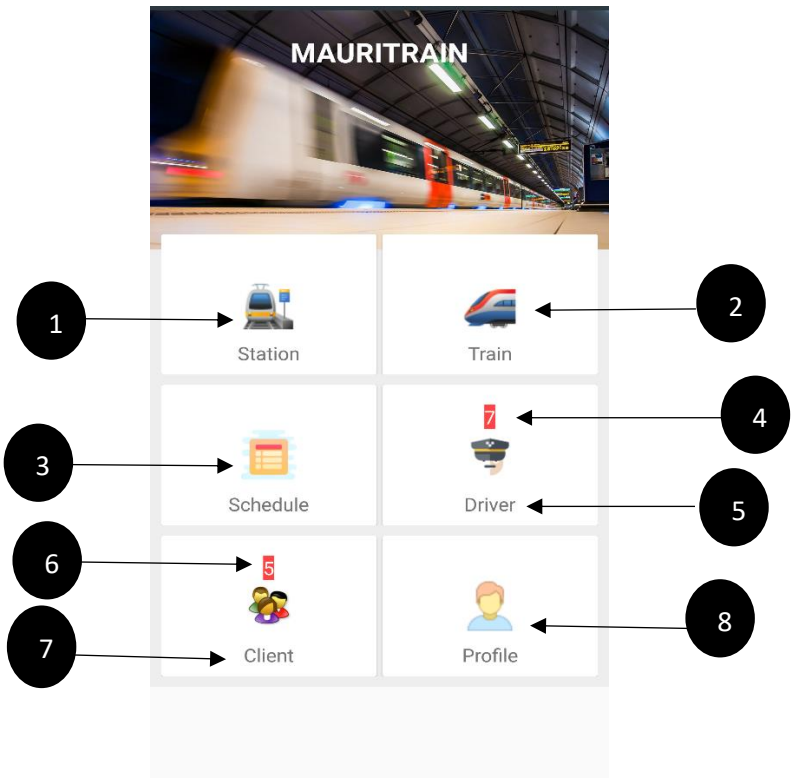
6. Click on forget password to reset password.

Forget Password



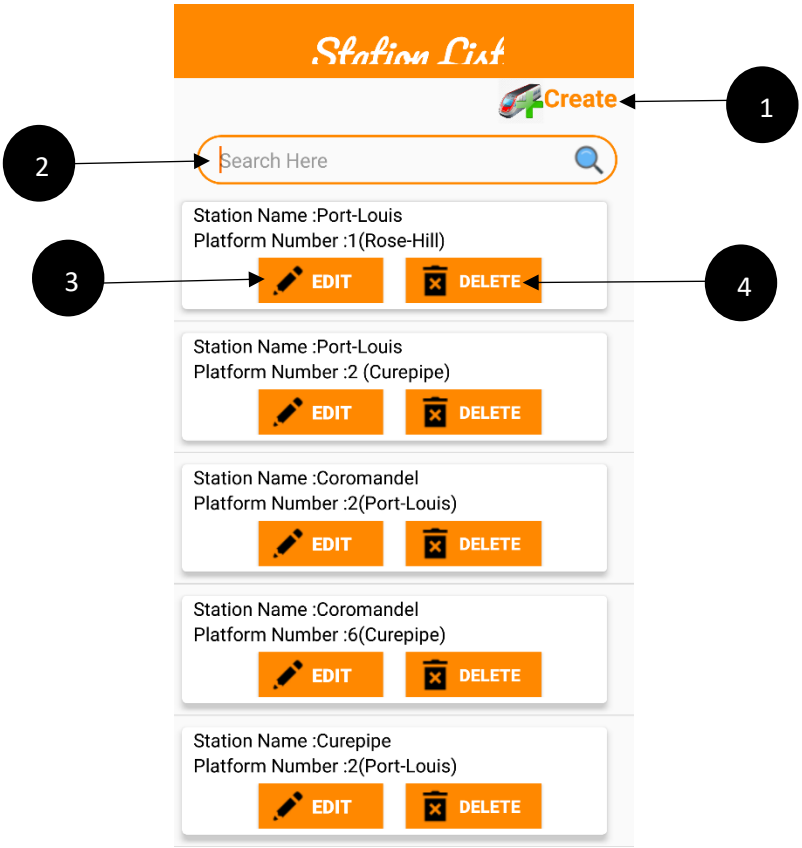
1. Enter email address.
2. Click on send to be able to reset password.

Main Menu



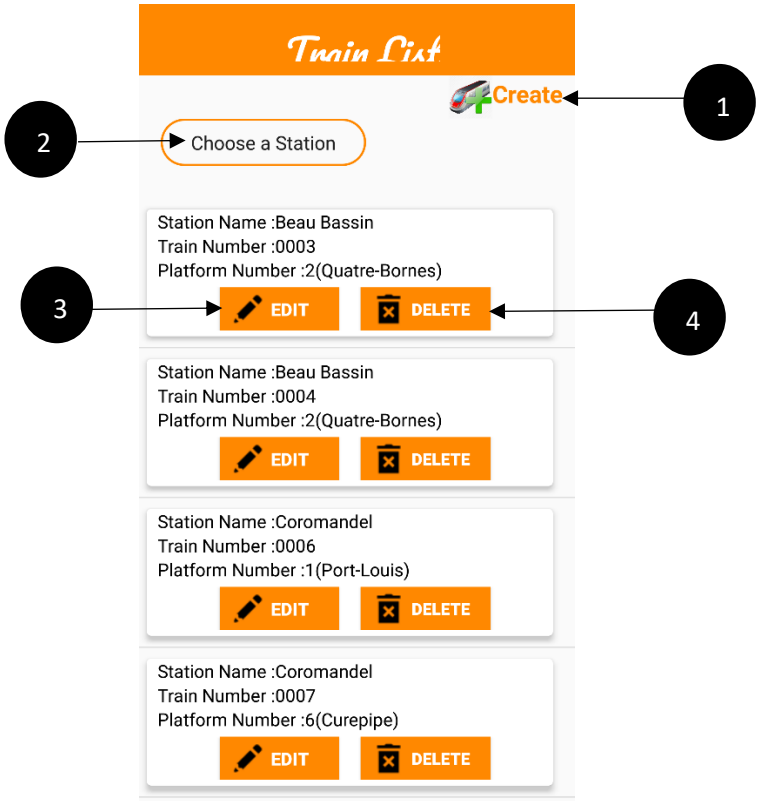
1. Click on Station to view list of station.
2. Click on Train to view list of Train.
3. Click on Schedule to view list of schedules.
4. Number to show how many notifications there are.
5. Click on Driver to view notifications.
6. Number to show how many queries there are.
7. Click on Client to view queries.
8. Click on Profile to view profile details.

Station Page



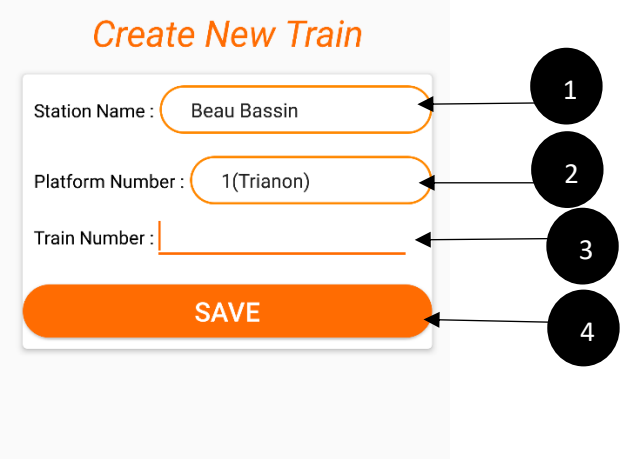
1. Create new platform number for any station.
2. Search for a station.
3. Edit any platform number.
4. Delete any station.

Train Page



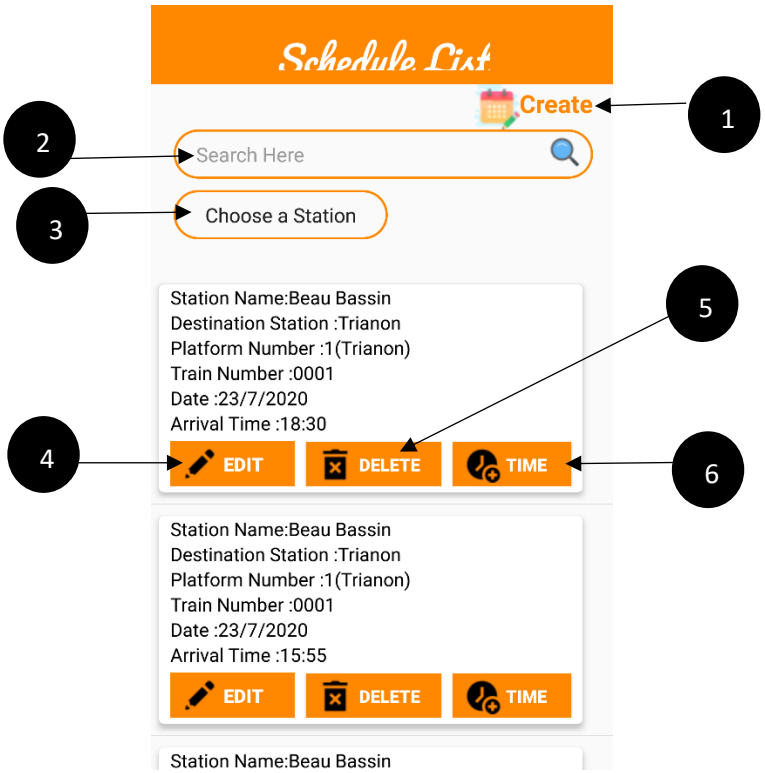
1. Create new train number if new trains have been bought.
2. Choose a Station from the spinner to search for a specific train according to its station name, platform number and train number.
3. Edit any train number.
4. Delete any train.

Create Train



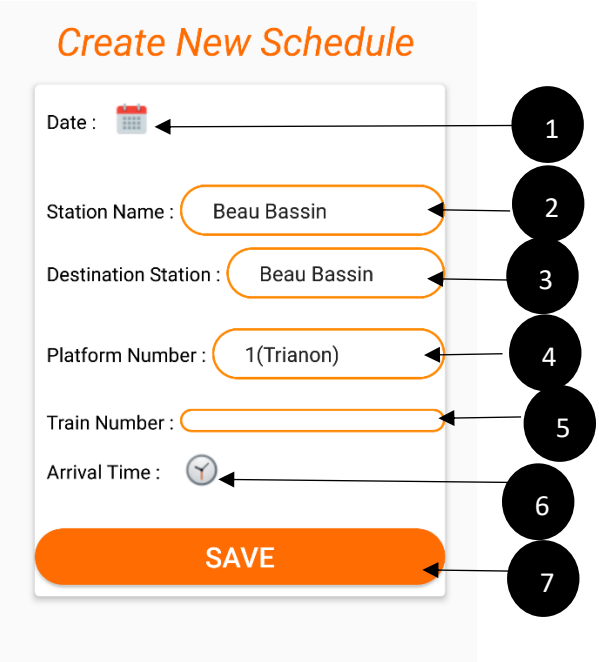
1. Select a station name.
2. Select a platform number.
3. Enter number to create a new train number.

Schedule page



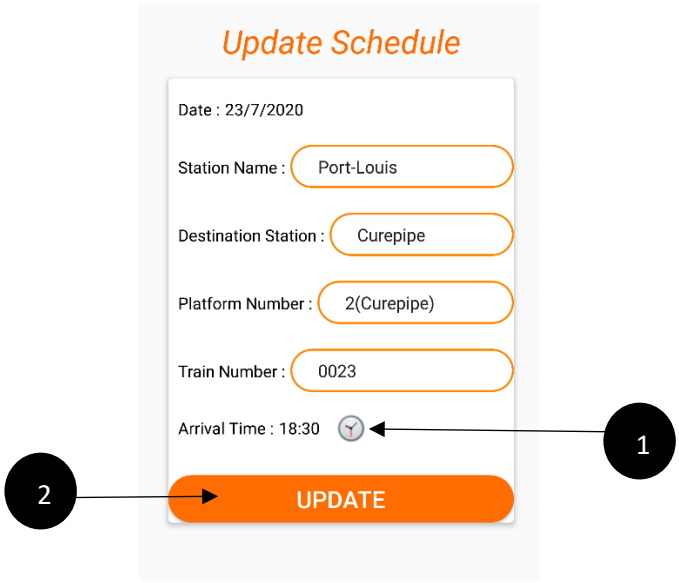
1. Create new schedule.
2. Search by date in format “dd/m/yyyy”.
3. Select a station name to search a schedule by filtering its station name platform number and train number.
4. Edit the schedule information.
5. Delete the schedule.
6. Edit the time for a schedule.

Create Schedule



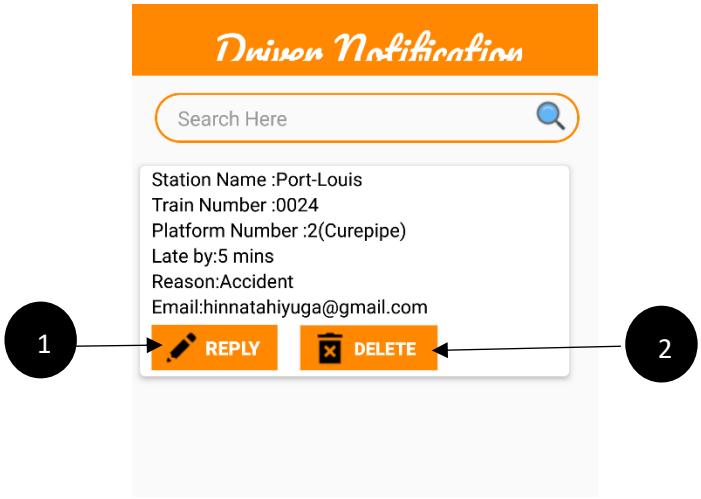
1. Click on calendar to add a date.
2. Select a station name.
3. Select a destination station name.
4. Select a platform number.
5. Select a train number.
6. Click on clock to add time.
7. Click on save to save the information.

Update Schedule



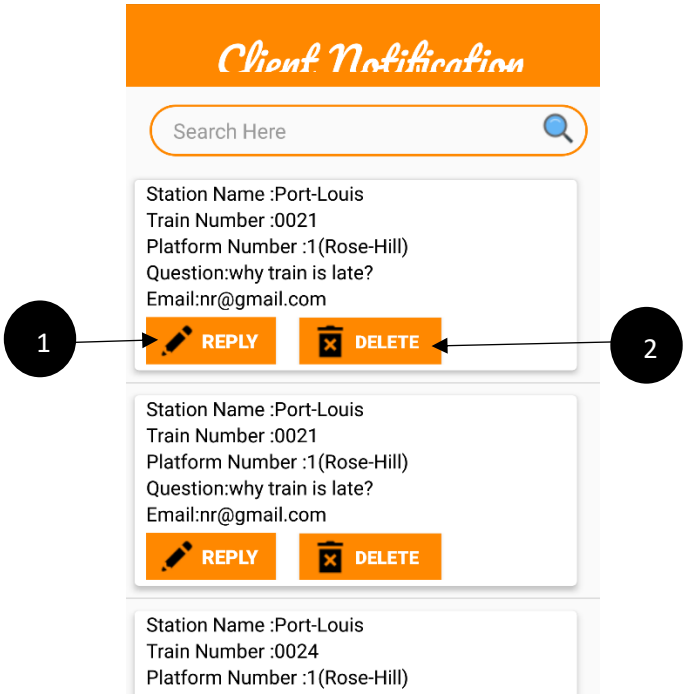
1. Click on the clock to edit the time.
2. Click on update to change the time.

Driver Notification



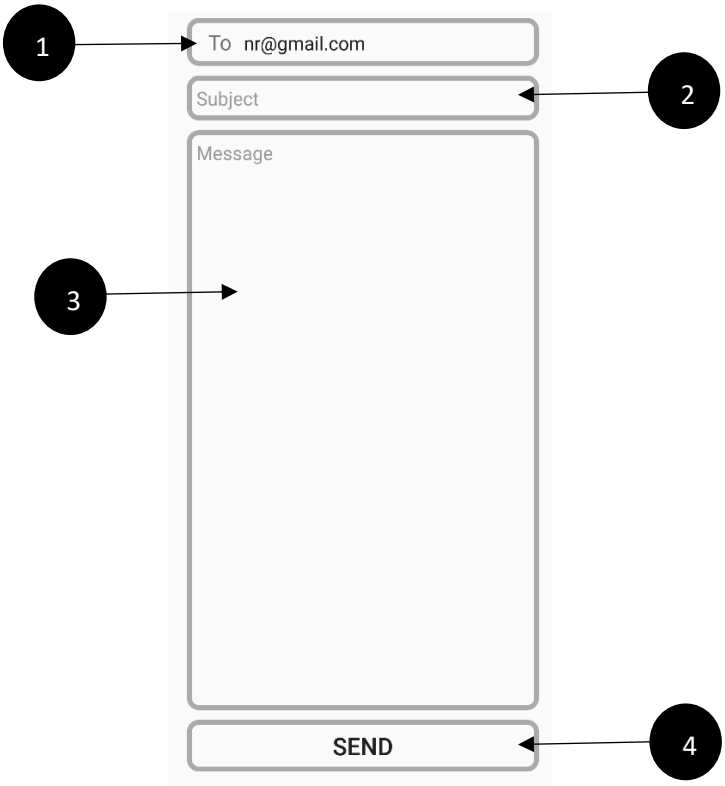
1. Admin can reply to driver.
2. After edit the time the admin can delete the notifications.

Client Query



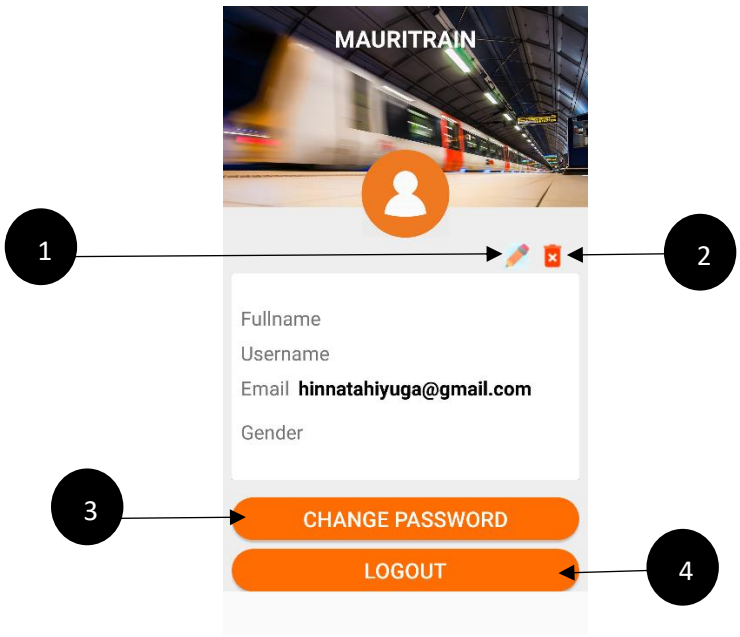
1. Admin reply to client query.
2. Admin can delete query after replying to them.

Reply to notification and query



1. The email of the client/driver will automatically be entered in the box To.
2. Subject- the title of the mail.
3. Message- reply to the query asks.
4. Send email to the client/driver.

Profile Detail



1. Click on it to edit personal detail.
2. Click on it to delete the account.
3. Click on it to change password.
4. Click on it to logout from the account.

USER MANUAL

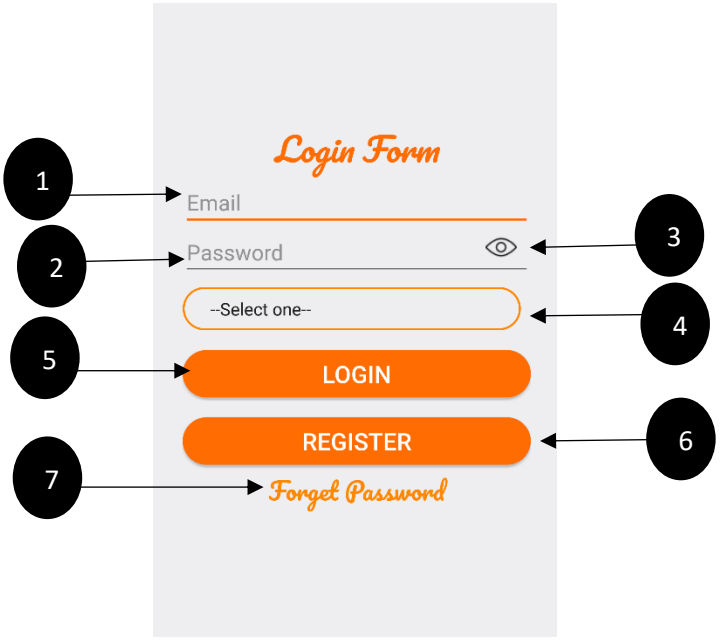
Application Requirements

1. Android Mobile phone
2. Internet Connection

Getting Started

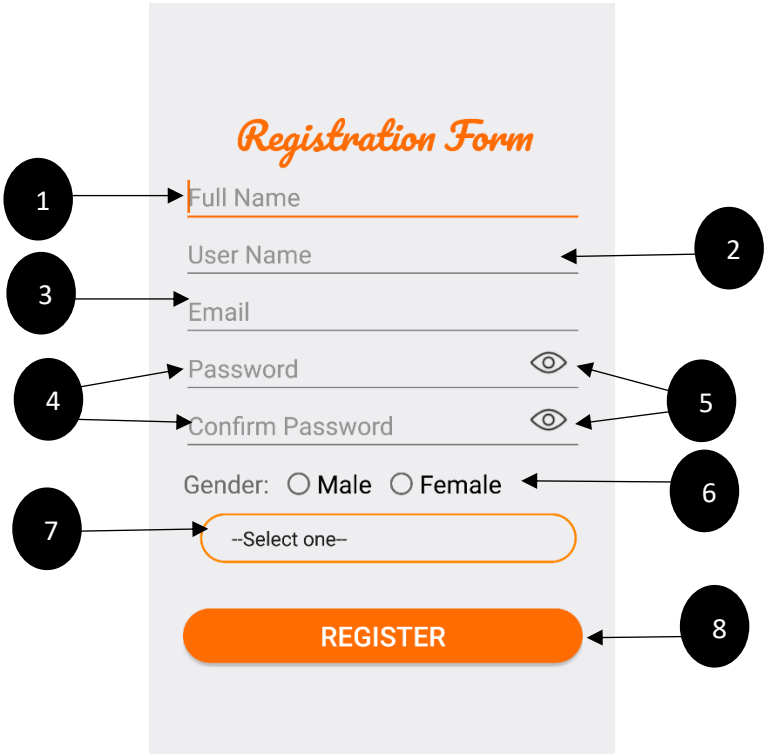
After installing the apk application on the mobile phone, clicked on the icon to open it.

Login



1. Enter registered email address.
2. Enter registered password.
3. Click to show and hide password.
4. Select user.
5. Click on register for registration.
6. Click on forget password to reset password.

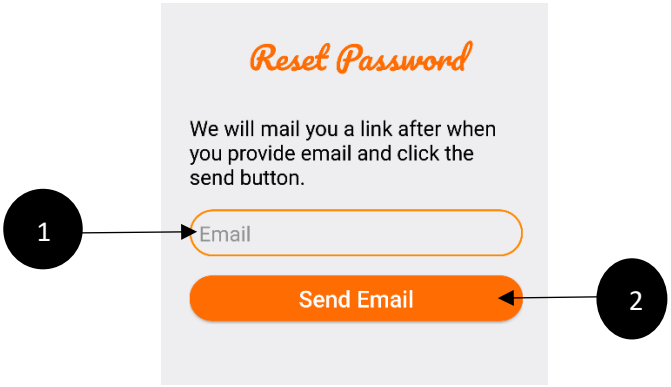
Registration



1. Enter full name.
2. Enter a user name.
3. Enter a valid email address.
4. Enter a password.
5. Click on it to show or hide password.
6. Select a gender.
7. Select user.
8. Click on register button to save the information.

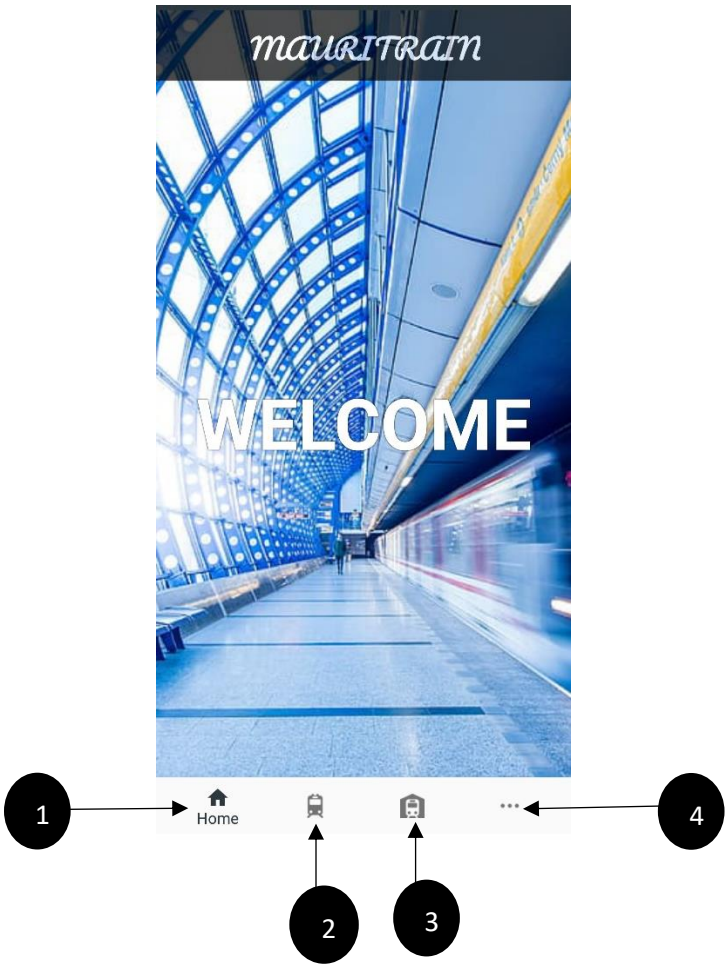
After the user has completed the registration, an email will be send to the user to verify his/her registered email address. Confirming the email, the latter will be able to login with his/her registered email.

Forget Password



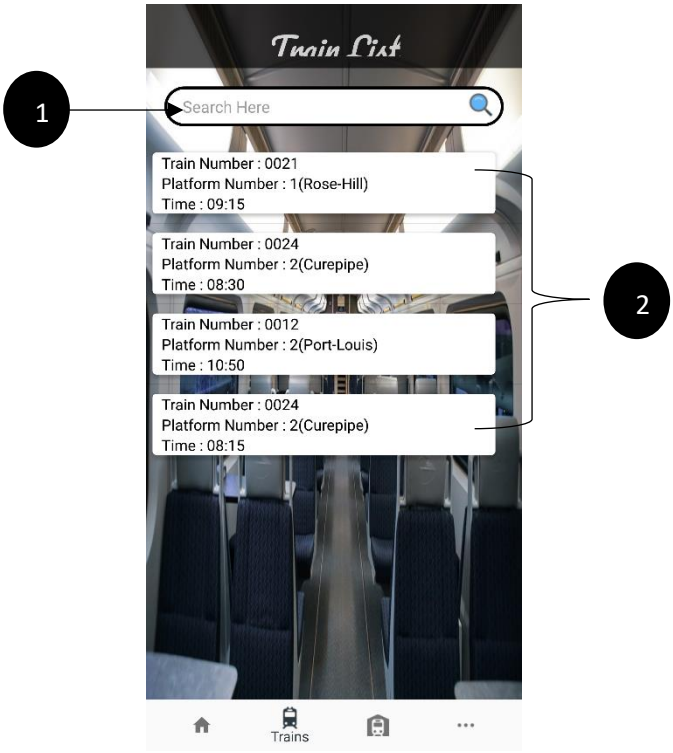
1. Enter registered email address.
2. Click on it to reset password.

Home Page



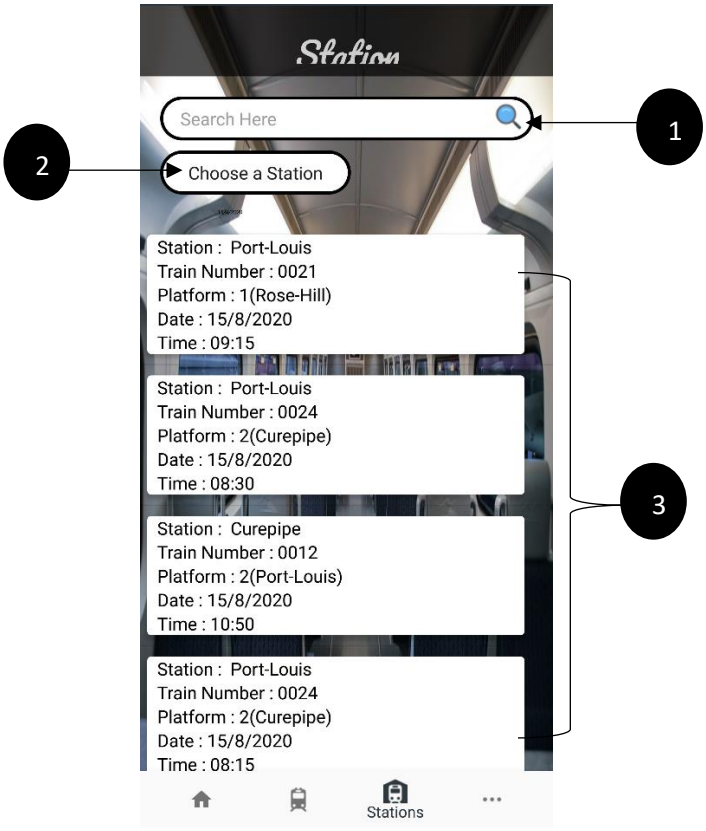
1. Click icon home to open home page.
2. Click on icon train to view list of train.
3. Click on icon station to view list of station.
4. Click on icon more to get more option.

Train Page



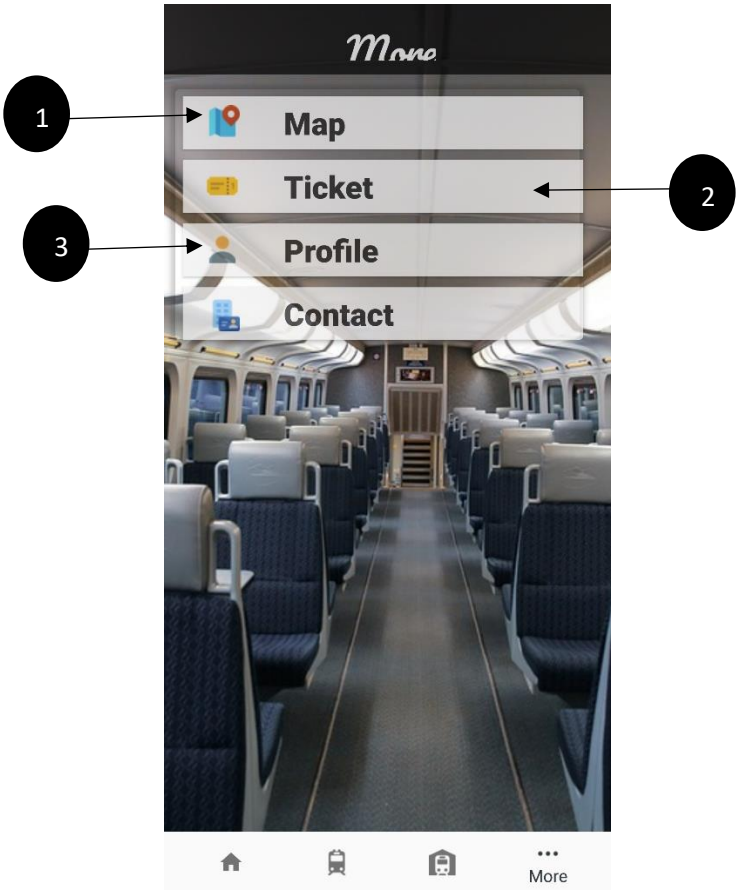
1. Search a train number/platform number.
2. List of trains for current date.

Station Page



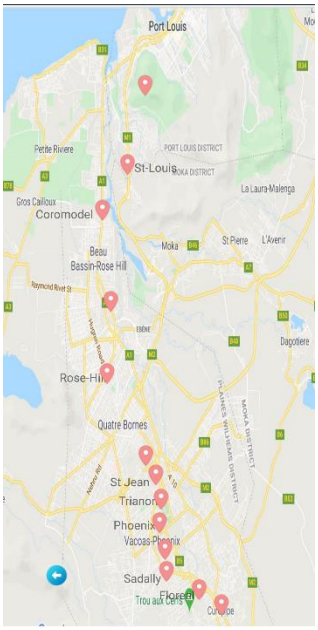
1. Search by date.
2. Choose a station name to search for a specific train by filtering its platform number and train number.
3. List of schedules for trains from departure station to destination station.

More Page



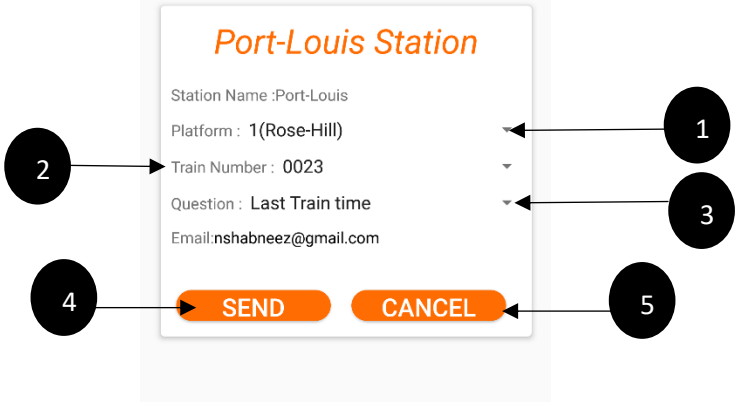
1. Click on map to view location of stations.
2. Click on ticket to view price of journey.
3. Click on profile to view personal detail.

Map



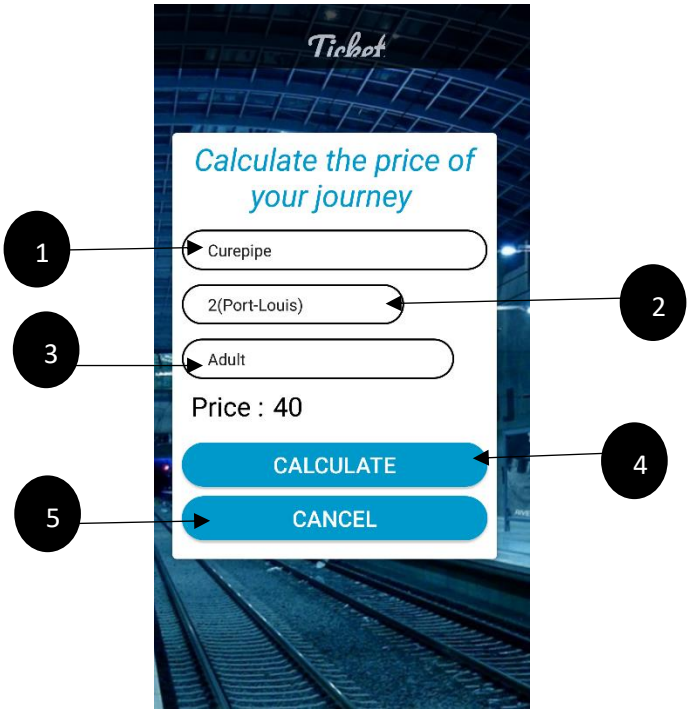
1. Location of station
    - Click on each location to be able to send any queries to the admin.

Query



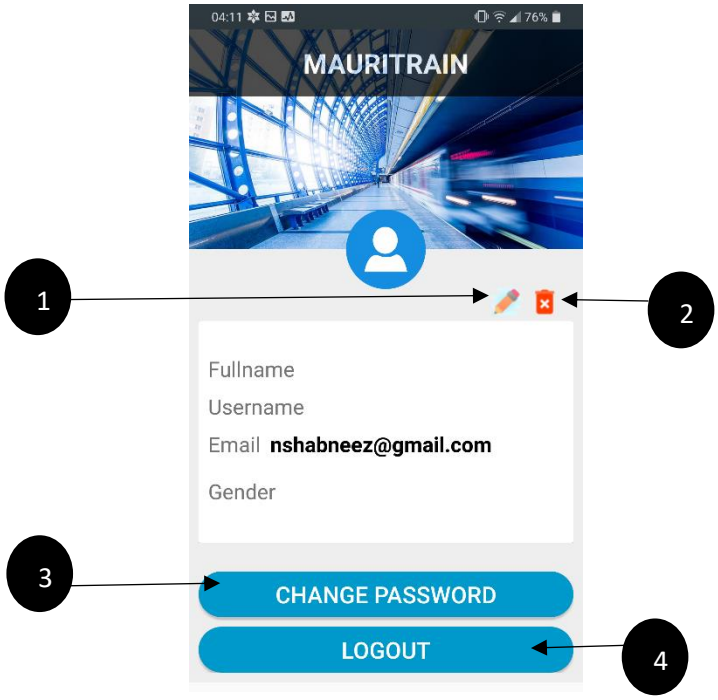
1. Select a platform destination.
2. Select a train number.
3. Select a query.
4. Send query to admin.
5. Cancel query and revert back to map.

Ticket



1. Select a station name.
2. Select a destination platform number.
3. Select a category.
4. Click to get the price of the journey.
5. Click to go back on home page.

Profile Detail



1. Click to edit personal detail.
2. Click to delete account.
3. Click to change password.
4. Click to logout from the application.

DRIVER MANUAL

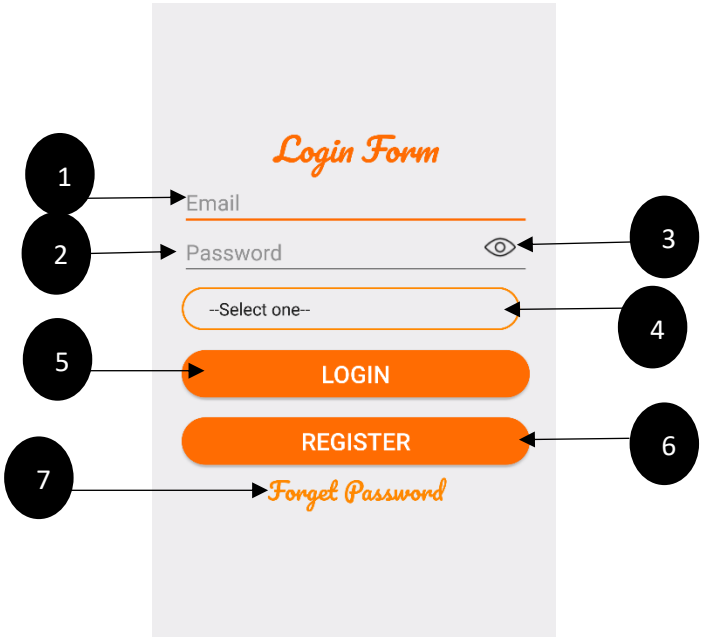
Application Requirements

1. Android Mobile phone
2. Internet Connection

Getting Started

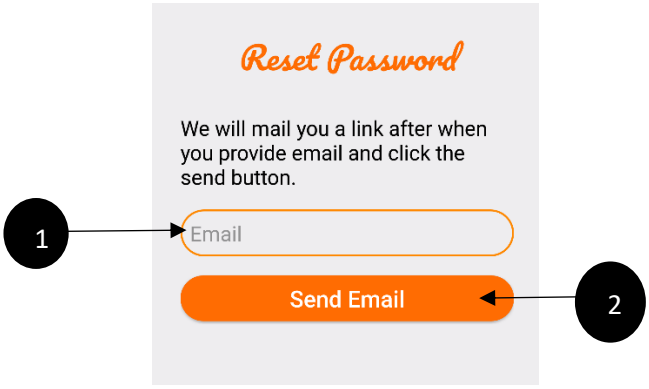
After installing the apk application on the mobile phone, clicked on the icon to open it.

Login



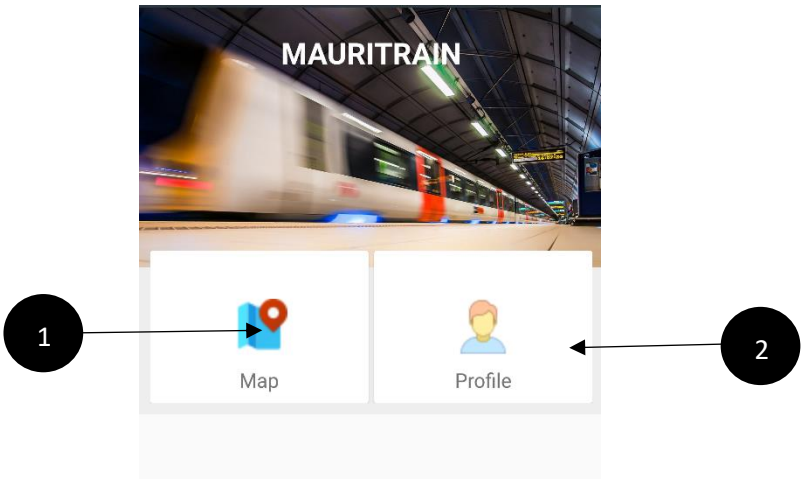
1. Enter registered email address.
2. Enter registered password.
3. Click to show and hide password.
4. Select user.
5. Click on register for registration.
6. Click on forget password to reset password.

Forget Password



1. Enter registered email address.
2. Click on it to reset password.

Main Menu

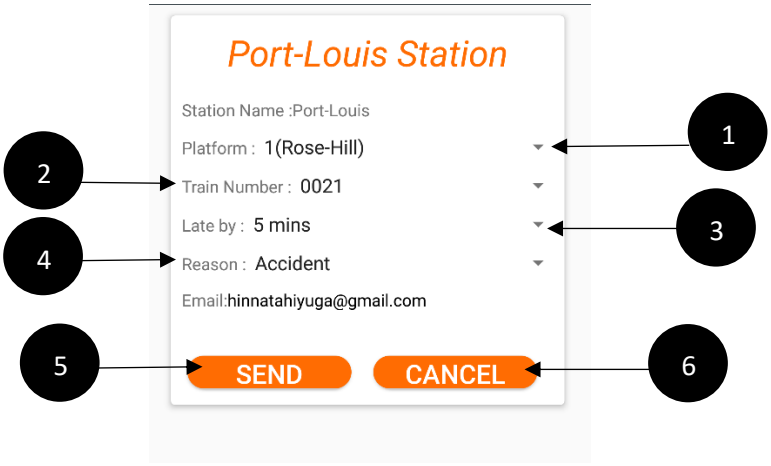


1. Click on the icon to view the location of stations.
2. Click on profile icon to view person detail.

Map

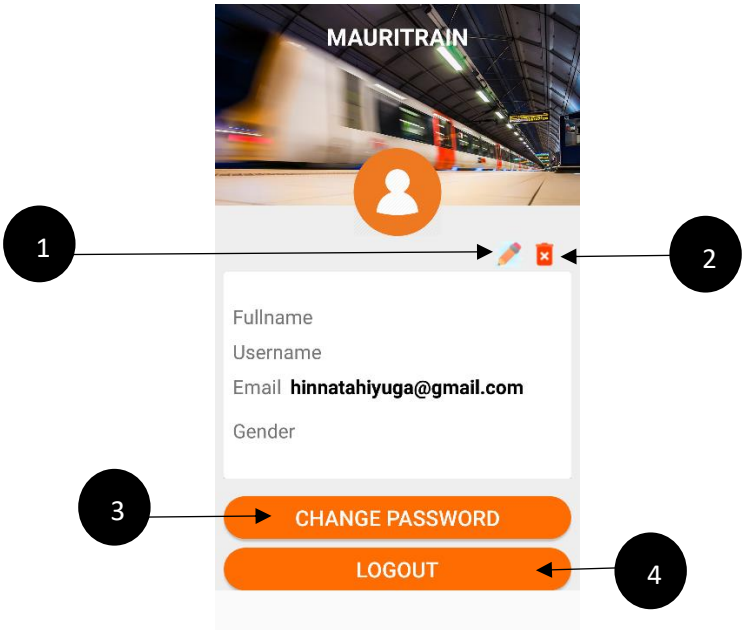
1. Location of stations.
    - Click on each station to be able to send notification to admin about lateness.

Send Notification



1. Select a platform number.
2. Select a train number.
3. Select a time by how much the train will be late.
4. Reason to be late.
5. Send notification to admin.
6. Cancel the notification and revert back to map.

Profile Detail



1. Click on it to edit personal detail.

2. Click on it to delete the account.
3. Click on it to change password.
4. Click on it to logout from the account.

UNIVERSITY of TECHNOLOGY, MAURITIUS

Annex A

SCHOOL OF INNOVATIVE TECHNOLOGIES AND ENGINEERING

Department of Industrial System and Engineering

Student Diary

1. Student's Name / ID: BISSESSUR Aïshani Prama / 170731

2. Programme Name: **BSc.(Hons) Computer Science with Network Security** Cohort: **BCNS17BFT.**

3. Supervisor's Name: Mr. TSe Kai Wai Dudley

4. Student Telephone No.: Residential/Office 6175152 Mobile: 5827 6184

5. Student Email Address: aish.bissessur@gmail.com

6. Date of Meeting: 29.08.2019

7. Comments

8. Actions Taken

Dicuss about the overall project

9. Action to be taken for next time

Write Aim, objective and problems of the project.

10. Date for next meeting: 16.09.2019

Student Signature: A.P.Bissessur Date: 29.08.19

Supervisor's Signature: Date: 29/8/19

Project Handbook BCNS17BFT Aug19_v2.0 – ISE Dissertation Committee 21

UNIVERSITY of TECHNOLOGY, MAURITIUS

Annex A

SCHOOL OF INNOVATIVE TECHNOLOGIES AND ENGINEERING

Department of Industrial System and Engineering

Student Diary

1. Student's Name / ID: BISSESSUR Aishani Prama / 170931
2. Programme Name: **BSc.(Hons) Computer Science with Network Security** Cohort: **BCNS17BFT.**
3. Supervisor's Name: Mr. Toe Kai Wai Dudley
4. Student Telephone No.: Residential/Office 6951152 Mobile: 5827 6184
5. Student Email Address: aish.bissessur@gmail.com
6. Date of Meeting: 16:09:2019
7. Comments

8. Actions Taken

- Should buy RFID Sensors (Receiver and Sender) - Understand the full scenario of the project - Use Firebase for storage

9. Action to be taken for next time

- Introduction of the project - gantt chart - literature Review

10. Date for next meeting: 30:09:19

Student Signature: A.P. BISSESSUR Date: 16:09:2019

Supervisor's Signature: [signature] Date: 16/9/19

UNIVERSITY of TECHNOLOGY, MAURITIUS

Annex A

SCHOOL OF INNOVATIVE TECHNOLOGIES AND ENGINEERING

Department of Industrial System and Engineering

Student Diary

1. Student's Name / ID: BISSESSUR Aishani Prama / 170731

2. Programme Name: BSc.(Hons) Computer Science with Network Security Cohort: BCNS17BFT.

3. Supervisor's Name: Mr. Tse Kai Wai Dudley

4. Student Telephone No.: Residential/Office 6795152 Mobile: 5827 6184

5. Student Email Address: aish.bissessur@gmail.com

6. Date of Meeting: 30.09.19

7. Comments

8. Actions Taken

- Application should have an admin and client side.
- Admin to add train detail.
- client to send queries to the admin.

9. Action to be taken for next time

- Create use case diagram, sequence diagram.
- literature Review.

10. Date for next meeting: 31.10.19

Student Signature: A.P-Bissessur Date: 30.09.19

Supervisor's Signature: Date: 30/9/19

UNIVERSITY OF TECHNOLOGY, MAURITIUS

Annex A

SCHOOL OF INNOVATIVE TECHNOLOGIES AND ENGINEERING

Department of Industrial System and Engineering

Student Diary

1. Student's Name / ID: Bissessur Aishani Prama /170231
2. Programme Name: BSc.(Hons) Computer Science with Network Security Cohort: BCNS17BFT.
3. Supervisor's Name: Mr. Tse Ka Wai Dudley
4. Student Telephone No.: Residential/Office 6175152 Mobile: 58276154
5. Student Email Address: aish.bissessur@gmail.com
6. Date of Meeting: 31.10.19
7. Comments

8. Actions Taken

- Discuss about the layout of the mobile app and function.

9. Action to be taken for next time

- Design interface of the application. - Try to learn to do the connection of the sensors.

10. Date for next meeting: 28.01.2020

Student Signature: A.P.Bissessur Date: 31.10.19

Supervisor's Signature: Date: 31/10/19

UNIVERSITY OF TECHNOLOGY, MAURITIUS

Annex A

SCHOOL OF INNOVATIVE TECHNOLOGIES AND ENGINEERING

Department of Industrial System and Engineering

Student Diary

1. Student's Name / ID: BISSESSUR Aishani Prama /190731
2. Programme Name: **BSc.(Hons) Computer Science with Network Security** Cohort: **BCNS17BFT.**
3. Supervisor's Name: Mr. Tse Kai Wai Dudley
4. Student Telephone No.: Residential/Office 695152 Mobile: 58276184
5. Student Email Address: aish.bissessur@gmail.com
6. Date of Meeting: 28.01.2020
7. Comments

8. Actions Taken

- Dicuss about the functional requirements. - Dicuss about the arduino sensors.

9. Action to be taken for next time

- Implementation of the application. - layout, UI and function.

10. Date for next meeting: 04.03.2020

Student Signature: A.P. Bissessur Date: 28.01.20.20

Supervisor's Signature: Date: 28/1/20

UNIVERSITY of TECHNOLOGY, MAURITIUS

Annex A

SCHOOL OF INNOVATIVE TECHNOLOGIES AND ENGINEERING

Department of Industrial System and Engineering

Student Diary

1. Student's Name / ID: BISSESSUR Aishani Prama / 190731

2. Programme Name: BSc.(Hons) Computer Science with Network Security Cohort: BCNS17BFT.

3. Supervisor's Name: Mr. Tse Kai Wai Dudley

4. Student Telephone No.: Residential/Office 8175152 Mobile: 5827 6184

5. Student Email Address: aish.bissessur@gmail.com

6. Date of Meeting: 04.03.2020

7. Comments

- The Station Name should already be stored in the database. Just need to call it on the spinner.

8. Actions Taken

- Driver app should be created also, so that the latter can send notification to the Admin about lateness and reason.
- Should be able to do different kind of search.

9. Action to be taken for next time

- Work on the driver part.
- Try to different kind of search.
- Driver send notification to driver.

10. Date for next meeting: 20.03.2020

Student Signature: A.P. Bissessur Date: 04.03.2020

Supervisor's Signature: [signature] Date: 4/3/20

UNIVERSITY OF TECHNOLOGY, MAURITIUS

Annex C

SCHOOL OF INNOVATIVE TECHNOLOGIES AND ENGINEERING

Department of Industrial System and Engineering

Project Submission Form

PART 1 – TO BE COMPLETED BY THE STUDENT IN BLOCK LETTERS

Surname: BISSESSUR

First Name(s): AÏSHANI PRAMA

Student ID Card No.: 170731

Undergraduate Degree: **BSc(Hons) Computer Science with Network Security**

Dissertation Title:

RAILWAY TRACKING AND ARRIVAL TIME PREDICTION

Submission Date: 17/08/2020

I wish to submit two copies of the Project detailed above in compliance with the requirements for the programme.

I confirm that this Project is submitted as part requirements of the above-mentioned programme.

Student Signature: A.P.BISSESSUR Date: 13/8/2020

PART 1 – TO BE COMPLETED BY THE SUPERVISOR

Supervisor's Name: TJE KAI UAM Dudley

I hereby grant the abovementioned student consent to submit his/her project. The student has complied with all the requirements as per the Project Guideline.

Supervisor's Signature: Date: 13/8/2020

UNIVERSITY of TECHNOLOGY, MAURITIUS

La Tour Koenig
Pointe-aux- Sables
Republic of Mauritius
Tel : (230) 234 7624
Fax : (230) 234 6727

CERTIFICATE OF ORIGINALITY

Annex D

For Undergraduate Dissertation / Project Report

I hereby declare that the intellectual content of this dissertation / project report* is the product of my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the University or any other institute, except where due acknowledgement and references are made in the text.

I understand that there are regulations against Plagiarism and that I am fully aware of consequences of breaching such regulations.

Student Name/Number: BISSESSUR AISHANI PRAMA

Title of Dissertation:

RAILWAY TRACKING AND ARRIVAL TIME PREDICTION

Programme: BSc.(Hons) Computer Science with Network Security

School in which registered: School of Innovative Technologies and Engineering

Signature: A P Bissessur

Date: 17/08/2020

(Please cross out as appropriate)*