Cairo University

Faculty of Computers & Artificial Intelligence

Operations Research & Decision Support Dept.

"Easy Trip" Metro Application

Student Name	ID
Mustafa Sayed Gad-Allah	20180277
Salah Mohamed	20180138
Malek Mohamed Amin	20180207
Amira Mosad Diab	20180058
Sara Adel Abd El-Rahman	20180115

Supervised By

Prof. Mohamed Saleh
Dr. Marwa Sabry
Eng. Yousra Mustafa

Cairo University

July 2022

ABSTRACT

Cairo, the capital of Egypt, is the most populated city in Africa, and as a result, it has the most crowded metro system on the continent, with an average of 3.5 million daily riders. The Cairo Metro will be busier than it can handle given the growth rate of population. In order to meet the potential demand that may arise in the upcoming few years, the government is already striving to improve the metro's capacities through the acquisition of additional trains and the rehabilitation of the existed old trains.

Our plan was to estimate the number of crowds that will increase in the upcoming years and plan the operation of metro trains, which couldn’t be accomplished due to data unavailability, so we changed our plans to help both sides, the riders, and the metro administration. We wanted to let the regular rider be a decision-maker on his own by giving him insights about the metro rush to let it easier to him to choose between the metro or other transportations based on the estimated rush. On the other hand, we wanted to give the metro administration a sustainable method to get metro insights. So, to assist both sides we created a mobile application that helps the normal user in different ways, he can use the application to know the source station location, the rush percentage for this station based on our calculations, the estimated rush for the line he will be in, and he can use QR code that will be used instead of using his subscription card. On the other hand, we will let the admin have several features, the admin will have the ability to send a custom message to the application users as a notification to advise for any emergencies or unpredicted events, and he can see insights about the metro in previous intervals as he wants it may be a week, month, or even a year. He can use this feature for a specific line or all of the three lines.

We employed a variety of tools to accomplish our primary objective, with Flutter serving as the major one. Google developed the open-source UI software development kit known as Flutter. It is used to develop cross-platform software from a single codebase for Google Fuchsia, Android, iOS, Linux, macOS, Windows, and the web. The primary technique we used to sort and clean the data was Microsoft Excel. Then we used the JSON format for creating the database. We worked using Firebase, a Google-developed platform that supports app development generally and offers a variety of features for developing mobile applications.

DECLARATION

We hereby declare that our dissertation is entirely our work and genuine / original. We understand that in case of discovery of any PLAGIARISM at any stage, our group will be assigned an F (FAIL) grade and it may result in withdrawal of our Bachelor’s degree.

Group Members

Name	**Signature**
Student name 1	___________________
Student name 2	___________________
Student name 3	___________________
Student name 4	___________________
Student name 5	___________________

PLAIGRISM CERTIFICATE

This is to certify that the project entitled "***Easy Trip Metro Application***", which is being submitted here with for the award of the "***Bachelor of Computer and Artificial Intelligence Degree***" in "***Operations Research and Decision Support***". This is the result of the original work by Mustafa Sayed, Salah Mohamed, Malek Mohamed, Amira Mosad, and Sara Adel under my supervision and guidance. The work embodied in this project has not been done earlier for the basis of award of any degree or compatible certificate or similar tile of this for any other diploma/examining body or university to the best of my knowledge and belief.

Turnitin Originality Report

Processed on 15-Jul-2022 00:14 PKT

ID: 300502964

Word Count: 8801

Similarity Index

10%

Similarity by Source

Internet Sources:	06%
Publications:	0 %
Student Papers:	08%

Date: 17/07/2022

Prof. Mohamed Saleh

Dr. Marwa Sabry

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Introduction

As Egyptians, we care about the advancement of our nation and do everything in our power to enhance the standard of living for Egyptians. We undertook this initiative in order to assist everyone who would use the Metro in Egypt. We developed a smartphone application that can help prevent congestion and provide you with all the information and support you require to be free to make your own decisions. We made it easy to use, functional, and fun!

The Egyptian National Railways (ENR) operates the Cairo Metro, which transports up to 3.5 million passengers per day. Cairo is one of Africa's most densely populated cities, with a population of almost 22 million people. Cairo Metro is Africa's first metro system, having been active since 1990. Currently, the government is working on developing the first three lines as well as adding three new ones to accommodate the growing population.

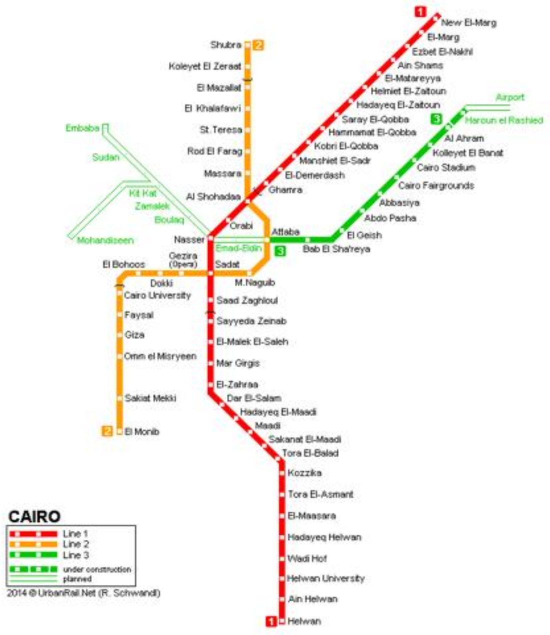


Figure 1.1-1 Egypt Metro map

The metro is regarded as the backbone of Cairo's transportation system. It is the first underground Metro in Africa, the Middle East, and the Arabian Peninsula. Cairo has made it a priority to adopt this innovative, safe, and efficient mode of public transportation.

- **The First Line**: The first line extends from El-Marg El-Gadidah Station to Helwan Station in south Cairo. It consists of 35 stations. It has three interchange stations, two of which are Anwar El-Sadat station and El-Shohadaa station connecting Line 1 and Line 2, and one Gamal Abdel Nasser station connecting Line 1 and Line 3 (still under construction).
- **The Second Line**: It extends from Shubra El-Kheima station to Giza station. It consists of 20 stations. It has three interchange stations Anwar Al-Sadat station and Al-Shohadaa station connecting Line 2 and Line 1 and Attaba station connecting Line 2 and Line 3.
- **The Third Line**: It extends from Cairo Airport station to Embaba station. It has three interchange stations Gamal Abdel Nasser station connecting Line 1 and Line 3, Attaba station connecting Line 2 and Line 3, and Cairo University station connecting Line 2 and Line 3. All of the stations located after Attaba station are still under construction,

For each line of the metro three lines, there’s a specific number of trains and a specific one for the trailing time.

- **The First Line**:
  - Number of trains: 63 trains (25 trains with air conditioning and 38 without).
  - Number of units in each train: 3 units.
  - Number of metro carriages per unit: 3 carriages.
  - The carrying capacity of the metro carriage: 300 passengers.
  - Min Trailing time: 140 seconds.
  - Max Trailing time: 360 seconds.
  - The line’s total length is 44 KM.
- **The Second Line**:
  - Number of trains: 39 trains (4 trains with air conditioning and 25 without).
  - Number of units in each train: 2 units.
  - Number of metro carriages per unit: 4 carriages.
  - The carrying capacity of the metro carriage: 300 passengers.
  - Min Trailing time: 165 seconds.

    - Max Trailing time: 360 seconds.
    - The line’s total length is 21.5 KM.
- **The Third Line**:
    - Number of trains: 15 trains (15 trains with air conditioning).
    - Number of metro carriages: 12 carriages.
    - The carrying capacity of the metro carriage: 300 passengers.
    - Min Trailing time: 300 seconds.
    - Max Trailing time: 420 seconds.
    - The line’s total length is 23 KM.

1.2 Problem Domain

Cairo’s population has been increasing with a growth rate average of 2% over the previous 25 years. Egypt, in general, ranked 14th over the whole world on the number population, based on statistics in Egypt there’s a newborn every 13 seconds.

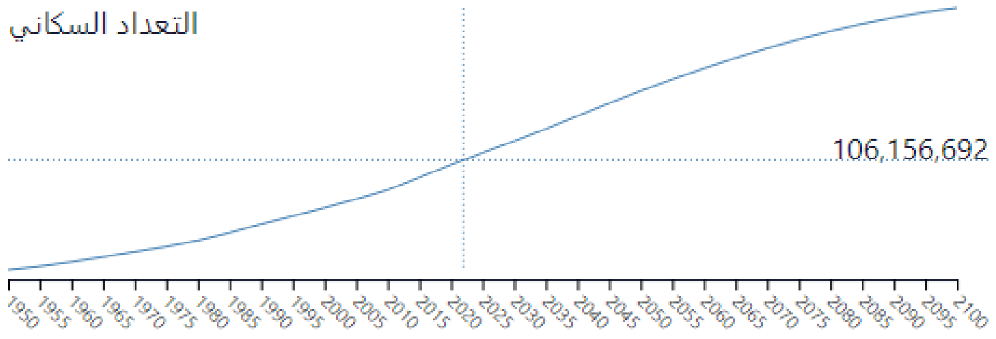


Figure 2.2-1 Egypt’s population

Cairo is considered the most populous city in Egypt, and the graph below shows its growth rate from 1950 to 2037 is very relevant while talking about Cairo congestion (the interval from 2022 to 2037 is predicted based on the historical data from the UN statistics).

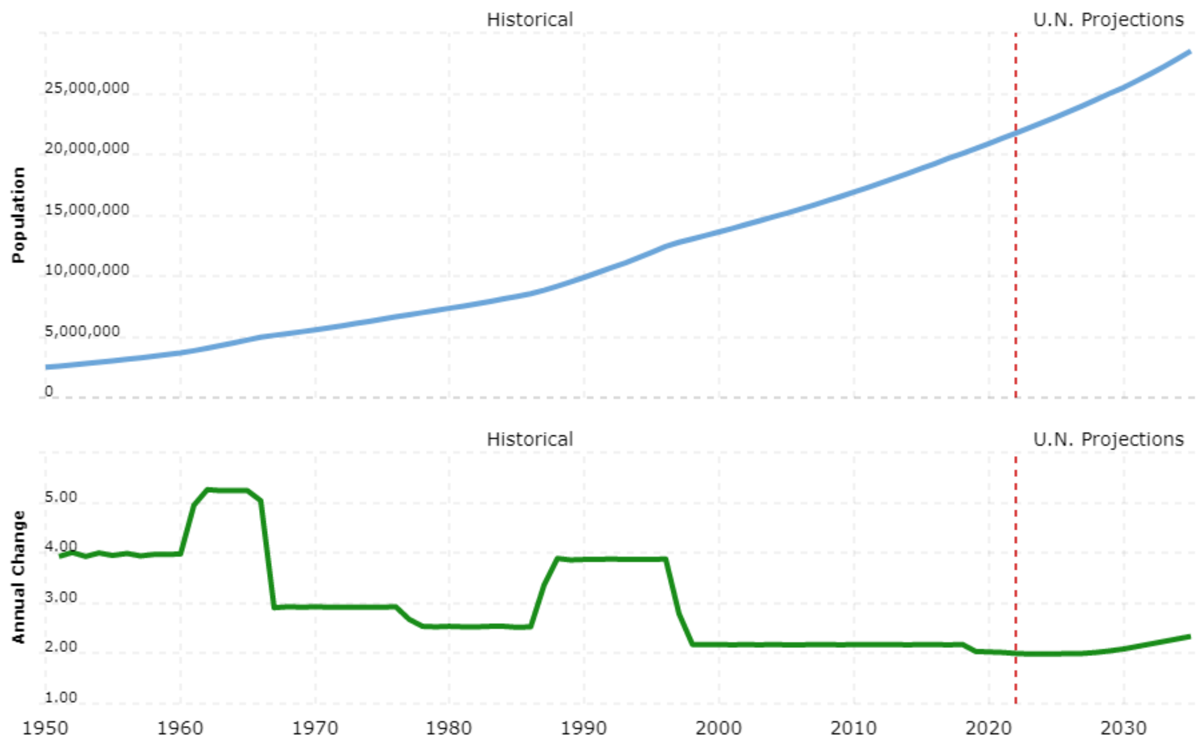


Figure 1.2-2 Cairo's population and growth rate

The Cairo Metro, which transports an average of 3.5 million people each year, is one of the most significant transportation systems that must be improved to handle the crowding that may occur according to the abovementioned data.

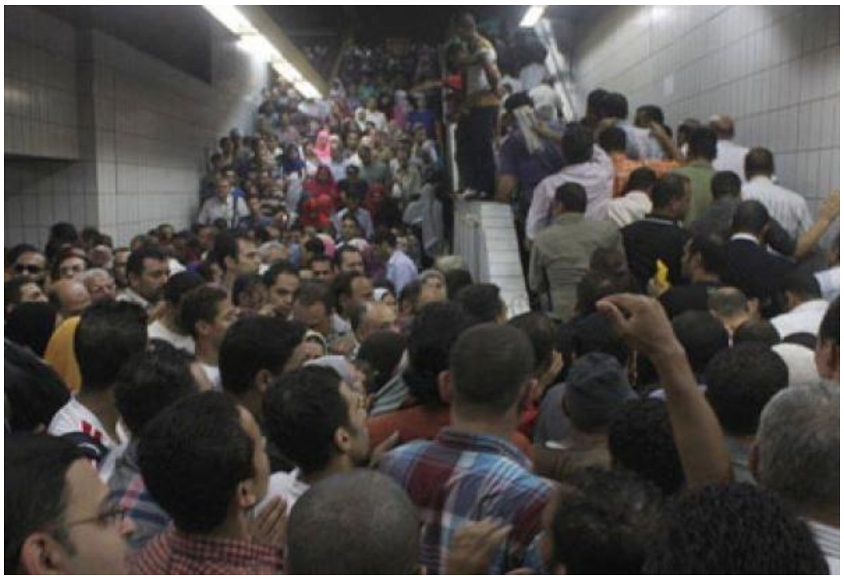


Figure 1.2-3 Crowd in Metro

1.3 Problem Statement

- There will be crowding in the Cairo Metro as a consequence of the rising population growth rate.
- The majority of the time, a very long line of customers waits in front of the ticket window since the passengers are unsure about the accurate ticket price that is adequate to their route (based on the number of stations).
- On occasion, individuals who have a Metro subscription forget their subscription card, which necessitates them to purchase a ticket and lengthen the line.
- There are occasionally crowded trains in the Metro because the Metro administration wasn't aware that a day would be overcrowded.
- Metro has a lot of prospective customers because it is faster than other modes of transportation, but because they are unsure of whether it is crowded or not, they choose to utilize other modes of transit.

Figure 1.3-1 Metro Crowd in the second line

1.4 Proposed System

There’s a mobile application called “Cairo Metro” we reviewed already before we started developing our “Easy Trip” application. We tried to solve his problem in our application, we also developed multiple new features that will help enhancing the overall user Metro experience.

1.4.1 Aims & Objectives

The overall purpose of our project is to reduce metro overcrowding in various ways while also benefiting Metro passengers and Metro administrators by utilizing the variety of features provided in our mobile application “Easy Trip”.

1.4.2 Proposed System Features

- We needed to make it simpler for the user to know the expected rush status, for both his route and a particular station.
- We want to inform the user the pricing for the tickets he wants so that he will be prepared in front of the ticket clerk window while he is paying for them.
- If a user has previously subscribed to one of the Metro subscriptions, we gave him the option to enter via a QR code scanner on his mobile device through our "Easy Trip" application.
- We need to give the Metro administration the essential insight and information so that it will be easy for them to organize operations and evaluate previous data to forecast the number of riders for the upcoming days, weeks, or even years. Therefore, we developed an analysis feature.
- The Metro administration can alert all passengers using the customized notification feature of the “Easy Trip” application if there is any emergency or unexpected incident that causes a delay or technical issue in subway carriages.

1.5 Development Methodology

We employed a hybrid methodology for our project, taking advantage of both the Waterfall and Agile methodologies' advantages. A hybrid approach aims to allow for both up-front requirements specification (planning, Designing, etc.) in the manner of a waterfall approach and changeover to an agile approach for design reviewing, objectives prioritizing, requirement reviewing, development, and testing.

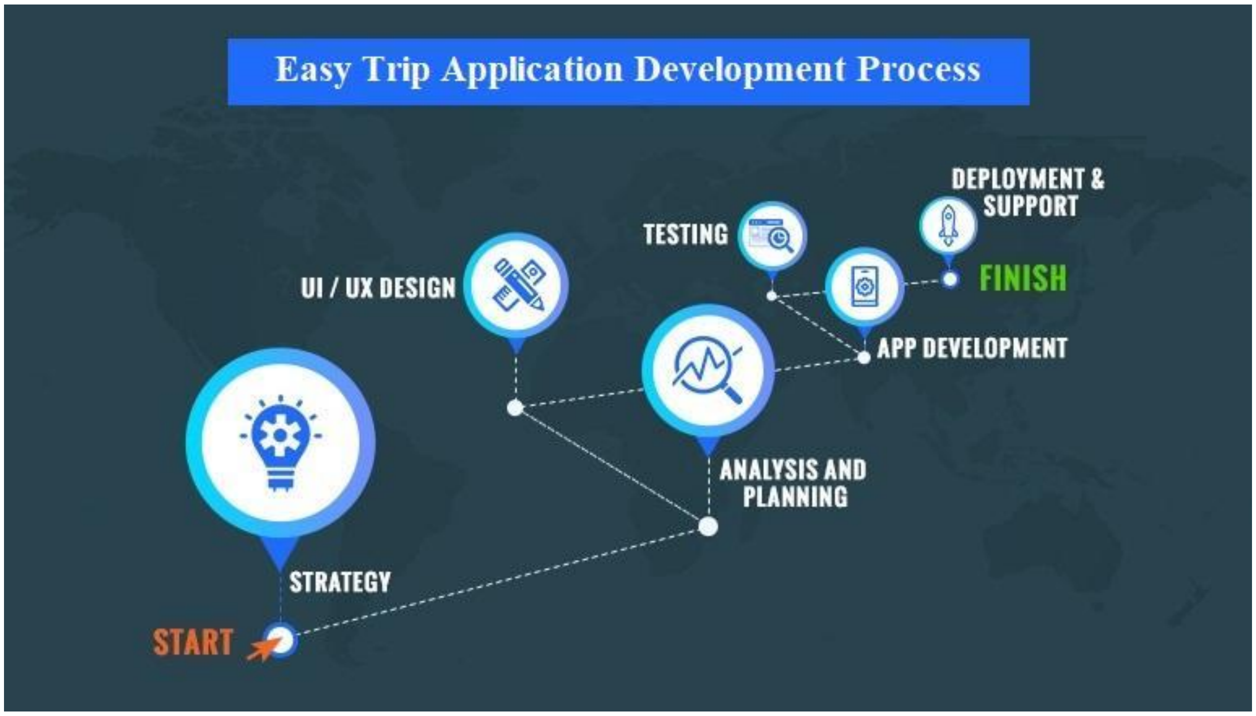


Figure 1.5-1 Development Process

We have multiple phases in our project, we started with the below sequence

- Data Creation: we used excel sheets and Google sheets to create database using probabilities.
- Data Sorting and Cleaning: we used excel sheets to sort data and Rapidminer tools for cleaning it.
- Database transforming: we used python programming language to transform data from CSV format to JSON format.

- Application Creation: we used Flutter to develop our mobile application, also we used the Google Firebase to support our application backend by using its Realtime Database and its advanced queries.
- Mathematical Model: we used online tools to solve our mathematical model beside using the Tora desktop application.

1.6 Resource Requirements

- Subsystem requirements
    - Flutter.
    - MySQL.
    - Databases JSON format.
    - Python programming language.
    - Excel sheets.
    - Photoshop.
    - Statistics and probabilities.
    - Visualizing diagrams.
    - Firebase handling.
- Localization requirements: Our application will be used only in Egypt as it’s concerned with enhancing Egypt Metro systems by providing a smooth experience to both users and administrators.
- Data requirements:
    - Well knowledge of Metro stations’ rush to define rush probabilities for each station within a specific range of probabilities.
    - Well knowledge in Metro rush timing to define rush probabilities for each hour within a specific range of probabilities.
    - Creating crowd numbers for each station based on probabilities regarding the current hour and date.
    - Data cleaning.

1.7 Report Layout

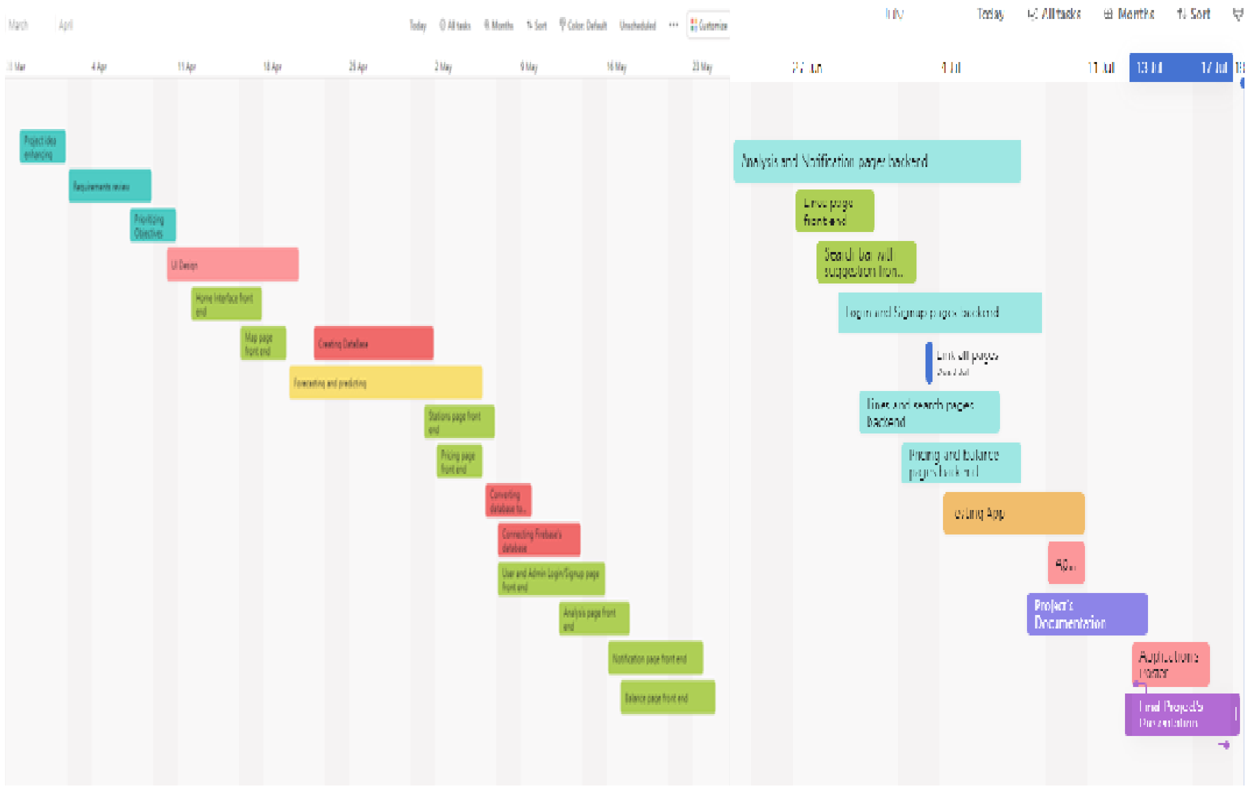


Figure 1.7-1 Report Layout

CHAPTER 2

BACKGROUND & EXISTING WORK

2.0 Introduction

Our mobile application “Easy Trip” Has been developed using Google's portable UI toolkit, Flutter, which allows the creation of stunning, natively built applications from a single codebase for desktop, mobile, and the web.

Flutter is free and open source, integrates with existing code, and is utilized by developers and businesses all over the world. It’s considered the most popular cross-platform UI toolkit as it has more than 0.5 million users. We developed this application to provide a smooth experience for Metro riders and to let it easier for the Metro administration to plan operations based on our analysis.

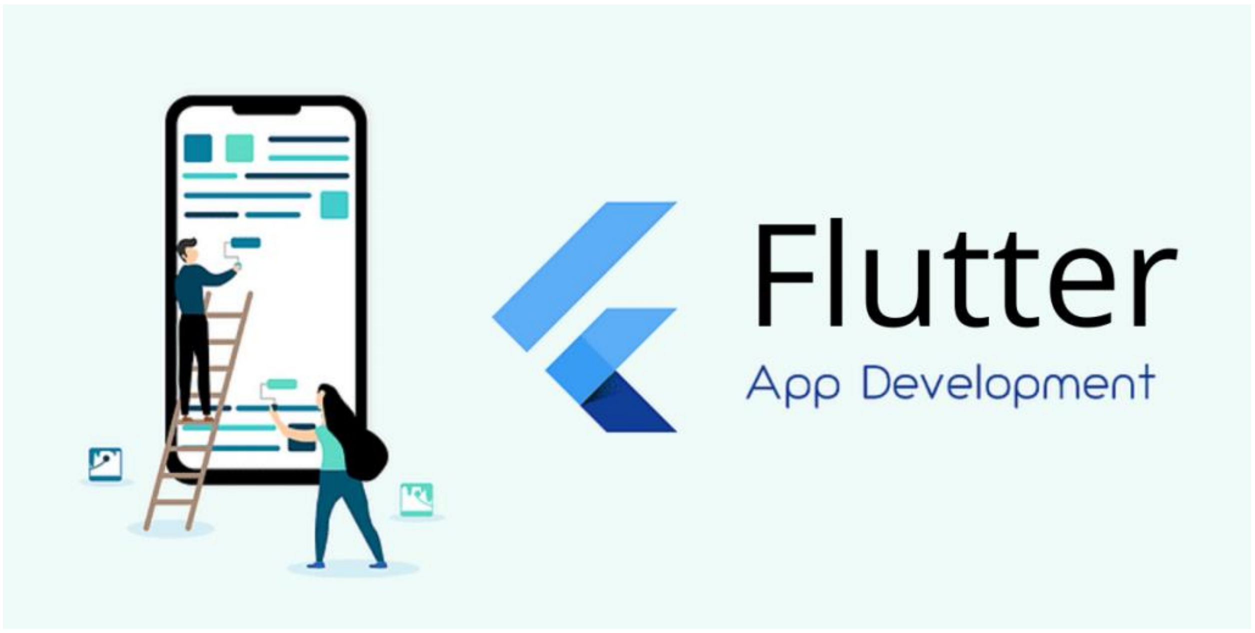


Figure 2.0-1 Flutter

- Flutter Advantages:
    - Hot reload in Flutter allows for quicker coding. In order to observe changes, iOS and Android developers previously have to create code, wait for it to compile, and then load it on the device. However, users can quickly or instantly check the impacts thanks to Flutter's hot reload.
    - Flutter has Ready-to-use widgets. Additionally, it is simple to construct apps of any complexity thanks to the customizable widget kit. Regardless of screen size, widgets are quick, dependable, and flexible. Additionally, a substantial selection of Material and Cupertino ready-made widgets are available. It is also feasible to deal with animation and gesture processing in a flexible manner. As a result, everything goes more quickly and easily.
    - Flutter is supported by a considerable proportion of the community. Google creates Flutter so that Google can provide Flutter developers with regular upgrades and bug fixes. Additionally, Flutter offers significantly better performance compared to competing technologies. The community for Flutter app development has been expanding widely.
    - Flutter is very good documentation and resources.
    - Flutter has a significant UI that is so much functional, and its applications are very smooth in performance.
- Flutter disadvantages:
    - We are aware that Google is the creator of Flutter. Flutter-built apps are highly entertaining on Android, but they have some issues on iOS. Google is working on it because they want to dominate the iPhone/iOS app development business in the USA.
    - Flutter doesn't support third-party libraries, which occasionally makes programming more difficult. However, Flutter has the potential to be the finest if an app is well planned.
    - Both Apple TV and Android TV are not supported by Flutter.
    - Dart is a suitable language for Flutter, yet coding can occasionally be tricky due to its demanding nature.
    - The size of Flutter apps is bigger.

We used Firebase which is a Google-backed platform for building mobile and online applications, Firebase is built on the backend as a service (BaaS) model and includes several useful services and useful APIs. Additionally, it includes a simple integration procedure with Android, iOS, and Unity installations that can be used to create programs for all popular mobile and online platforms. Firebase is used by many bespoke mobile app development businesses to create applications for their clients because it can manage data in real-time and give consumers a better app-usage experience.

We used one of Firebase's core features, we used Firebase Realtime Database. The data is stored as JSON and continuously synced to each related client in the cloud-hosted database that Firebase supports. Modern applications require a real-time database instance that refreshes the most recent data.

- JSON features:
  - JSON is self-describing.
  - Text is straightforward in JSON. Due to the fact that more complicated document types are not easily shared between platforms and operating systems, it is suitable and secure for doing so. JSON may be easily displayed and edited in basic editors as text.
  - JSON is an abbreviated language. The size of an average JSON string is roughly two-thirds that of the equivalent XML data.
  - JSON is a simple language to pick up, read, and comprehend.

2.1 Project Overview

The mobile application that we worked on will minimize the crowd from two perspectives, the Metro Administration's perspective, and the metro rider's perspective.

- The Metro Administration perspective: our application will be integrated with the metro gates so we can automatically store all the ins and outs from Metro gates in our database, which will help us in providing the Metro admins with analysis of this data to give them insights about the riders' peaks and changes. We asked one of the Metro engineers in our first steps in collecting data about the operation plans and how they operate the trains on holidays and weekdays, and he stated that the Metro is operated based on fixed plans that have been operated for more than twenty years based on expertise and their view of how many cars should be operated and in which days, also they determined the trailing time between trains with the same way based on their experience and vision but not on technology not on data and calculations, so we will take advantage from the stored ins and outs data and give them insights by analyzing this data and visualizing it to see changes over time for each line. We added another feature to let them have a wider vision by using forecasting techniques to forecast metro crowd based on data stored in our database, this will help them in planning the operation and minimizing both the crowd and the operation cost.

- The riders' perspective: We previously stated that our mobile application will be integrated with the Metro gates and because of that we can instantly store all ins and outs in our database. We will reap the benefits of this advantage to assist the riders in knowing based on our calculations the estimated rush for their nearest station that they will start their path from, as well as we will provide an estimated rush for lines also based on our calculations, so the rider can decide whether he should ride the Metro or not. We'll give the user additional features. He can use our app to enter the Metro instead of the subscription card if he has a Metro subscription, but that won't be possible until the application is fully integrated with the Metro systems. The rider can also use our application to get station location and ticket pricing depending on the number of stations and the number of tickets. The entire Metro map can also be seen by the passenger using our application.

Figure 2.1-1 Metro in Egypt

2.2 Project Limitations

2.2.1 Innovations in project

We solved the issues and limitations that occurred in the project in different ways:

- We created dummy data based on probabilities, we put an average number of riders (Passengers) in the station per hour, then we added a table that contains an interval of rush probabilities based on the station rush in reality, also we considered a probability for hours' rush because every hour has its own factors in determining rush.
- We grouped every 6 months in our database together, then converted them using python to JSON format. After converting all the 7 years' 14 files to the JSON format we grouped them together again into one JSON file.
- Instead of using Google Maps APIs we used hyperlinks that direct users to station location on Google Maps.
- We created a balance page for user to let him able to access Metro gates using QR scanner which let be too much easier than the current situation. This balance page contains user's trips and his subscription interval.
- We created a feature for the Metro Administration that provide them with the Metro crowd insights that helps improving the Metro operations planning.
- For the Metro administration we added a feature that helped in notifying the Metro riders if there's any unpredictable emergency.
- We created a new feature for the user that provide him with the Metro lines estimated rush, and also if he wants to know specific station's rush, he will use our application to determine its rush status.

2.2.2 Design of project

Each page in our "Easy Trip" mobile application is unique and has its own features and qualities. All of our pages are included on our main page. We may access the Login page from the home page to log in as a user, the Metro map from the home page to preview all the Metro stations, and the Pricing page from the home page so that the user can benefit from it by using the pricing or previewing the line rush. Additionally, we may access the Lines page, which lists all three lines as well as each line's stations and their respective station pages includes every station estimated rush based on our database.

Figure 2.2-1 Metro Egypt

CHAPTER 3
REQUIREMENTS AND WORK DETAILS

3.1 Project Stakeholders

Our application benefits two different categories of users who will utilize our program.

- Metro Riders: We will be able to immediately store all ins and outs in our database because of the integration of the Easy Trip mobile application with the Metro gates. We will leverage the power of this advantage to enable the riders by letting them know the estimated rush hour for the station closest to where they will begin their journey, as well as by providing an estimated rush hour for the lines based on our computation, so the rider can decide whether it would be better for him to take the Metro or not. The user will get more features from us. If he has a Metro subscription, he can use our app to access the Metro rather than using the subscription card, but it won't be available until the app is completely integrated with the Metro infrastructure. The user of our application can also use it to find out the price of a ticket based on the number of stations and the number of tickets. The passenger can use our mobile application to view the whole Metro map as well.
- Metro Administration: The metro gates will be integrated with the Easy Trip application, allowing us to automatically store all ins and outs from Metro gates in our database. This will enable us to provide the Metro admins with an analysis of this data, giving them insights into the peaks and fluctuations in ridership. By analyzing and visualizing this data to monitor changes that occur for each line, we will take advantage of the ins and outs data that has been stored and provide the Metro Administration with insights. By applying forecasting techniques to predict metro crowds based on data contained in our database, we introduced another function to provide them with a wider perspective. This will help them plan the operation and reduce both the crowd and the operation cost.

3.2 Use Case Requirements

3.2.1 Requirements

Table 3.2.1-1 Requirements

Requirements	Description
User Interface	Includes home page and its content.
Ticket Pricing	Allow users to calculate their tickets pricing.
Route Rush	Allow users to know the estimated rush for the lines.
Search Bar	Allow users to search for a specific station.
Lines page	Previews all the Metro lines, and every line contains all of its stations
Station Page	Allow users to know the station details, location, rate, and the estimated rush.
User Register	User account creation.
User Login	Allow the user to access his account.
User Balance	Allow the user to check his subscription card balance and the remaining days till the renewal.
Admin Login	Allow the admin to access his account.
Analysis Page	Allow the admin to analyze the crowd in a specific line for a specific interval.
Custom Notification Page	Allows the admin to send a custom message as a notification to all the application users to notify them about a specific incident.

3.2.2 User System Requirements

- Home page: includes four buttons and a search bar
  - The search bar: open the station details page e for the specific page that the user searched for.
  - Pricing: open a page that determines ticket pricing and route in.
  - Map: open page previewing all the Metro lines map.
  - Login: open the login page.
  - Lines page: open lines page to preview.
- Station page: includes the station description, rating, location, and the estimated rush for this station.
- Pricing page: includes two bars and three buttons
  - The first bar to select your starting point (the station you will start your trip from).
  - The second bar to select your ending point (the station you will end your trip at).
  - A minus (-) button and (+) button: determine the number of tickets you want to check pricing.
  - Route rush: open a page that contains the rush estimation for each line.
- Lines page: contains the three lines and every line contains its stations details.
- Map page: it has the photo of Metro’s three lines mapped.
- Login page: contain user-required details to use in login, and three buttons
  - Login: open the user page after the required data is inserted.
  - Signup: open the sign-up page to create a new account.
  - Login as admin: open the admin page to log in.
- User Page: contains the user’s subscription card’s balance listing the remaining trips out of the total, and the remaining days till the renewal. It also contains a button that opens the camera to scan the QR code.

3.2.3 Admin System Requirements

- Admin Page: contains two buttons
    - Analysis: open the analysis page where we visualize the data and analyze it in.
    - Custom Notification: open a page to send a custom notification to all users.
- Analysis Page: contains three bars and a button
    - The first bar's to choose the line(s).
    - The second bar is to determine the starting date.
    - The third bar is to determine the ending date.
    - The button is used to visualize.
- Custom Notification Page: Contains a box to write a custom message in and a button to send it to the application users.

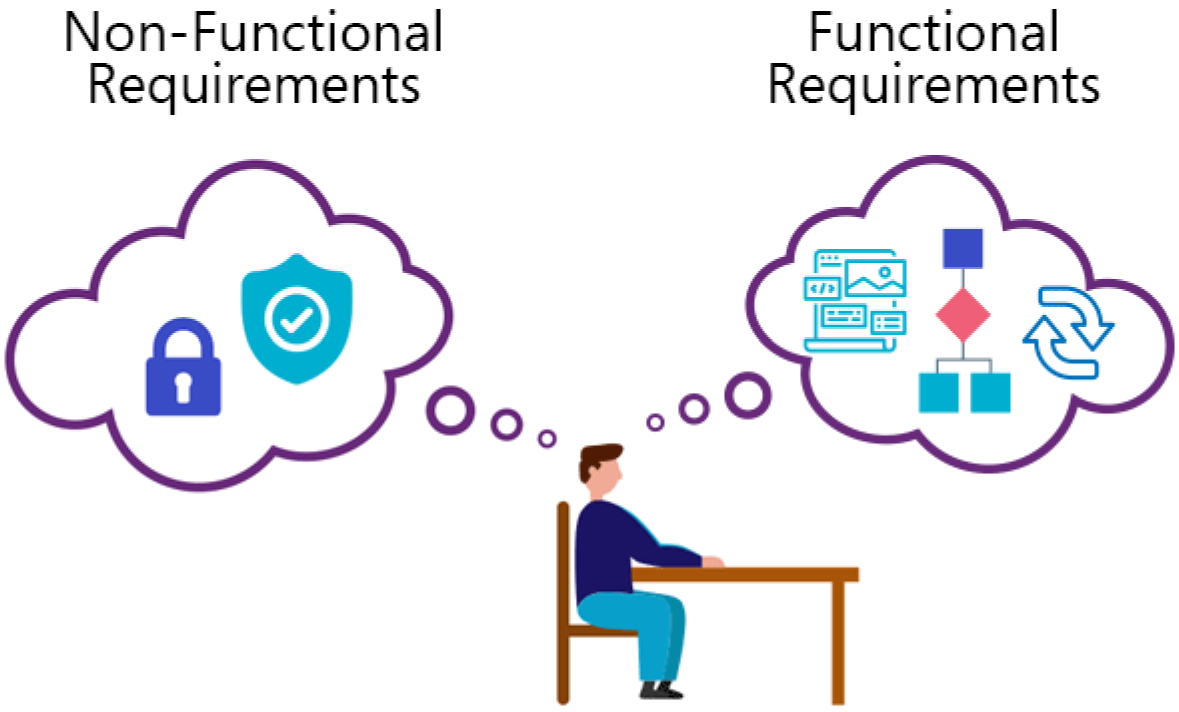


Figure 3.2-1 Project Requirements

3.3 Project Non-Functional Requirements

3.3.1 Execution qualities

- Availability: Our application works 24/7 as we supported it with the Google Firebase.
- Reliability: In case of alarm status our application data will be retrieved automatically from the Firebase whenever it restarts.
- Performance: Thanks to the Google Firebase support our application can easily handle multiple users simultaneously.
- Security: As we authenticate users and create security rules, our application is properly protected, and we can completely limit who may read and write to our Firebase data.
- Usability: Our application has a user-friendly interface that allows user to smoothly use our application without facing any type of difficulties.

3.3.2 Evaluation qualities

- Testability: We are testing our application by entering both valid and invalid data into the login pages. We can also test by looking at how busy Metro lines and stations are at various hours to see how rush changes according to the hour. Finally, we test pricing by selecting various starting stations and various destination stations to see how prices change according to the number of stations in our path.
- Maintenance: Our application can be maintained at any moment, so if an upgrade, modification, or even maintenance is required, we can accomplish it immediately anytime we need to.

Figure 3.3.1 Requirements

3.4 Alternative Flow

Alternative flow diagram

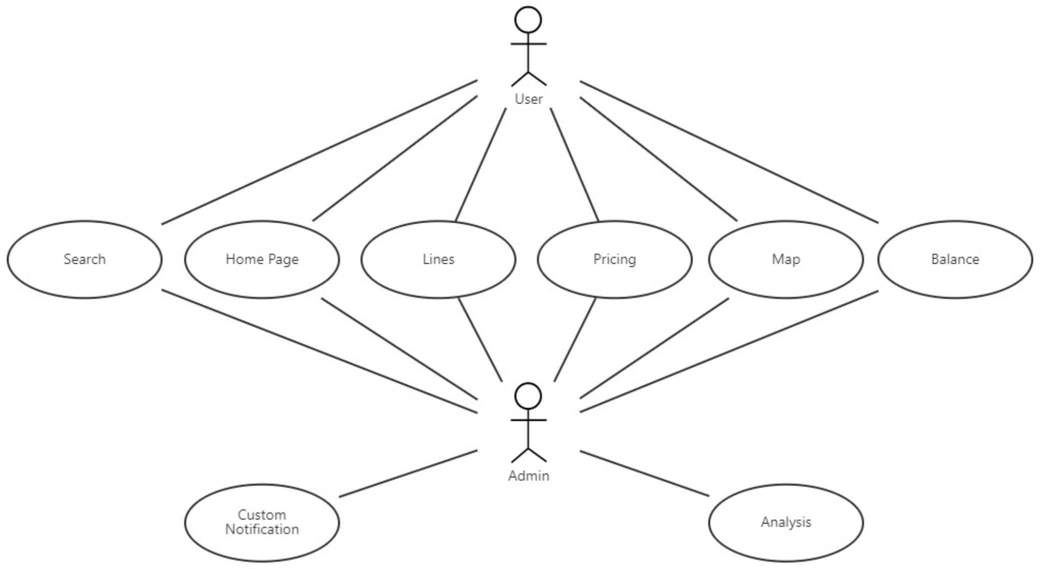


Figure 3.4-1 Alternative Flow

Alternative flow diagram for operations

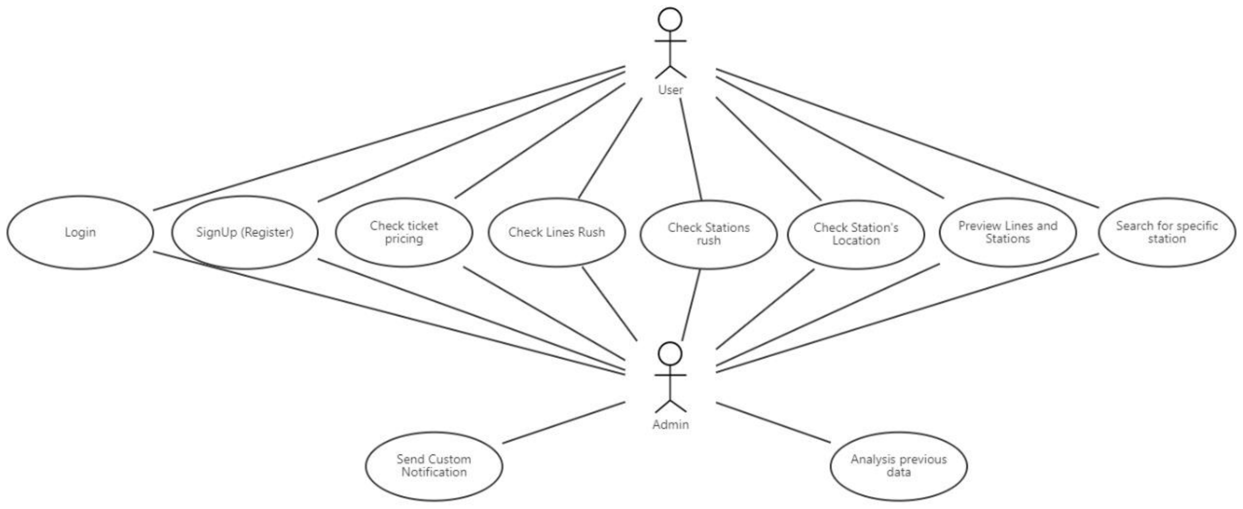


Figure 3.4-2 Alternative Flow Operation

CHAPTER 4

MATHEMATICAL OPTIMIZATION

4.1 Introduction

Mathematical optimization is the selection of a best element, with regard to some criterion, from some set of available alternatives. Optimization problems of sorts arise in all quantitative disciplines from computer science and engineering to operations research and economics, and the development of solution methods has been of interest in mathematics for centuries.

The Mathematical model is a branch of the Applied Mathematics, it has applications across numerous industries, found below some industries of them:

- Manufacturing
- Production
- Inventory control
- Transportation
- Scheduling
- Networks
- Finance

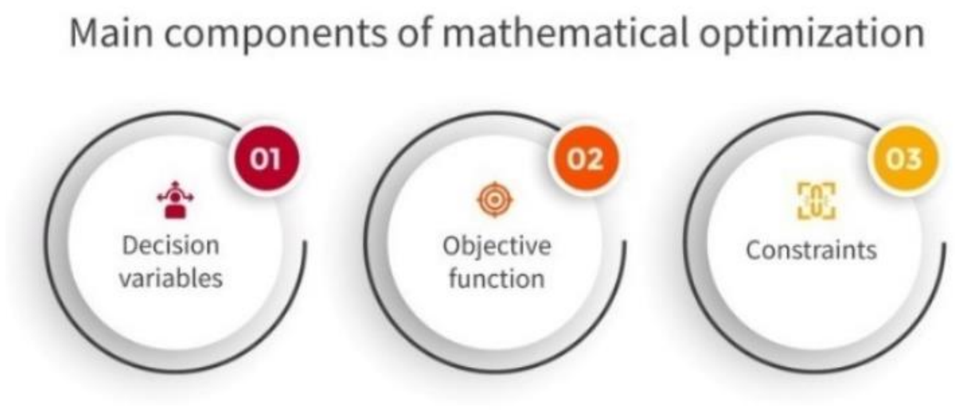


Figure 4.1-1 Mathematical Optimization Main components

4.2 The Fundamental Optimization Problem Components

The fundamental optimization problem consists of:

- The output you're attempting to maximize or minimize is represented by the objective function, f(x).
- Variables, or things you can control, such as x1, x2, x3, and so on. They can be referred to by the acronyms xn for individuals or x for a group.
- Equations called constraints set upper and lower bounds on the size of particular variables. In most cases, equality constraints are marked as hn(x) and inequality restrictions as gn (x).

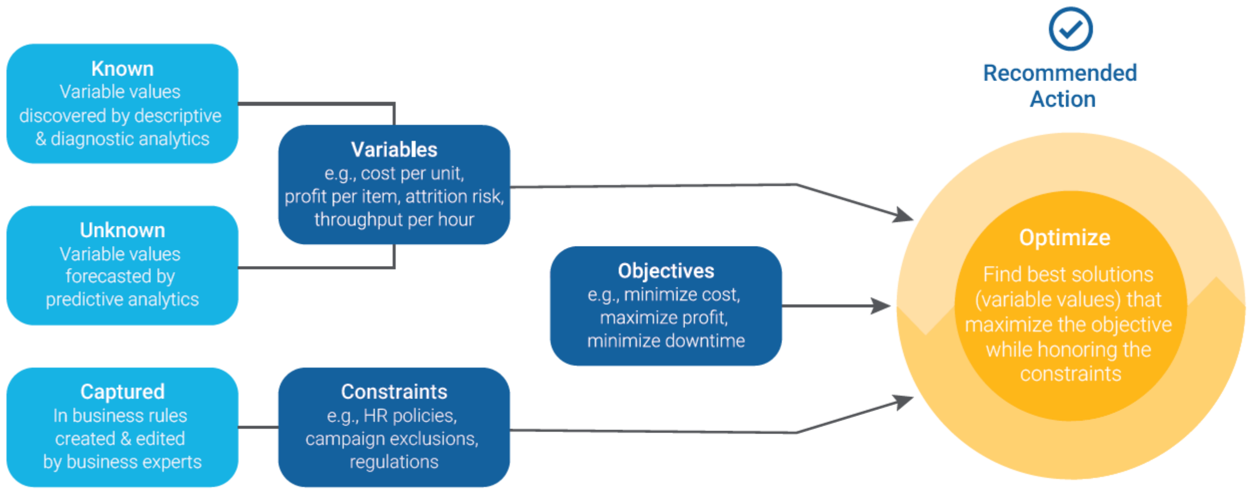

Known
Variable values discovered by descriptive & diagnostic analytics
Unknown
Variable values forecasted by predictive analytics
Captured
In business rules created & edited by business experts
Variables
e.g., cost per unit, profit per item, attrition risk, throughput per hour
Objectives
e.g., minimize cost, maximize profit, minimize downtime
Constraints
e.g., HR policies, campaign exclusions, regulations
Recommended Action
Optimize
Find best solutions (variable values) that maximize the objective while honoring the constraints

Figure 4.2-1 Mathematical Optimization components

4.3 Optimization Problem Types

There are many types of optimization problems stated below:

- Certain problems are constrained, while others are not.
- One or more variables may be provided.
- Variables might be continuous or discrete (for example, only have integer values).
- Some problems are static (don't change over time), whilst others are dynamic (need constant adjustments as changes take place).

- Systems can be stochastic (involve randomness/probability) or deterministic (particular causes produce specified consequences).
- Equations can be nonlinear or linear (graph to lines) (graph to curves)

If we want to declare why mathematical optimization are important, we need to consider the below:

- Mathematical optimization outperforms conventional "guess-and-check" techniques in many situations.
- It is far less expensive to use mathematical optimization than to develop and test
- Microseconds, microns, and cents all matter in the modern world.

4.4 Rush Mathematical Optimization

4.4.1 Problem Overview

We wanted to create a mathematical model based on our database to help us adequately address the constraints of the optimization problem in order to minimize the crowd in the Metro transportation systems.

Our optimization problem's goal is to minimize the metro crowd based on our database, and we're doing our best to take all of our restrictions into account.

Using the maximum capacity per hour of the Metro trains, we may calculate the optimization problem constraints. Each train has a limited capacity, and we have a limitation on the number of trains that can be trailed in a specific hour. We can understand that each Metro line has unique peculiarities by taking rush hour into consideration.

For the tailing time we will find that there is a MIN trailing time that Metro Administration operate during the rush times, and MAX trailing time that they operate during the normal times and holidays.

Table 4.4.1-1 Trailing time

Lines	**MAX Trailing time**	**MIN Trailing Time**	**AVG Trailing Time**
The First Line	360 Seconds	140 Seconds	250 Seconds
The Second Line	360 Seconds	165 Seconds	262.5 Seconds
The Third Line	420 Seconds	300 Seconds	360 Seconds

We also have a specific capacity for each line as they are not the same, the train in the first line consists of 9 carriages, for the second line consists of 8 carriages, and the third line train consists of 12 carriages.

Table 4.4.4-2 number of carriages

Lines	**No. of Carriages**	**MAX Capacity**
The First Line	9 Carriages	2700 Passengers
The Second Line	8 Carriages	2400 Passengers
The Third Line	12 Carriages	3600 Passengers

If we want to calculate the number of trains per hour, we need to consider whether we are in rush hour or not. For constructing our Mathematical Model constraints, we will use both the MAX possible number of trains and the MIN possible number of trains.

The mathematical model’s decision variables:

Table 4.4.4-3 Mathematical Optimization

Variable	**Description**
Z	The number of riders in all Metro lines.
$\mathbf{X_1}$	The number of riders in the first line
$\mathbf{X_2}$	The number of riders in the second line
$\mathbf{X_3}$	The number of riders in the third line

4.4.2 Considerations

We need to consider the MAX/MIN number of population that can be in the Meto line in hour.

- For the First line:
    - As per the MIN trailing time from the Metro Administration we have every 140 seconds a new train which means we have about 26 trains every hour, so that the total possible MAX riders in Metro should be 70200 riders.
    - As per the MAX trailing time from the Metro Administration we have every 360 seconds a new train which means that we have about 10 trains every hour, so that the total possible MIN riders in Metro should be 27000 riders.
- For the Second line:
    - As per the MIN trailing time from the Metro Administration we have every 165 seconds a new train which means we have about 21 trains every hour, so that the total possible MAX riders in Metro should be 50400 riders.
    - As per the MAX trailing time from the Metro Administration we have every 360 seconds a new train which means that we have about 10 trains every hour, so that the total possible MIN riders in Metro should be 24000 riders.
- For the Third line:
    - As per the MIN trailing time from the Metro Administration we have every 300 seconds a new train which means we have about 12 trains every hour, so that the total possible MAX riders in Metro should be 43200 riders.
    - As per the MAX trailing time from the Metro Administration we have every 420 seconds a new train which means that we have about 8 trains every hour, so that the total possible MIN riders in Metro should be 28800 riders.

We have to consider that the Metro lines have intersections in between so that we should take intersections into consideration as it is affecting the total number of riders in each line.

4.4.3 Model Calculations

The mathematical model’s objective function:

$$\text{MIN } Z = X_1 + X_2 + X_3$$

The mathematical model’s constraints:

Based on the above information and calculations we can construct four constraints, every constraint’s corresponding value appeared based on calculations will be discussed in details below.

- The first constraint: We have mentioned before that there’s intersections between the first line and the second line in two intersectional stations (Sadat, and El-Shohadaa), so that we need to construct a constraint to consider the intersectional stations and in the same time consider the MAX possible number of riders, and the MIN trailing time: We will aggregate the number of riders in the first line and the second line together then multiply them with 90% so that there’s an upper boundary less than the MAX possible riders.

(Rush)

$$(50400 + 70200) * 0.9 = 108540$$

$$X_1 + X_2 <= 108540$$

- The second constraint is the same as the first one but for the second line and the third. We will consider the intersectional station in between (Attaaba).

(Rush)

$$(50400 + 43200) * 0.9 = 84240$$

$$X_2 + X_3 <= 84240$$

- The third constraint will get advantage from the MAX trialing time to consider the lower boundary and we should multiply the number that has been calculated using the aggregation equation by 25% as we don’t want the Metro trains to be empty.

(Normal)

$$(27000 + 24000) * 0.25 = 12750$$

$$X_1 + X_2 >= 12750$$

- The fourth constraint is the same the second but for the third and second line.

(Normal)

$$(28800 + 24000) * 0.25 = 13200$$

$$X_2 + X_3 >= 13200$$

4.4.4 TORA Solutions

4.4.4.1 TORA Overview

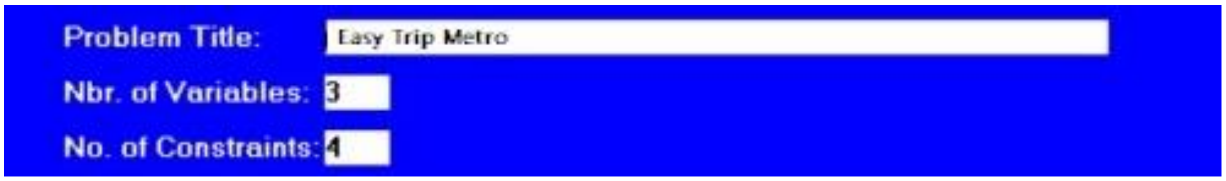
The TORA Package is a set of software tools for statistical calculation and analysis. It is an existing software or software application created for statistical applications. Operations Research (OR) analysis is essentially where it is applied. An interface that is graphical is called TORA Optimization Window (GUI). This distinguishes it from other statistical software programs that include spreadsheet windows.

Several of the methods listed below can be used using the Windows-based TORA Optimization System:

4.4.4.2 Mathematical Model Solution with TORA

We used Tora for solving our Linear Mathematical Model. We will illustrate the solution in steps below:

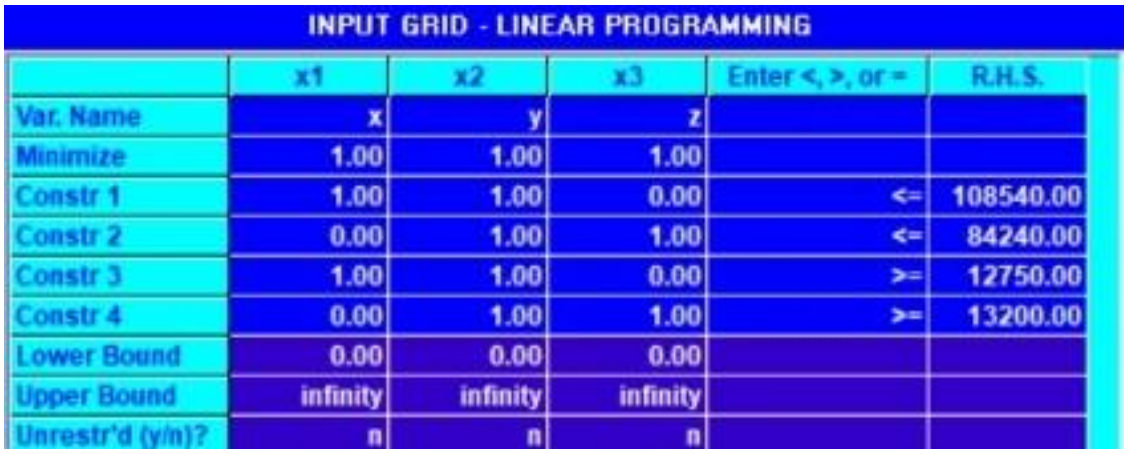
- The first step that we entered our project name, number of Variables, and number of Constraints.



Problem Title: Easy Trip Metro
Nbr. of Variables: 3
No. of Constraints: 4

Figure 4.4.4.2-1 Tora Sol-1

- The second step is the formulation step by adding the values of the Objective function and the constraints.

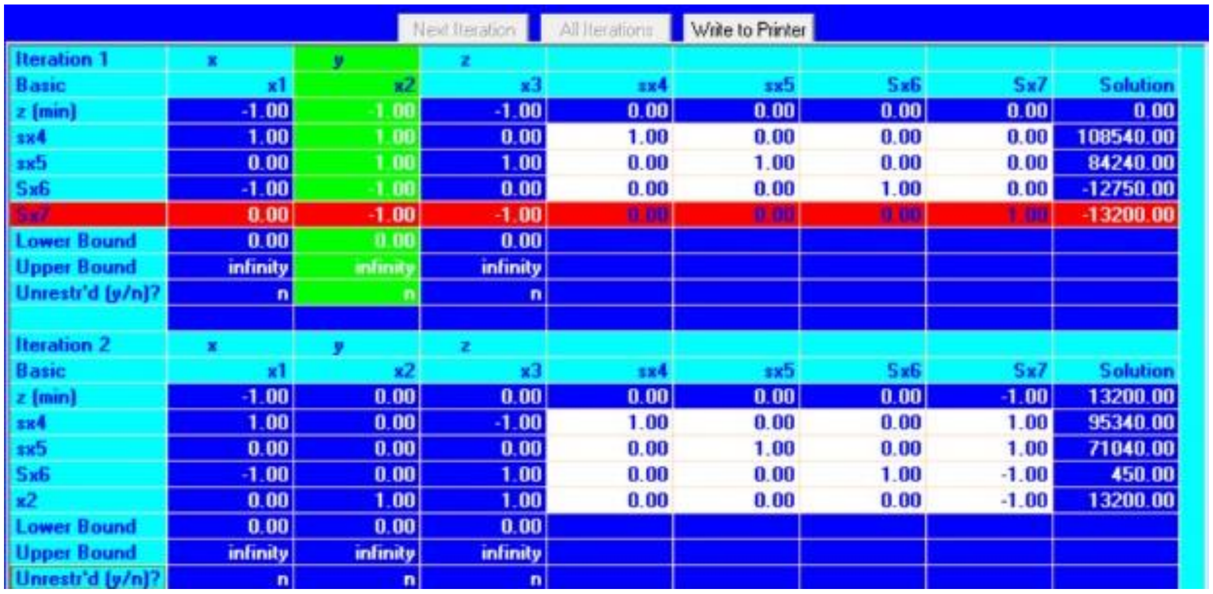


INPUT GRID - LINEAR PROGRAMMING

	x1	x2	x3	Enter <, >, or =	R.H.S.
Var. Name	x	y	z		
Minimize	1.00	1.00	1.00		
Constr 1	1.00	1.00	0.00	<=	108540.00
Constr 2	0.00	1.00	1.00	<=	84240.00
Constr 3	1.00	1.00	0.00	>=	12750.00
Constr 4	0.00	1.00	1.00	>=	13200.00
Lower Bound	0.00	0.00	0.00		
Upper Bound	infinity	infinity	infinity		
Unrestr'd (y/n)?	n	n	n		

Figure 4.4.4.2-2 Tora Sol-2

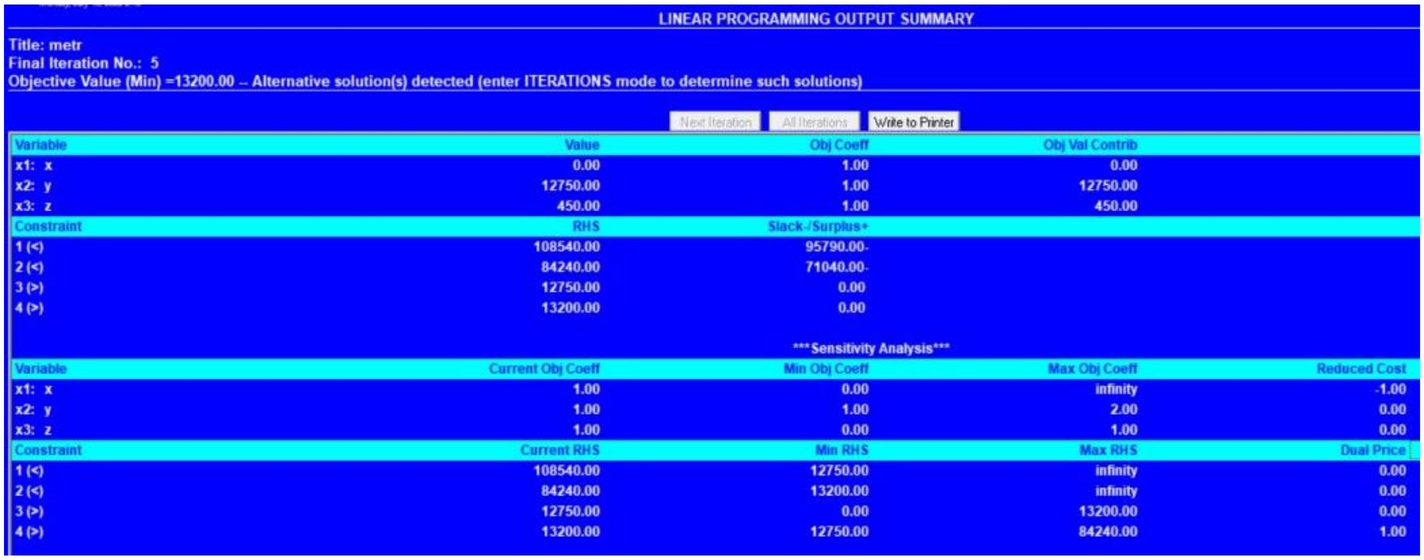
- The third step is running the program, then it will provide the iteration



Iteration 1	x	y	z					
Basic	x1	x2	x3	sx4	sx5	Sx6	Sx7	Solution
z (min)	-1.00	1.00	-1.00	0.00	0.00	0.00	0.00	0.00
sx4	1.00	1.00	0.00	1.00	0.00	0.00	0.00	108540.00
sx5	0.00	1.00	1.00	0.00	1.00	0.00	0.00	84240.00
Sx6	-1.00	-1.00	0.00	0.00	0.00	1.00	0.00	-12750.00
Sx7	0.00	-1.00	-1.00	0.00	0.00	0.00	1.00	-13200.00
Lower Bound	0.00	0.00	0.00					
Upper Bound	infinity	infinity	infinity					
Unrestr'd (y/n)?	n	n	n					
Iteration 2	x	y	z					
Basic	x1	x2	x3	sx4	sx5	Sx6	Sx7	Solution
z (min)	-1.00	0.00	0.00	0.00	0.00	0.00	-1.00	13200.00
sx4	1.00	0.00	-1.00	1.00	0.00	0.00	1.00	95340.00
sx5	0.00	0.00	0.00	0.00	1.00	0.00	1.00	71040.00
Sx6	-1.00	0.00	1.00	0.00	0.00	1.00	-1.00	450.00
x2	0.00	1.00	1.00	0.00	0.00	0.00	-1.00	13200.00
Lower Bound	0.00	0.00	0.00					
Upper Bound	infinity	infinity	infinity					
Unrestr'd (y/n)?	n	n	n					

Figure 4.4.4.2-3 Tora Sol-3

- The last step we will find the solutions table as below



LINEAR PROGRAMMING OUTPUT SUMMARY

Title: metr
Final Iteration No.: 5
Objective Value (Min) =13200.00 -- Alternative solution(s) detected (enter ITERATIONS mode to determine such solutions)

Variable	Value	Obj Coeff	Obj Val Contrib
x1: x	0.00	1.00	0.00
x2: y	12750.00	1.00	12750.00
x3: z	450.00	1.00	450.00

Constraint	RHS	Slack-/Surplus+
1 (<)	108540.00	95790.00-
2 (<)	84240.00	71040.00-
3 (>)	12750.00	0.00
4 (>)	13200.00	0.00

Sensitivity Analysis

Variable	Current Obj Coeff	Min Obj Coeff	Max Obj Coeff	Reduced Cost
x1: x	1.00	0.00	infinity	-1.00
x2: y	1.00	1.00	2.00	0.00
x3: z	1.00	0.00	1.00	0.00

Constraint	Current RHS	Min RHS	Max RHS	Dual Price
1 (<)	108540.00	12750.00	infinity	0.00
2 (<)	84240.00	13200.00	infinity	0.00
3 (>)	12750.00	0.00	13200.00	0.00
4 (>)	13200.00	12750.00	84240.00	1.00

Figure 4.4.4.2-4 Tora Sol-4

4.4.5 MATLAB solutions

4.4.5.1 MATLAB Overview

Matrix laboratory is referred to as MATLAB. This technical computing's high-performance language is called MATLAB. It is a piece of MathWorks' work. In 1984, Cleve Moler, Jack Little, and Steve Bangert established the MathWorks company. Control design engineers were the ones that first improved MATLAB. It swiftly spread to the fields of signal and image processing. The teaching of linear algebra and numerical methods now makes considerable use of it in education.

Figure 4.4.5.1-1 MATLAB Logo

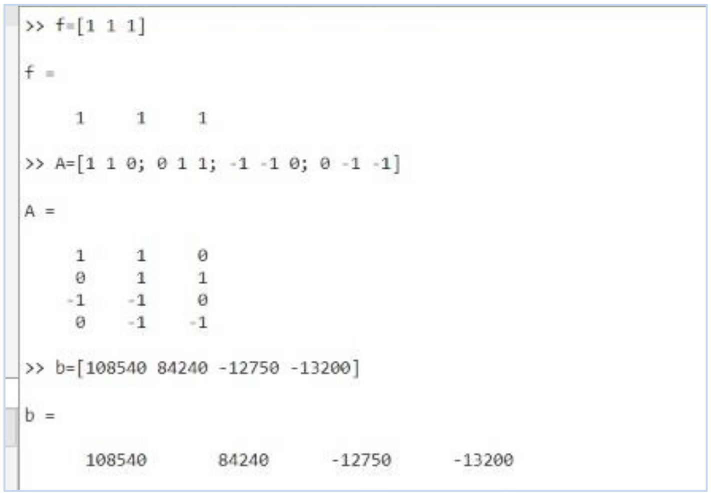
MATLAB Advantages:

- You can quickly implement and test your algorithms.
- quickly create the computational codes
- Easy to debug
- Utilize a big database of integrated algorithms
- Create simulations videos and simply process still pictures.
- It is simple to perform symbolic computation.
- User is able to use both internal and external libraries
- comprehensively analyze and visualize your data
- Make a graphical user interface for your program.

4.4.5.2 Mathematical Model Solution with MATLAB

We used Tora for solving our Linear Mathematical Model. We will illustrate the solution in steps below:

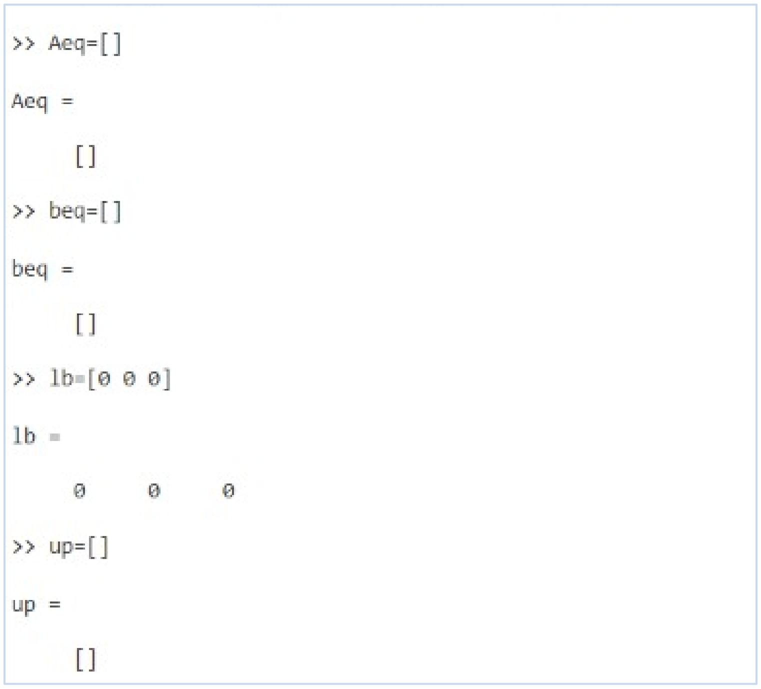
- The first step is to construct the Mathematical Model Formulation by adding the below values for coefficients and the corresponding values for the RHS.



```
>> f=[1 1 1]

f =

     1     1     1

>> A=[1 1 0; 0 1 1; -1 -1 0; 0 -1 -1]

A =

     1     1     0
     0     1     1
    -1    -1     0
     0    -1    -1

>> b=[108540 84240 -12750 -13200]

b =

      108540       84240      -12750      -13200
```

Figure 4.4.5.2-1 MATLAB Sol-1

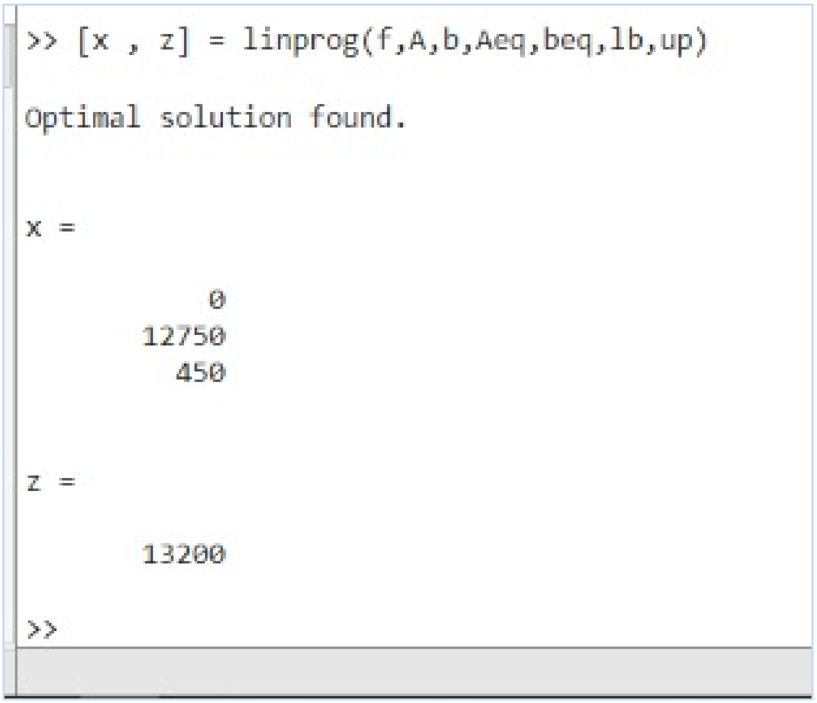
- Then adding the below



```
>> Aeq=[]

Aeq =

     []

>> beq=[]

beq =

     []

>> lb=[0 0 0]

lb =

     0     0     0

>> up=[]

up =

     []
```

Figure 4.4.5.2-2 MATLAB Sol-4

- The final step that we will run using the below function, then the solution as the below.



```
>> [x , z] = linprog(f,A,b,Aeq,beq,lb,up)

Optimal solution found.


x =

           0
       12750
         450


z =

       13200

>>
```

Figure 4.4.5.2-3 MATLAB Sol-5

CHAPTER 5

Database and Software Architecture & Design

5.1 Class Diagram

Easy Trip Application's class diagram

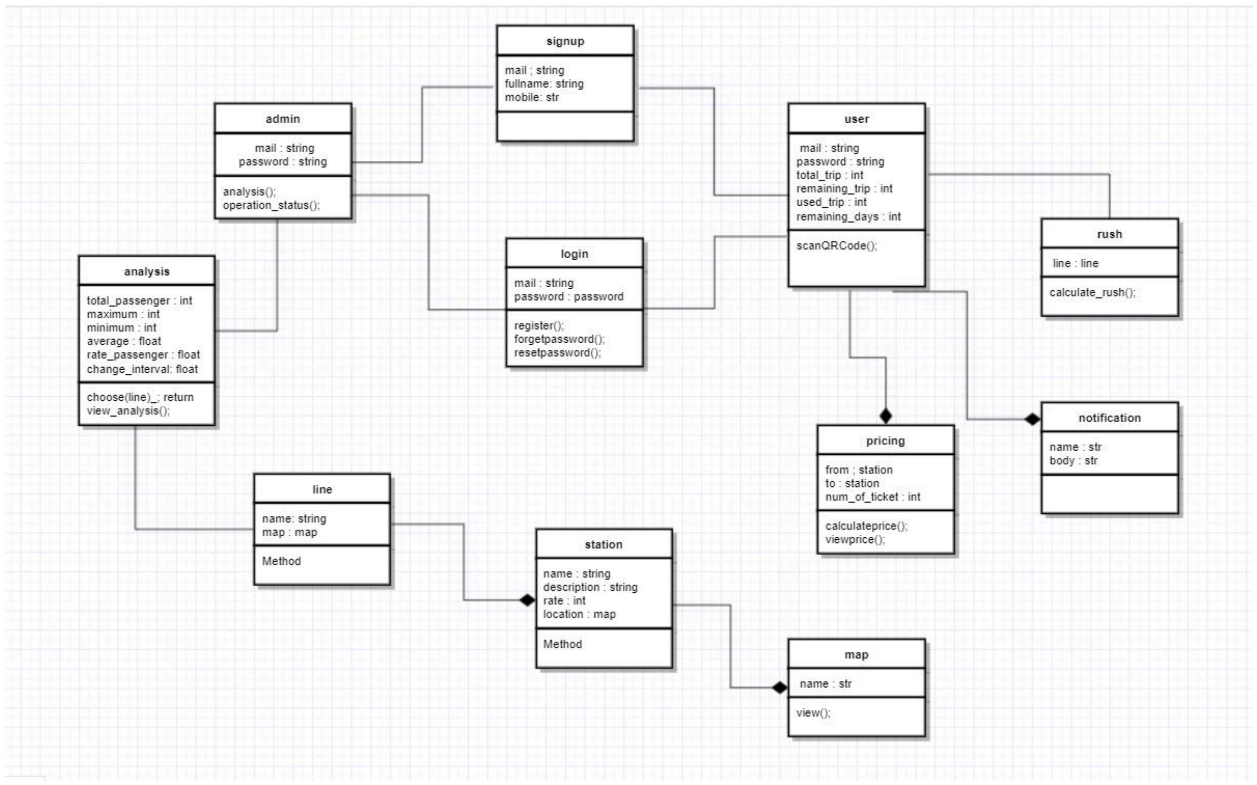


Figure 5.1-1 Class Diagram

5.2 State Transition Diagram

User’s state transition diagram for the sign-up process

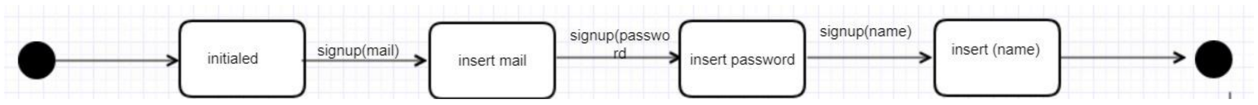


Figure 5.2-1 STM Sign-up Process

User’s state transition diagram for previewing stations map

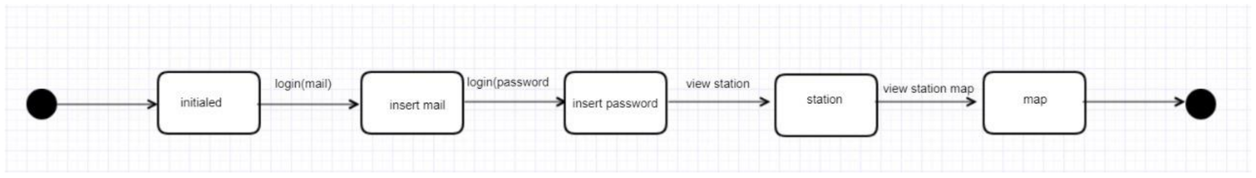


Figure 5.2-2 STM Map Previewing

User’s state transition diagram for checking ticket pricing

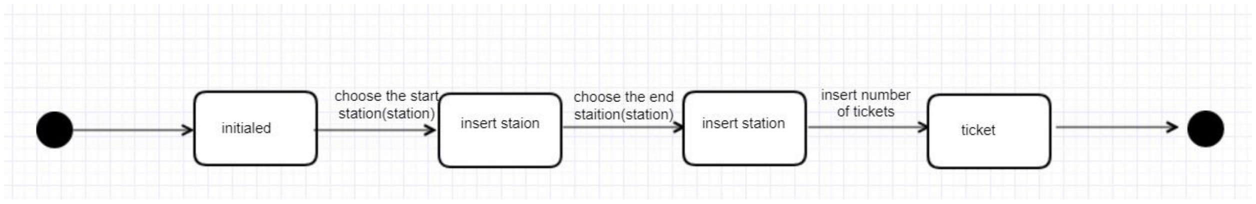


Figure 5.2-3 STM Ticket Pricing

Admin’s state transition diagram for sending the custom notification

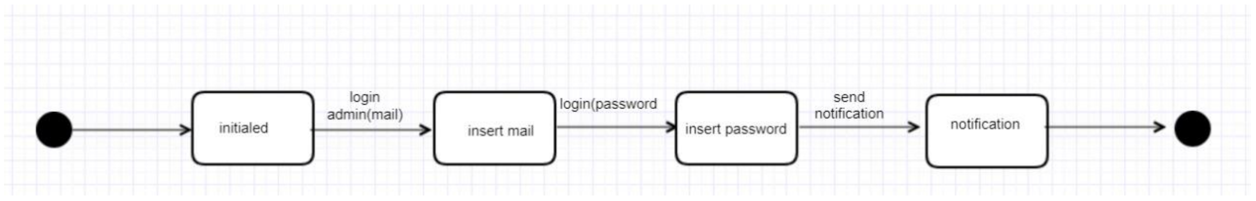


Figure 5.2-4 STM Sending Notification

Admin’s state transition diagram for the analysis process

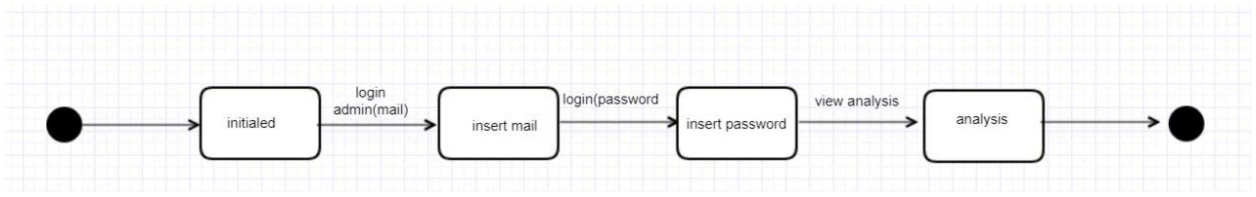


Figure 5.2-5 STM Analysis Process

5.3 UML Sequence Diagram

User’s UML sequence diagram for the log-in and sign-up.

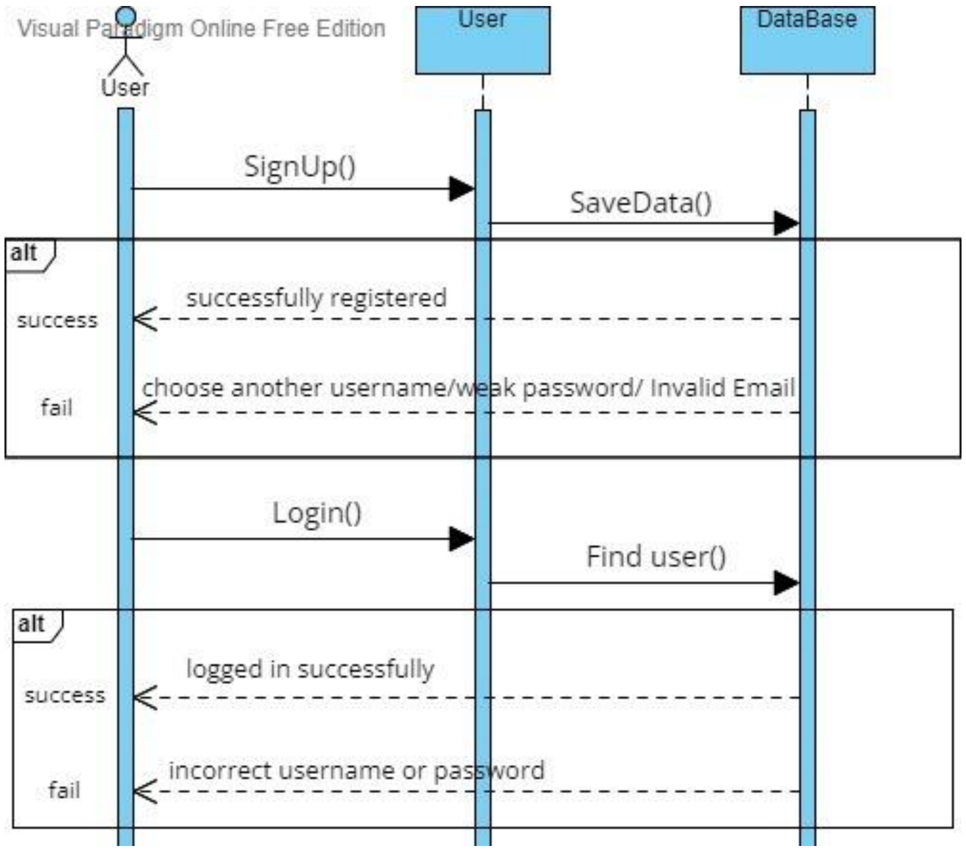


Figure 5.3-1 UML Sequence diagram for login & signup

User's UML Sequence diagram for checking both line rush and station rush.

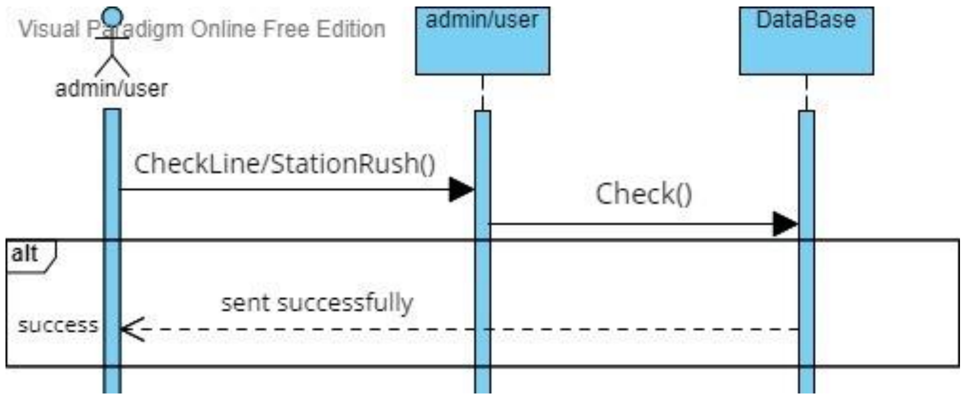


Figure 5.3-2 UML Sequence diagram for checking line rush and station rush

Admin's UML Sequence diagram for both the log-in and add admin processes

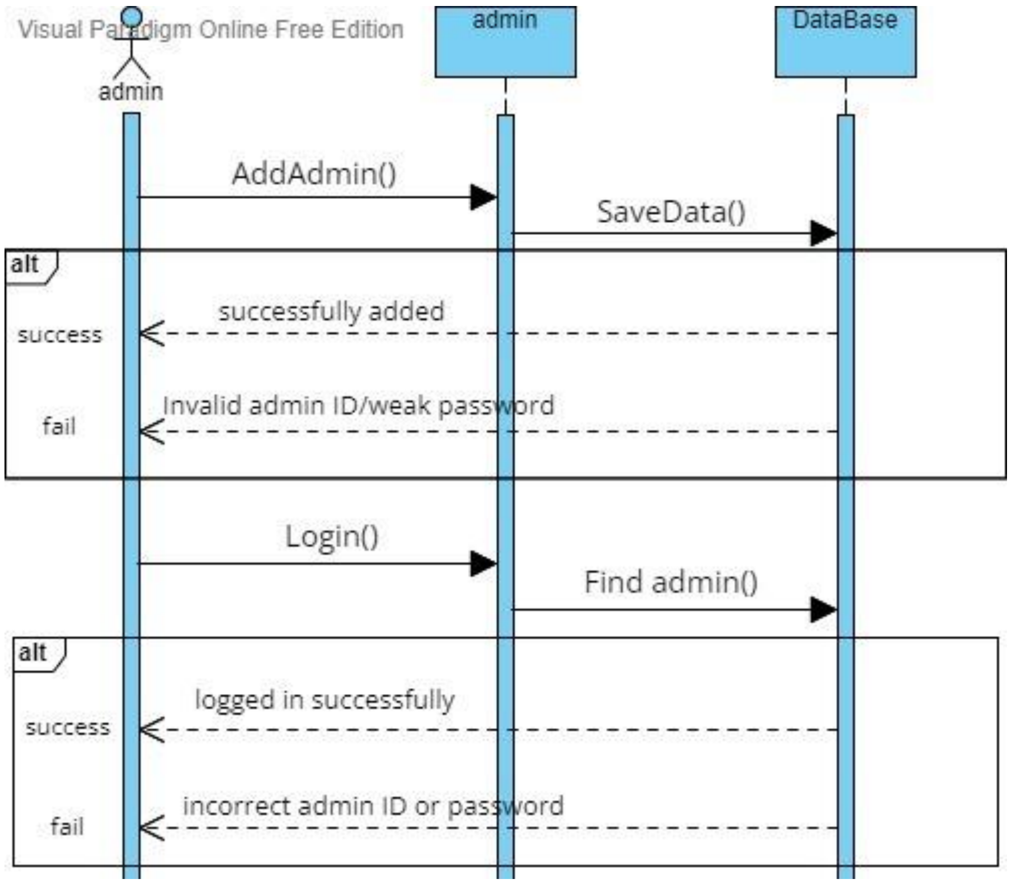


Figure 5.3-3 UML Sequence diagram for admin both login and add admin processes

Admin’s UML Sequence diagram for sending user custom notification

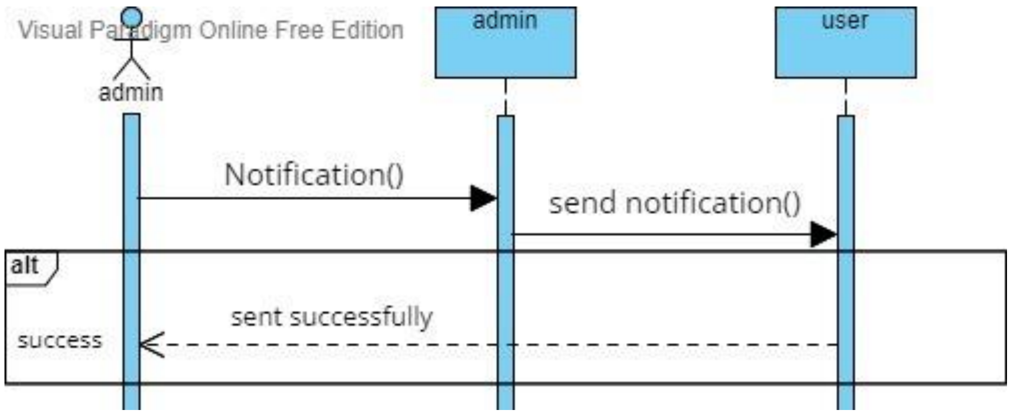


Figure 5.3-4 UML Sequence diagram for sending custom notification

Admin’s UML Sequence diagram for the analysis process

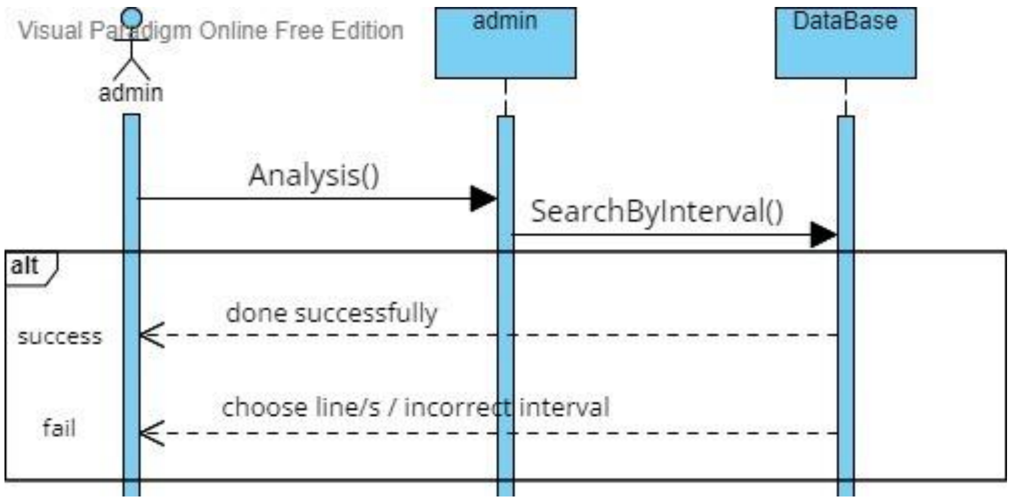


Figure 5.3-5 UML Sequence diagram for the analysis feature

5.4 Database

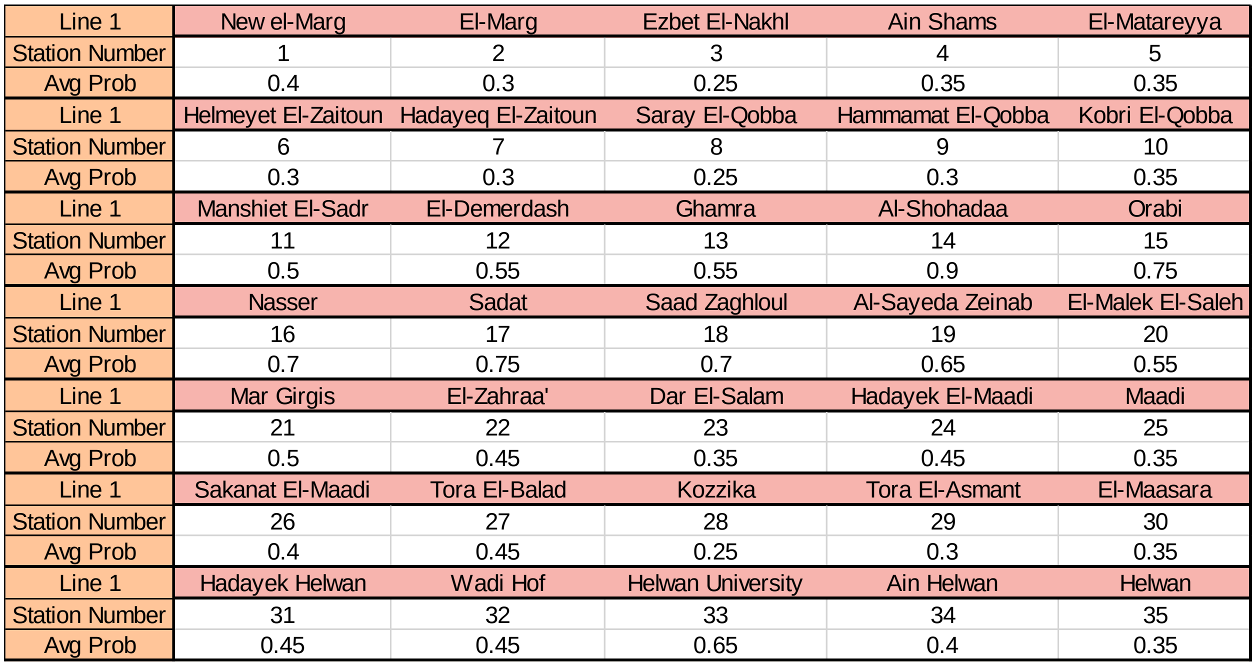
Since we don't yet have access to the Metro infrastructure, it is currently not possible to apply the features we discussed in the earlier chapters because we need a database with Metro details and the instantaneous ridership numbers at each station in order to estimate the probability of the rush status f boorth lines and stations will experience rush hour traffic. Therefore, we needed to build a database to deal with this issue.

5.4.1 Database Creation

A probabilistic database has been developed. In the early stages of the data collection process, we encountered an Engineer from the Metro Administration. We contacted him to obtain a ranking for each station based on its rush to ascertain the rush probabilities interval for it, as well as to take into account the hour's impact on rush percentage.

For each station we determined a probability as per the expertise, we also added a probability for each hour to add it into account.

For the First Line:



Line 1	New el-Marg	El-Marg	Ezbet El-Nakhl	Ain Shams	El-Matareyya
Station Number	1	2	3	4	5
Avg Prob	0.4	0.3	0.25	0.35	0.35
Line 1	Helmeyet El-Zaitoun	Hadayeq El-Zaitoun	Saray El-Qobba	Hammamat El-Qobba	Kobri El-Qobba
Station Number	6	7	8	9	10
Avg Prob	0.3	0.3	0.25	0.3	0.35
Line 1	Manshiet El-Sadr	El-Demerdash	Ghamra	Al-Shohadaa	Orabi
Station Number	11	12	13	14	15
Avg Prob	0.5	0.55	0.55	0.9	0.75
Line 1	Nasser	Sadat	Saad Zaghloul	Al-Sayeda Zeinab	El-Malek El-Saleh
Station Number	16	17	18	19	20
Avg Prob	0.7	0.75	0.7	0.65	0.55
Line 1	Mar Girgis	El-Zahraa'	Dar El-Salam	Hadayek El-Maadi	Maadi
Station Number	21	22	23	24	25
Avg Prob	0.5	0.45	0.35	0.45	0.35
Line 1	Sakanat El-Maadi	Tora El-Balad	Kozzika	Tora El-Asmant	El-Maasara
Station Number	26	27	28	29	30
Avg Prob	0.4	0.45	0.25	0.3	0.35
Line 1	Hadayek Helwan	Wadi Hof	Helwan University	Ain Helwan	Helwan
Station Number	31	32	33	34	35
Avg Prob	0.45	0.45	0.65	0.4	0.35

Figure 5.4.1-1 Line1 prob

The second line:

Line 2	Shubra El-Kheima	Kolleyyet El-Zeraa	Mezallat	Khalafawy	St. Teresa
Station Number	36	37	38	39	40
Avg Prob	0.35	0.4	0.3	0.25	0.15
Line 2	Road El-Farag	Masarra	Attaba	Mohamed Naguib	Opera
Station Number	41	42	43	44	45
Avg Prob	0.4	0.55	1	0.6	0.4
Line 2	Dokki	El Bohoth	Cairo University	Faisal	El Giza
Station Number	46	47	48	49	50
Avg Prob	0.55	0.4	0.6	0.5	0.5
Line 2	Omm El-Masryeen	Sakiat Mekky	El-Mounib		
Station Number	51	52	53		
Avg Prob	0.35	0.3	0.6		

Figure 5.4.1-2 Line2 prob

The third line:

Line 3	Adly Mansour	El-Haykestep	Omar Ibn El-Khattab	Qobaa	Hesham Barakat
Station Number	54	55	56	57	58
Avg Prob	0.4	0.3	0.25	0.3	0.35
Line 3	El-Nozha	Nadi El-Shams	Alf Maskan	Heliopolis Square	Haroun
Station Number	59	60	61	62	63
Avg Perc	0.45	0.4	0.6	0.55	0.5
Line 3	Al-Ahram	Koleyet El-Banat	Stadium	Fair Zone	Abbassia
Station Number	64	65	66	67	68
Avg Prob	0.5	0.5	0.55	0.55	0.7
Line 3	Abdou Pasha	El-Giesh	Bab El-Shaaria		
Station Number	69	70	71		
Avg Prob	0.65	0.65	0.7		

Figure 5.4.1-3 Line3 prob

For each hour average probability:

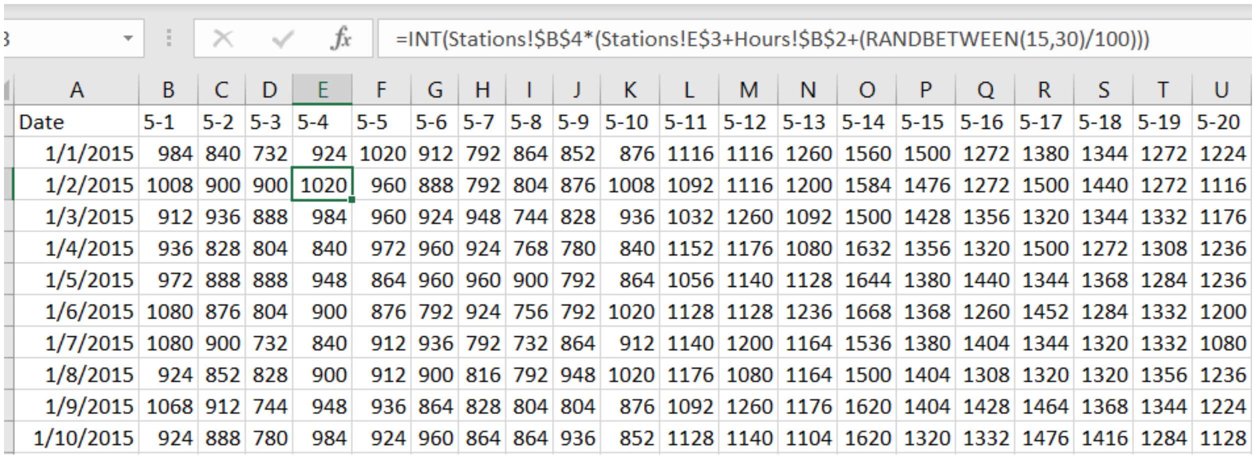
Hour	5:00 AM	6:00 AM	7:00 AM	8:00 AM	9:00 AM	10:00 AM	11:00 AM	12:00 PM	1:00 PM	2:00 PM
AVG prob	0.2	0.4	0.85	0.7	0.65	0.5	0.35	0.3	0.42	0.55
Hour	3:00 PM	4:00 PM	5:00 PM	6:00 PM	7:00 PM	8:00 PM	9:00 PM	10:00 PM	11:00 PM	12:00 AM
AVG prob	0.59	0.8	0.56	0.34	0.3	0.33	0.25	0.2	0.18	0.11

Figure 5.4.1-4 Hour prob

Then we used this probability to generate our database, by considering an average number for all stations as 1200 Passenger, then we multiplied this average population with the average percentage of station's rush and the average probability of hour's rush then we add a random number between 0.15 and 0.3. we added this number to let data change every time.

The rule of creating the database:

*Average population * (average percentage of station's rush + average probability of hour's rush) + (Random number between 0.15 and 0.3)*



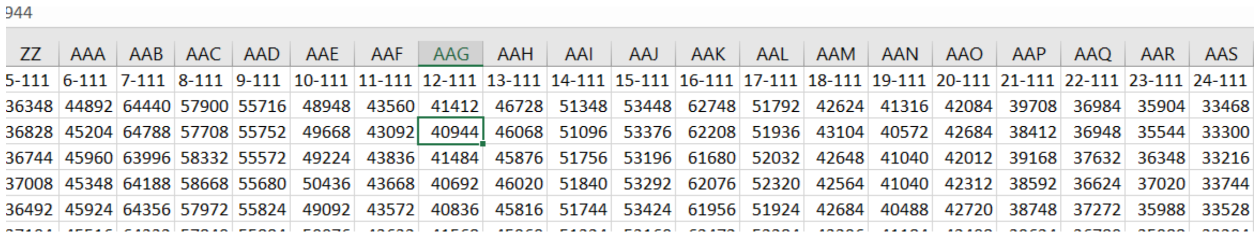
```
=INT(Stations!$B$4*(Stations!E$3+Hours!$B$2+(RANDBETWEEN(15,30)/100)))
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
Date	5-1	5-2	5-3	5-4	5-5	5-6	5-7	5-8	5-9	5-10	5-11	5-12	5-13	5-14	5-15	5-16	5-17	5-18	5-19	5-20
1/1/2015	984	840	732	924	1020	912	792	864	852	876	1116	1116	1260	1560	1500	1272	1380	1344	1272	1224
1/2/2015	1008	900	900	1020	960	888	792	804	876	1008	1092	1116	1200	1584	1476	1272	1500	1440	1272	1116
1/3/2015	912	936	888	984	960	924	948	744	828	936	1032	1260	1092	1500	1428	1356	1320	1344	1332	1176
1/4/2015	936	828	804	840	972	960	924	768	780	840	1152	1176	1080	1632	1356	1320	1500	1272	1308	1236
1/5/2015	972	888	888	948	864	960	960	900	792	864	1056	1140	1128	1644	1380	1440	1344	1368	1284	1236
1/6/2015	1080	876	804	900	876	792	924	756	792	1020	1128	1128	1236	1668	1368	1260	1452	1284	1332	1200
1/7/2015	1080	900	732	840	912	936	792	732	864	912	1140	1200	1164	1536	1380	1404	1344	1320	1332	1080
1/8/2015	924	852	828	900	912	900	816	792	948	1020	1176	1080	1164	1500	1404	1308	1320	1320	1356	1236
1/9/2015	1068	912	744	948	936	864	828	804	804	876	1092	1260	1176	1620	1404	1428	1464	1368	1344	1224
1/10/2015	924	888	780	984	924	960	864	864	936	852	1128	1140	1104	1620	1320	1332	1476	1416	1284	1128

Figure 5.4.1-5 Database rule

We built a database going back seven years. As we can see from the fact that rows are date commencing on January 1, 2015, and cols refers to the hour-station number, our database takes into account the dates and hours for both stations and lines. Each station has an ID, starting with line 1, followed by lines 2, and finally lines 3. Since the Metro runs from 5 AM to 1 AM, there are 20 columns of data for each station covering those 20 hours. There are 71 stations and 20 operating hours, so that we have a total of 1420 columns. Additionally, we also made columns for the overall number of people in each line during a particular hour so that there's additional 60 columns, so the total would be 1481 columns and 2558 rows.

For the lines example we have below the data for the first line (Hour-111)

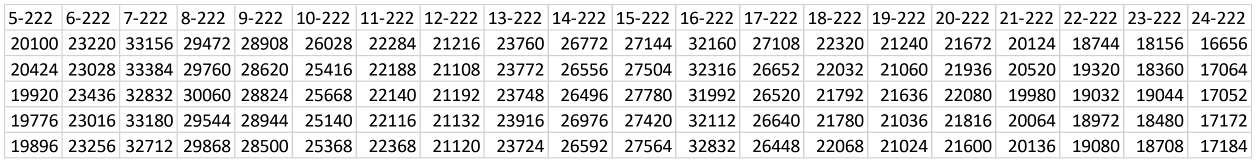


944

ZZ	AAA	AAB	AAC	AAD	AAE	AAF	AAG	AAH	AAI	AAJ	AAK	AAL	AAM	AAN	AAO	AAP	AAQ	AAR	AAS
5-111	6-111	7-111	8-111	9-111	10-111	11-111	12-111	13-111	14-111	15-111	16-111	17-111	18-111	19-111	20-111	21-111	22-111	23-111	24-111
36348	44892	64440	57900	55716	48948	43560	41412	46728	51348	53448	62748	51792	42624	41316	42084	39708	36984	35904	33468
36828	45204	64788	57708	55752	49668	43092	40944	46068	51096	53376	62208	51936	43104	40572	42684	38412	36948	35544	33300
36744	45960	63996	58332	55572	49224	43836	41484	45876	51756	53196	61680	52032	42648	41040	42012	39168	37632	36348	33216
37008	45348	64188	58668	55680	50436	43668	40692	46020	51840	53292	62076	52320	42564	41040	42312	38592	36624	37020	33744
36492	45924	64356	57972	55824	49092	43572	40836	45816	51744	53424	61956	51924	42684	40488	42720	38748	37272	35988	33528

Figure 5.4.1-6 Database line 1

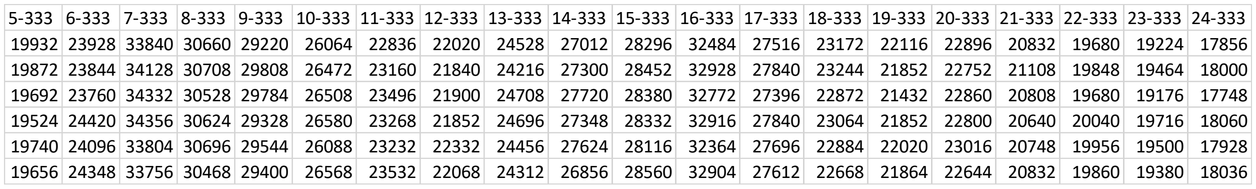
For the second line (Hour-222)



5-222	6-222	7-222	8-222	9-222	10-222	11-222	12-222	13-222	14-222	15-222	16-222	17-222	18-222	19-222	20-222	21-222	22-222	23-222	24-222
20100	23220	33156	29472	28908	26028	22284	21216	23760	26772	27144	32160	27108	22320	21240	21672	20124	18744	18156	16656
20424	23028	33384	29760	28620	25416	22188	21108	23772	26556	27504	32316	26652	22032	21060	21936	20520	19320	18360	17064
19920	23436	32832	30060	28824	25668	22140	21192	23748	26496	27780	31992	26520	21792	21636	22080	19980	19032	19044	17052
19776	23016	33180	29544	28944	25140	22116	21132	23916	26976	27420	32112	26640	21780	21036	21816	20064	18972	18480	17172
19896	23256	32712	29868	28500	25368	22368	21120	23724	26592	27564	32832	26448	22068	21024	21600	20136	19080	18708	17184

Figure 5.4.1-7 Database line 2

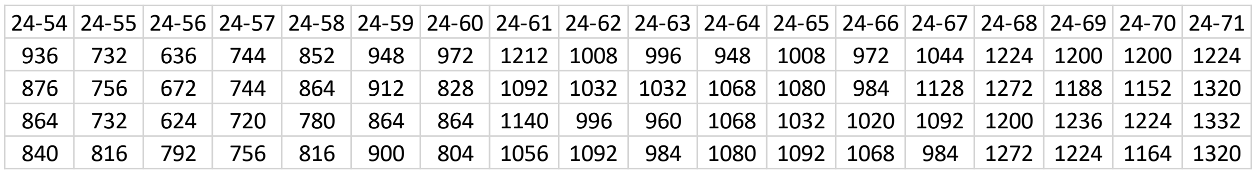
For the third line (Hour-333)



5-333	6-333	7-333	8-333	9-333	10-333	11-333	12-333	13-333	14-333	15-333	16-333	17-333	18-333	19-333	20-333	21-333	22-333	23-333	24-333
19932	23928	33840	30660	29220	26064	22836	22020	24528	27012	28296	32484	27516	23172	22116	22896	20832	19680	19224	17856
19872	23844	34128	30708	29808	26472	23160	21840	24216	27300	28452	32928	27840	23244	21852	22752	21108	19848	19464	18000
19692	23760	34332	30528	29784	26508	23496	21900	24708	27720	28380	32772	27396	22872	21432	22860	20808	19680	19176	17748
19524	24420	34356	30624	29328	26580	23268	21852	24696	27348	28332	32916	27840	23064	21852	22800	20640	20040	19716	18060
19740	24096	33804	30696	29544	26088	23232	22332	24456	27624	28116	32364	27696	22884	22020	23016	20748	19956	19500	17928
19656	24348	33756	30468	29400	26568	23532	22068	24312	26856	28560	32904	27612	22668	21864	22644	20832	19860	19380	18036

Figure 5.4.1-8 Database line 3

Stations data for the third line (Hour-station ID)



24-54	24-55	24-56	24-57	24-58	24-59	24-60	24-61	24-62	24-63	24-64	24-65	24-66	24-67	24-68	24-69	24-70	24-71
936	732	636	744	852	948	972	1212	1008	996	948	1008	972	1044	1224	1200	1200	1224
876	756	672	744	864	912	828	1092	1032	1032	1068	1080	984	1128	1272	1188	1152	1320
864	732	624	720	780	864	864	1140	996	960	1068	1032	1020	1092	1200	1236	1224	1332
840	816	792	756	816	900	804	1056	1092	984	1080	1092	1068	984	1272	1224	1164	1320

Figure 5.4.1-9 Database stations in line 3

Stations data for the second line (Hour-station ID)

5-36	5-37	5-38	5-39	5-40	5-41	5-42	5-43	5-44	5-45	5-46	5-47	5-48	5-49	5-50	5-51	5-52	5-53
948	972	780	828	648	984	1260	1764	1200	960	1236	1008	1284	1152	1128	840	804	1260
888	948	948	900	756	972	1152	1788	1260	936	1152	1044	1224	1056	1200	984	852	1284
948	948	780	732	708	984	1224	1620	1236	1080	1236	984	1140	1104	1080	924	888	1224
960	936	912	732	732	912	1248	1656	1188	984	1092	1032	1140	1164	1068	840	948	1248

Figure 5.4.1-10 Database stations in line 2

Stations data for the first line (Hour-station ID)

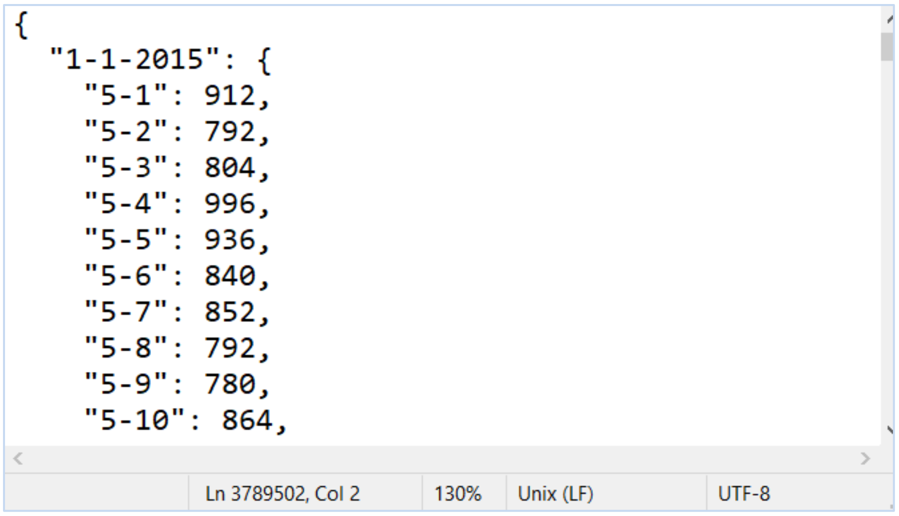
8-1	8-2	8-3	8-4	8-5	8-6	8-7	8-8	8-9	8-10	8-11	8-12	8-13	8-14	8-15	8-16	8-17	8-18	8-19	8-20	8-21	8-22	8-23	8-24	8-25	8-26	8-27	8-28	8-29	8-30	8-31	8-32	8-33	8-34	8-35
1524	1536	1476	1548	1560	1440	1392	1332	1440	1512	1644	1860	1716	2208	1956	1932	1968	1944	1896	1848	1692	1692	1620	1596	1596	1548	1596	1380	1536	1512	1560	1632	1980	1632	1596
1680	1380	1500	1608	1452	1392	1548	1368	1560	1452	1716	1692	1680	2220	1992	2028	1932	1944	1944	1728	1692	1680	1452	1560	1620	1560	1692	1440	1380	1512	1584	1560	1956	1620	1584
1620	1392	1332	1524	1584	1524	1536	1392	1524	1608	1680	1860	1716	2268	1968	2040	2016	1884	1884	1788	1644	1644	1548	1716	1548	1500	1704	1368	1416	1572	1680	1656	1932	1656	1608

Figure 5.4.1-11 Database stations in line 1

5.4.2 Database connection with Firebase

As we are working on developing the application then we needed to convert our Database to a JSON format file to be uploaded to the Firebase to work as our real time database.

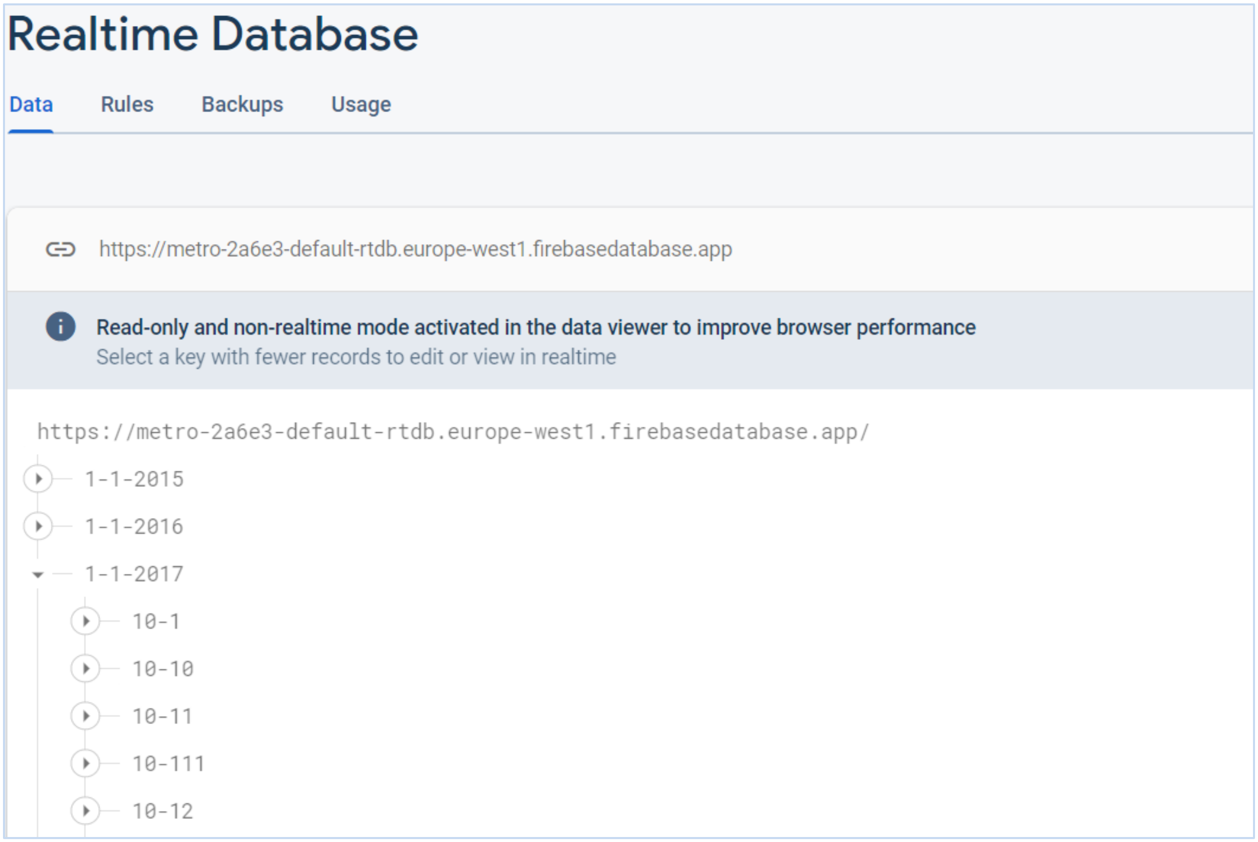
We used python programming language to convert the 3.5 million cells, we grouped every 6 month together then we convert every database of the 14 to JSON then we grouped them together, that was because data was too large to be converted.



```
{
  "1-1-2015": {
    "5-1": 912,
    "5-2": 792,
    "5-3": 804,
    "5-4": 996,
    "5-5": 936,
    "5-6": 840,
    "5-7": 852,
    "5-8": 792,
    "5-9": 780,
    "5-10": 864,
```

Figure 5.4.2-1 Database JSON format

After Database conversion we uploaded our database to the firebase



https://metro-2a6e3-default-rtdb.europe-west1.firebasedatabase.app

Read-only and non-realtime mode activated in the data viewer to improve browser performance
Select a key with fewer records to edit or view in realtime

```
https://metro-2a6e3-default-rtdb.europe-west1.firebasedatabase.app/
 1-1-2015
 1-1-2016
 1-1-2017
   10-1
   10-10
   10-11
   10-111
   10-12
```

Figure 5.4.2-2 Realtime Database JSON

5.5 User Interface

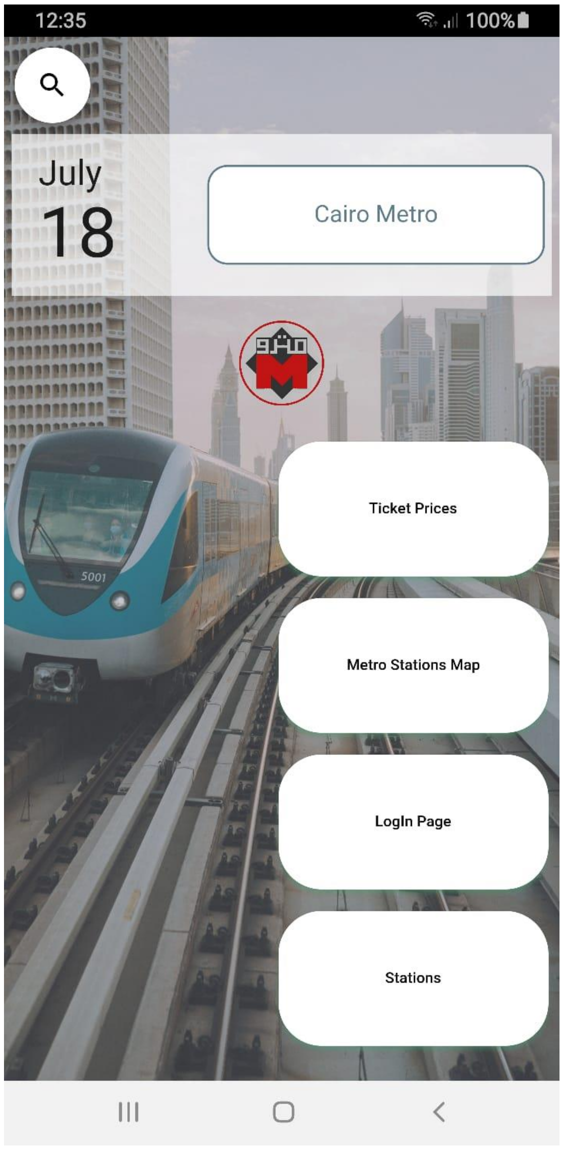


*Figure 5.5-**1 Home Page***

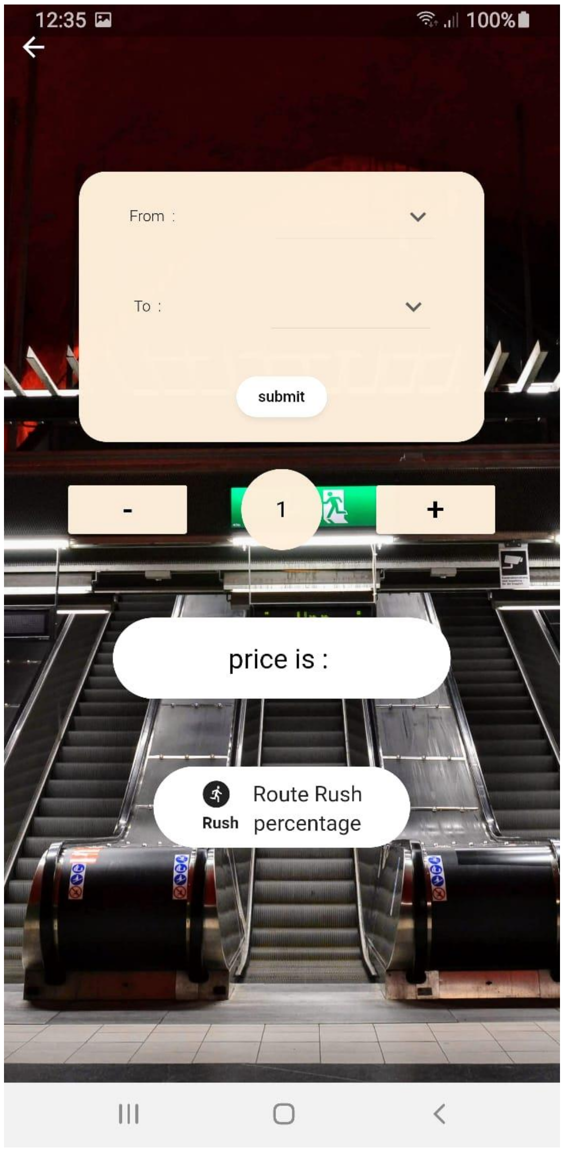
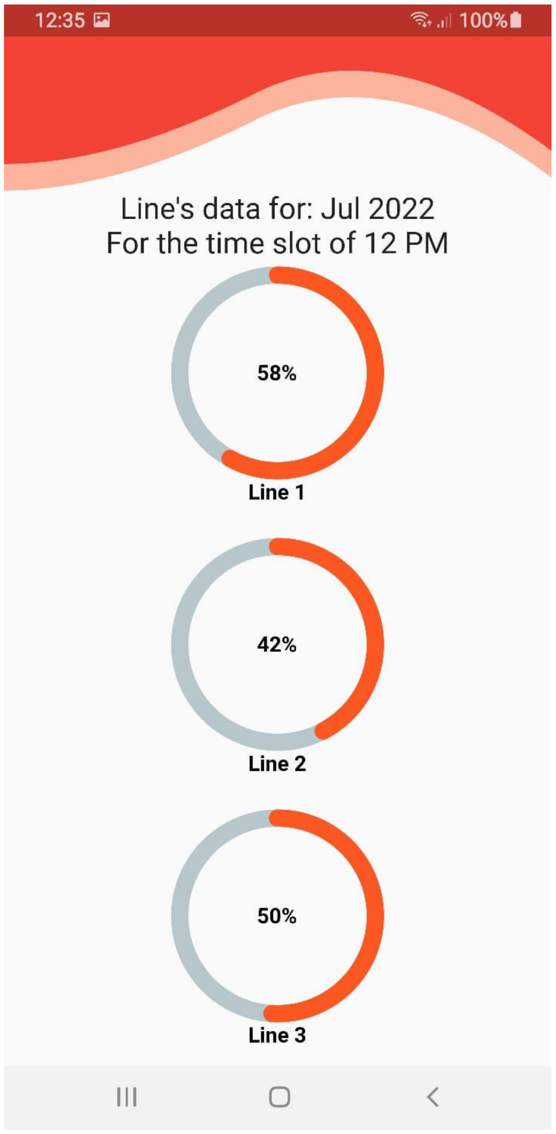


*Figure 5.5-**2 Ticket pricing***



Line's data for: Jul 2022
For the time slot of 12 PM

58%

42%

50%

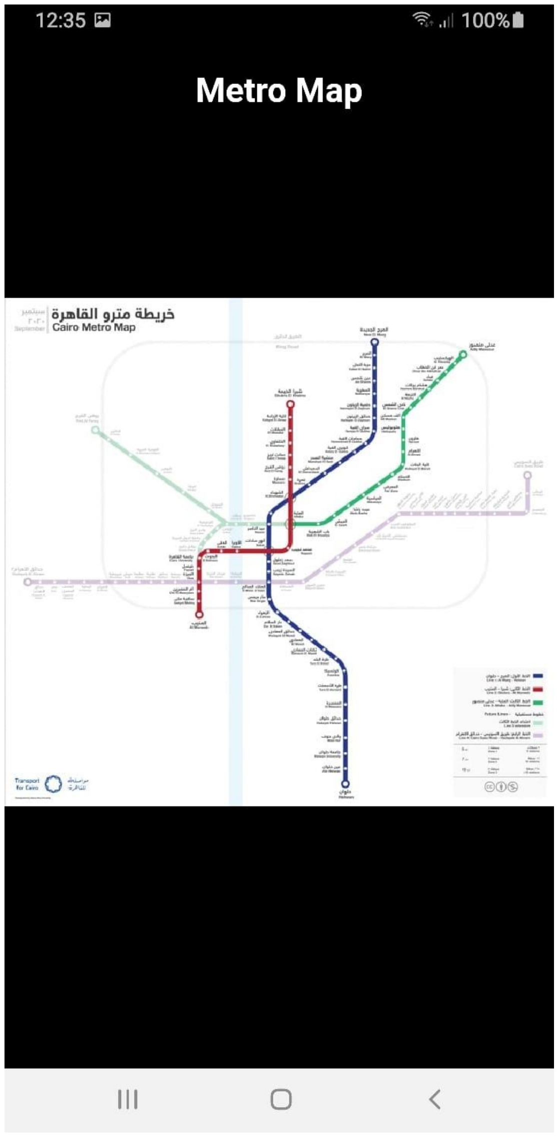


Figure 5.5-3 ***Line Rush***

Figure 5.5-4 ***Metro Map***

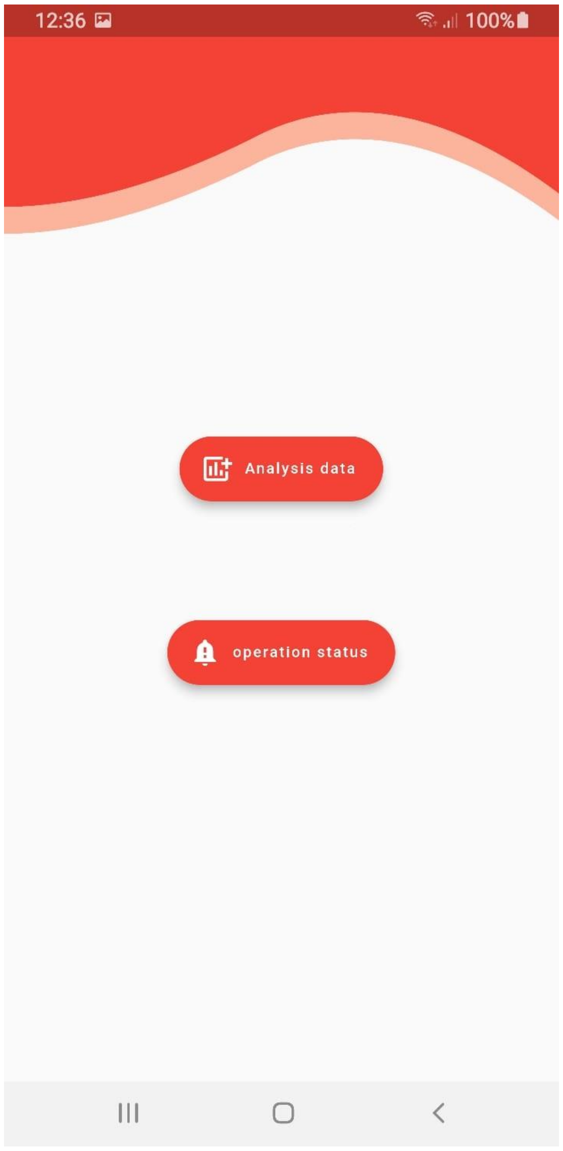


*Figure 5.5-**5 Admin page***

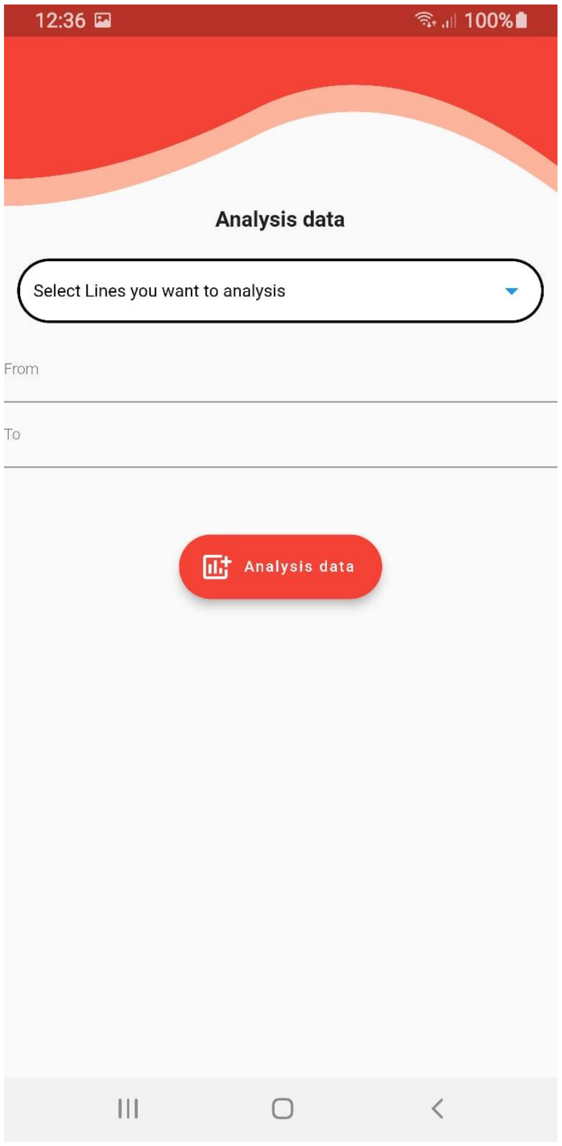
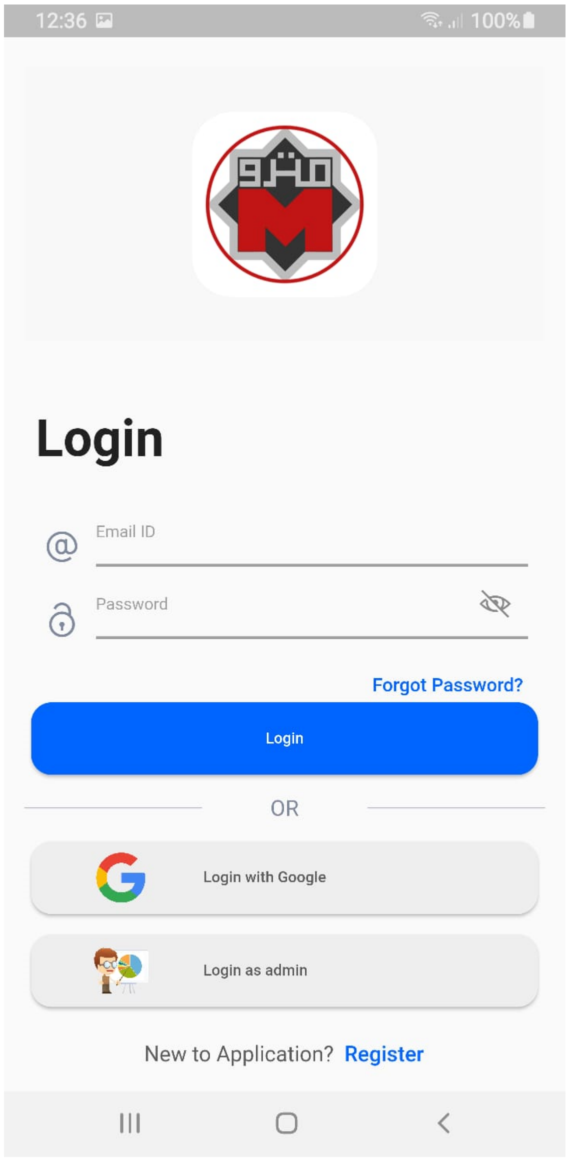


*Figure 5.5-**6 Analysis feature***



12:36 100%

Login

Email ID

Password

Forgot Password?

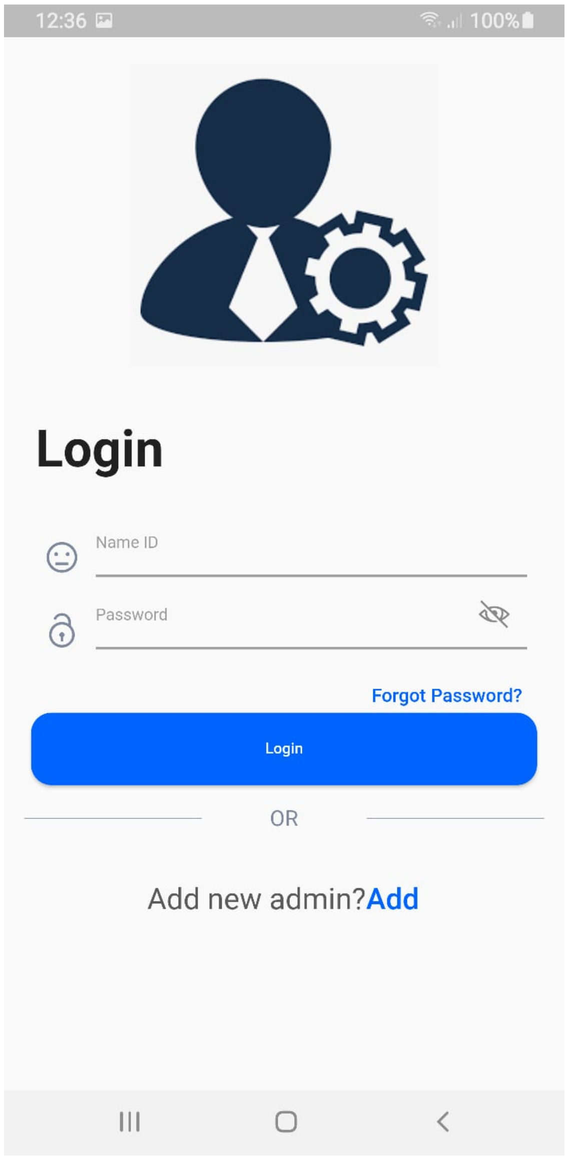
Login

OR

Login with Google

Login as admin

New to Application? Register



12:36 100%

Login

Name ID

Password

Forgot Password?

Login

OR

Add new admin?Add

*Figure 5.5-**7 User login page***

*Figure 5.5-**8 Admin login page***

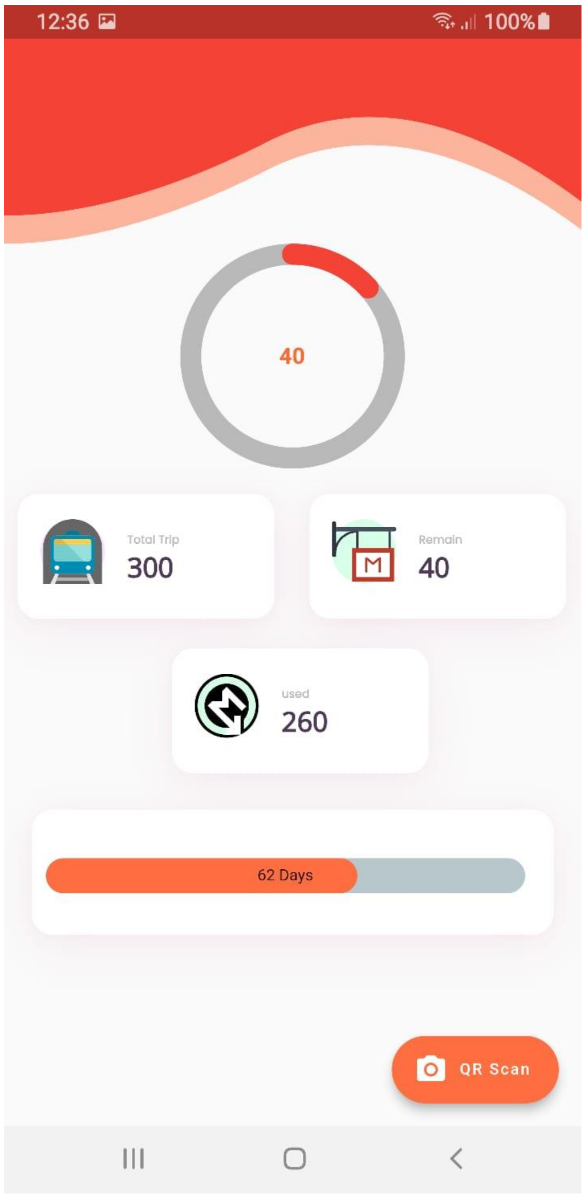


*Figure 5.5-**9 User Balance page***

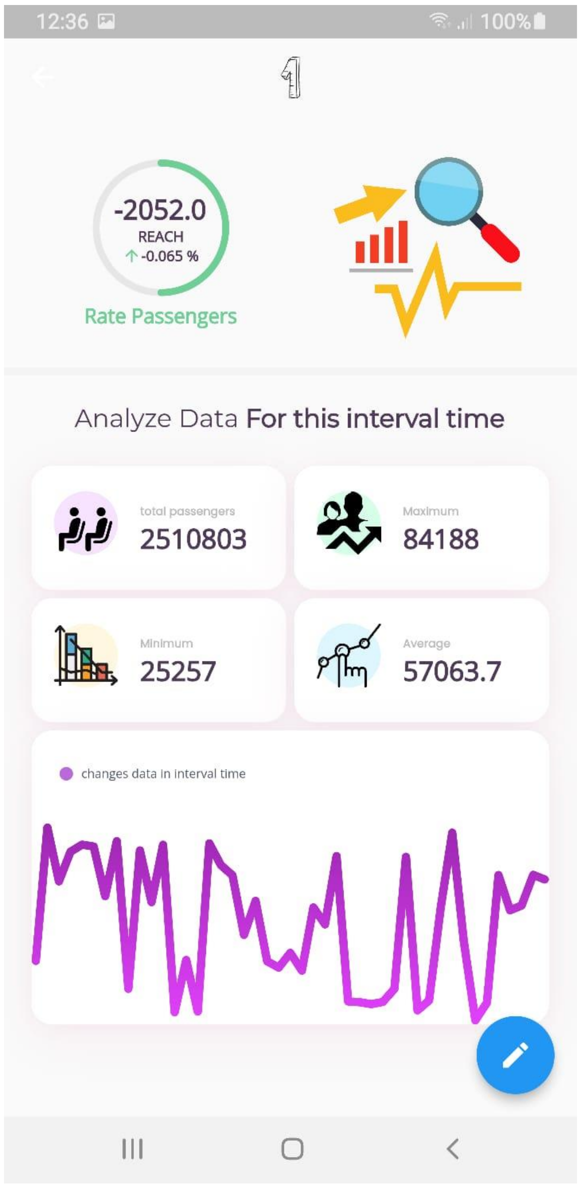


*Figure 5.5-**10 Line analysis page***

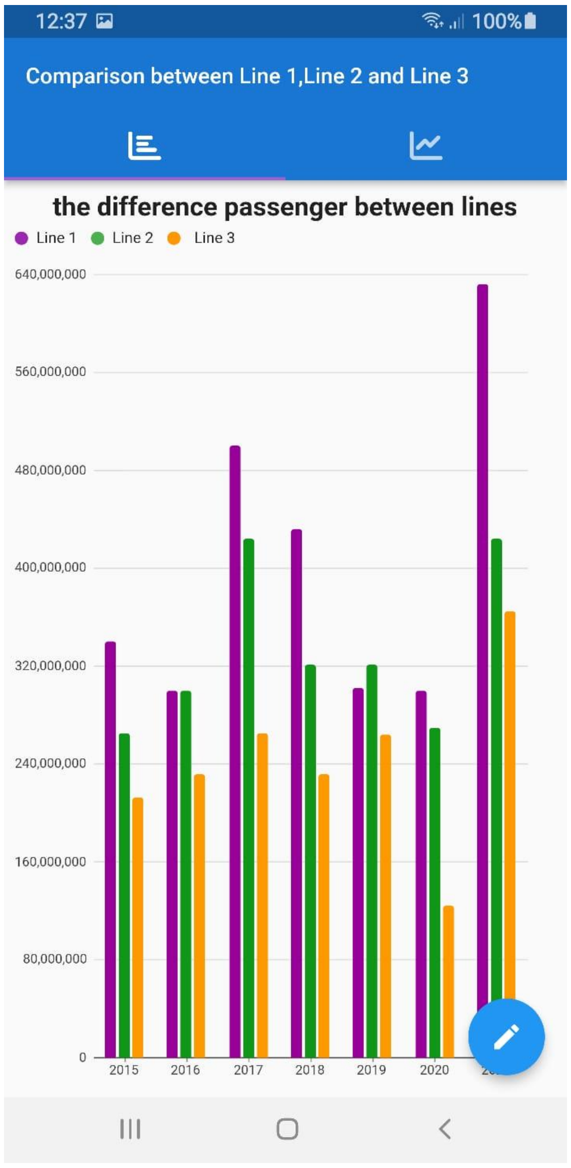


*Figure 5.5-**11 Line analysis bar chart***

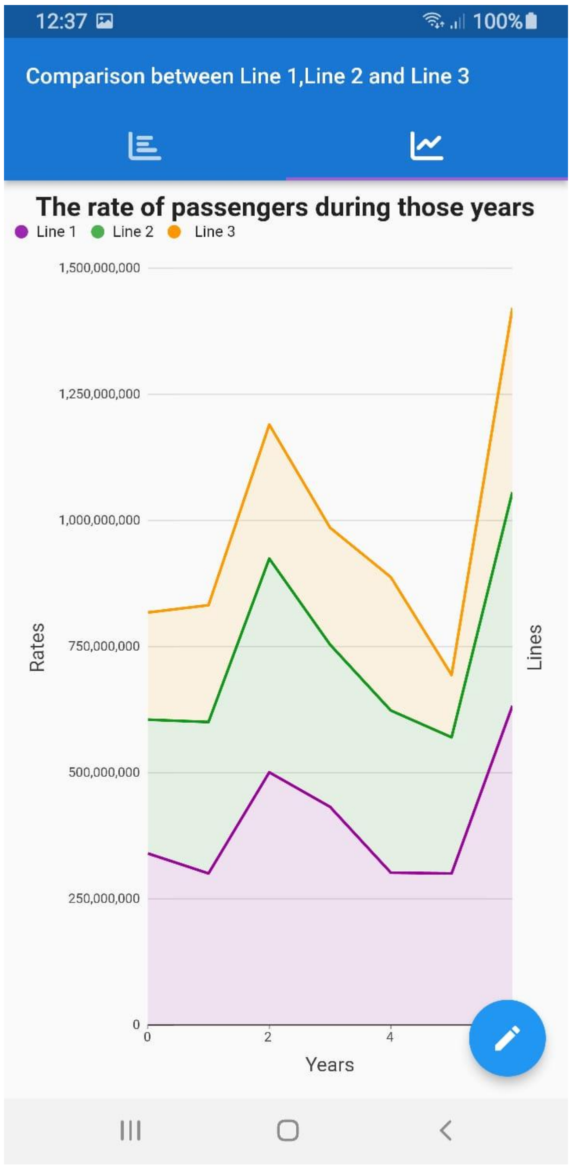


*Figure 5.5-**12 Line analysis time series***

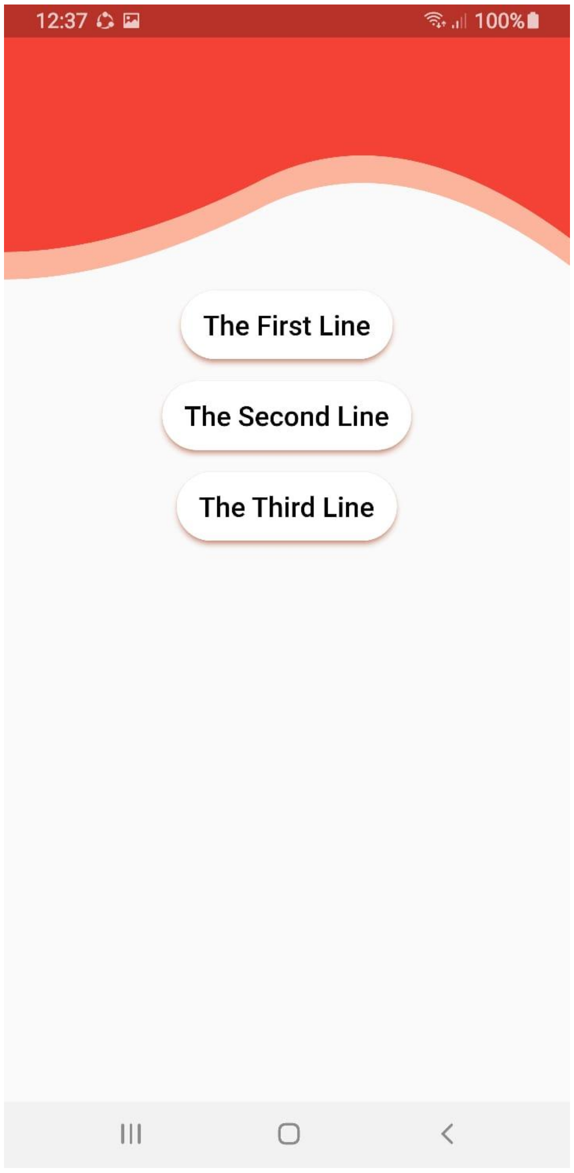
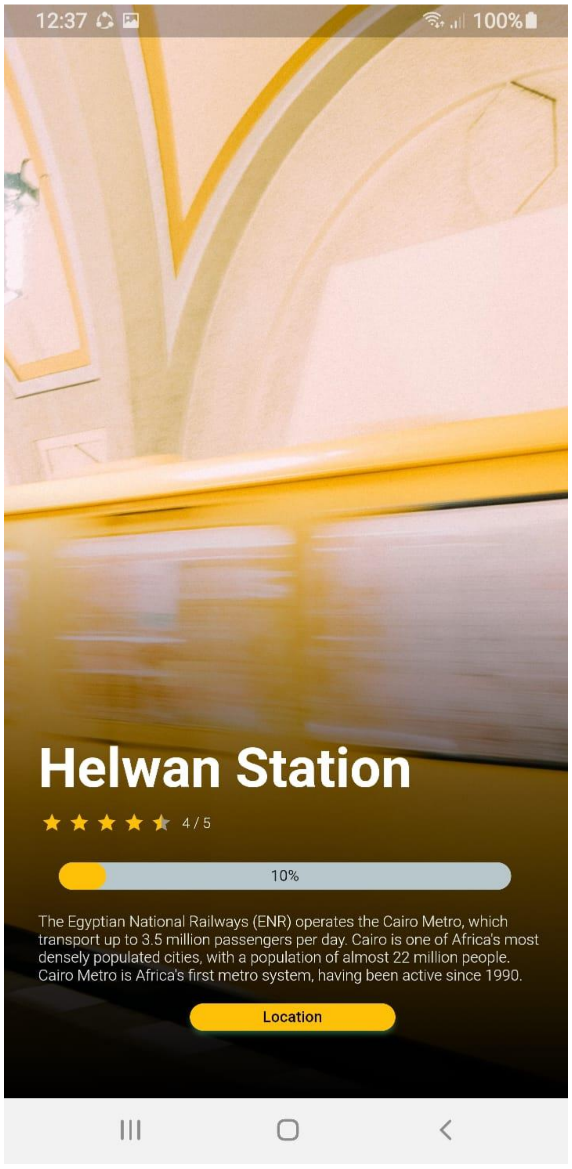


*Figure 5.5-**13 Lines Menu***



The Egyptian National Railways (ENR) operates the Cairo Metro, which transport up to 3.5 million passengers per day. Cairo is one of Africa's most densely populated cities, with a population of almost 22 million people. Cairo Metro is Africa's first metro system, having been active since 1990.

*Figure 5.5-**14 Station pages***

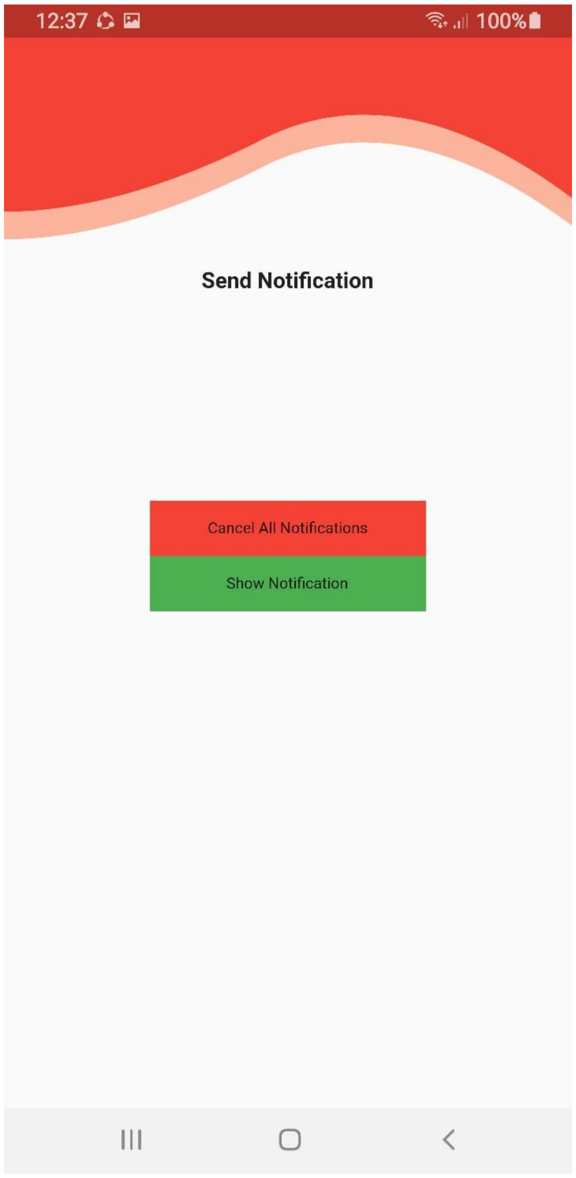


*Figure 5.5-**15 Sending Notification***

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Project Conclusion

As a conclusion, it's a mobile application that aids in improving the quality of one of Egypt's most essential transportation systems. It makes things a lot easier for both Metro Administrators and Metro riders.

We took a look at the current Cairo Metro application and saw there were many improvements we could make. Although we have already made some, we want to make even more in order to increase user dependence and make it more popular.

Figure 6.1-1 Mobile Development

6.2 Project Limitations

The project limitations were as below

- We wanted to use Metro stored historical data to use in the forecasting feature to help the Metro administrations by using our prediction to use in planning the operation plans, we already communicated with the Metro Administration to get the Metro ins and out for the previous 7 years in hourly basis but they only provided the total number of passengers yearly.
- We created dummy data for the previous 7 years. It contains more than 3.5 million cells so it was hard to convert to JSON format using traditional tools like python or online tools.
- By taking advantage from being supplied from Google's Machine Learning kit, we used a machine learning algorithm to use in forecasting features using Python programming language, then converted it to TensorFlow format. Unfortunately, the algorithm was unable to recognize a pattern in data and had poor accuracy because it only had the number of passengers in a specific line for a particular date, and it still needs more data to capture patterns.
- We couldn’t use Google Maps API in our application to add it as a feature to help the user to check the station’s location.
- Data unavailability. Metro administrations refused to provide detailed data for the ins and outs on an hourly basis and just provided the yearly number of riders, so we created dummy data based on probabilities to use in our application.
- Huge Database with almost 4.5 million cells, so we portioned data into groups. We have data for the previous 7 years, we group every 6 months together and worked on cleaning and transferring to JSON files using python.
- We couldn’t create the outs from stations, we just created ins, as we don’t have any reference or source to get a way to create out, we searched multiple sites (Kaggle, Data World, etc.).

6.3 Future Work

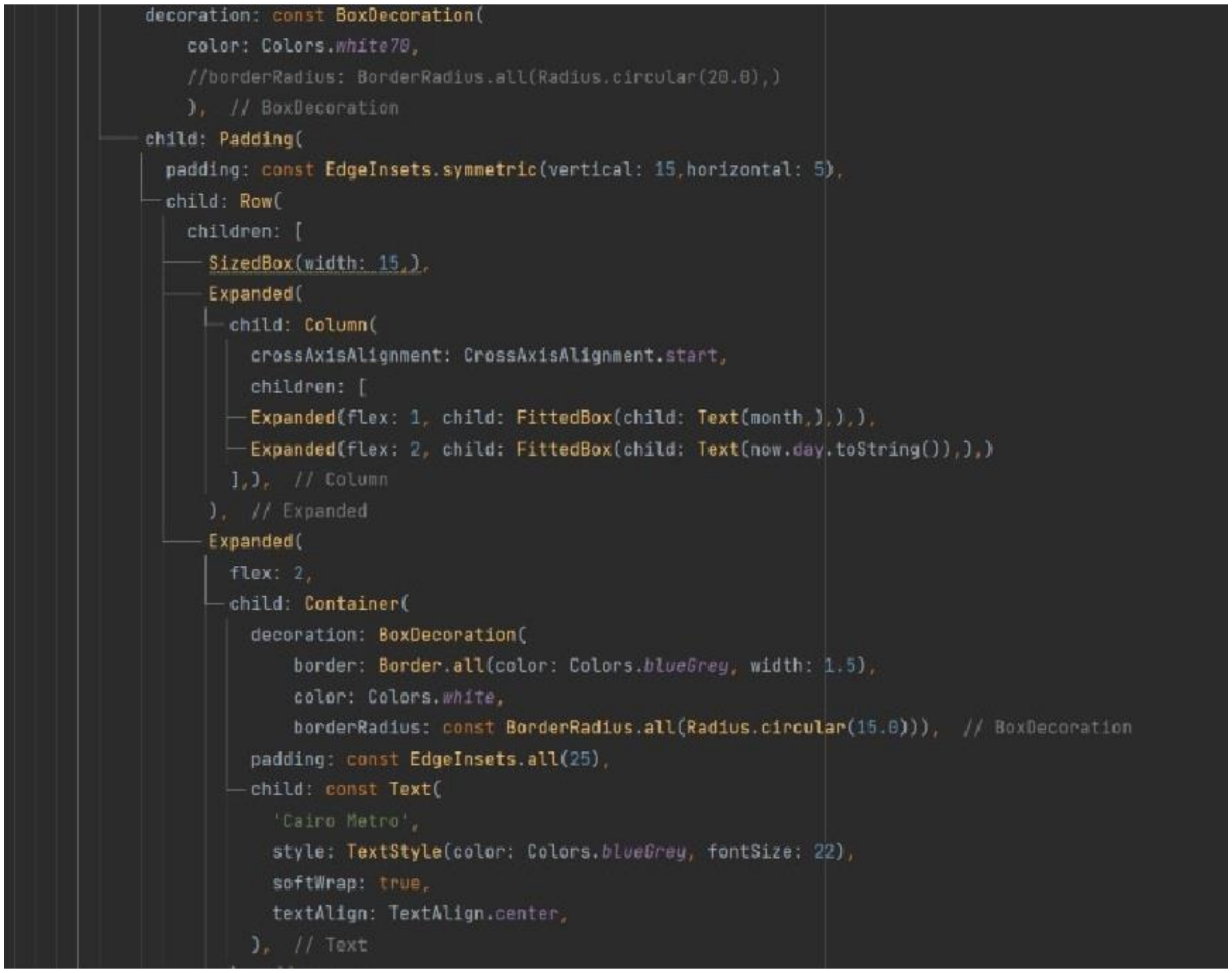
For the future aspect we want the application to be integrated with the Metro infrastructure for many reasons that we will discuss below:

- To store all the ins and outs automatically to our database in order to calculate the estimated rush for both lines and stations.
- To store all the metro users instantly to our database so that we will be able to analyze all these stored data and have the opportunity to provide insights to the Metro administration that helps in organizing operations and evaluating previous data to use as a reference.
- While our application is integrated with the Metro infrastructure we will have the ability to provide the user with the QR code feature that will used in entering gates. The application will have access to the Metro subscription system to decrease user's balance every time.
- When we have a real data in our database we will be able to use our forecasting model, we will recognize a pattern in data which will help improve Metro insights and operation plans.
- We need to change the philosophy of the tickets pricing, by using the real data analytics and insights we will be able to exactly determine the most rushed time intervals. Then, we will use these intervals to set a different pricing system, we will let be high price in the rush time and lower in the minimum crowded intervals.

APPENDICES

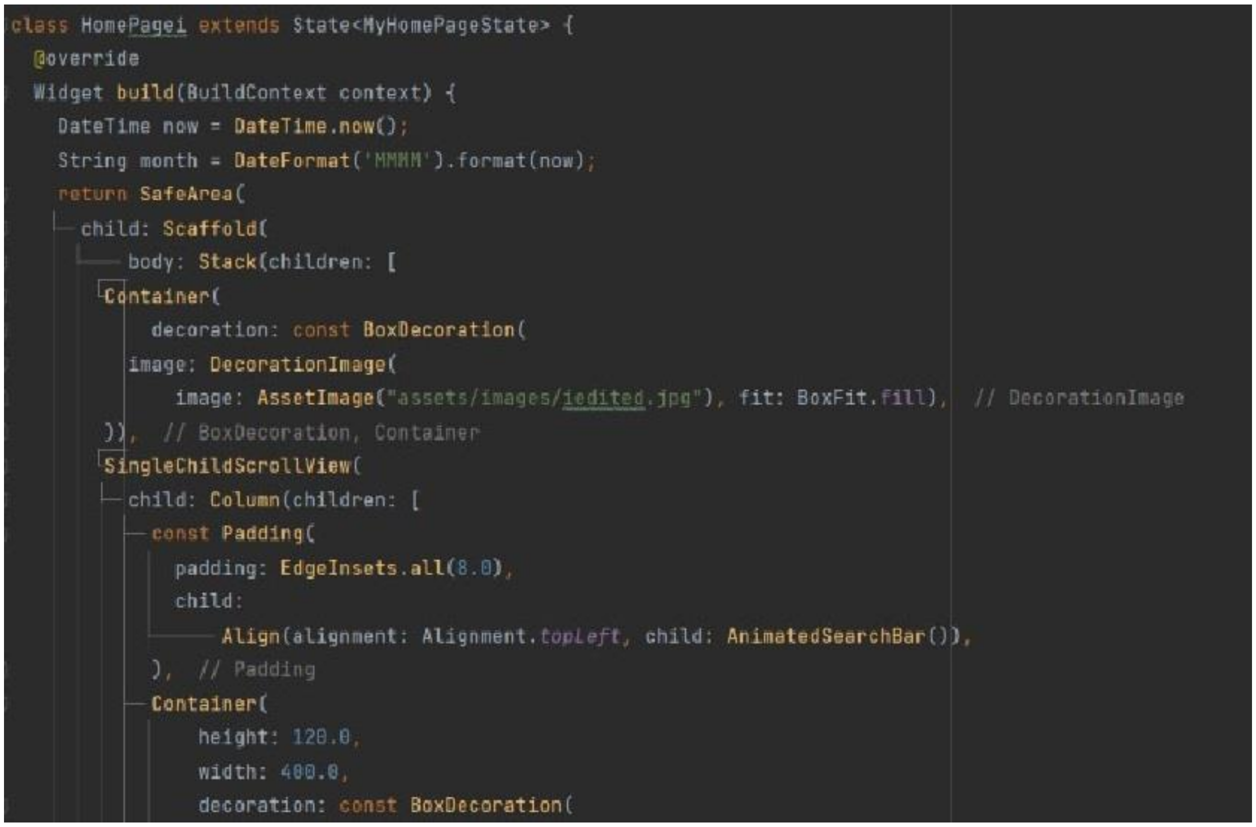
Some of the mobile application coding process attached below

Here we will find the home interface’s backend



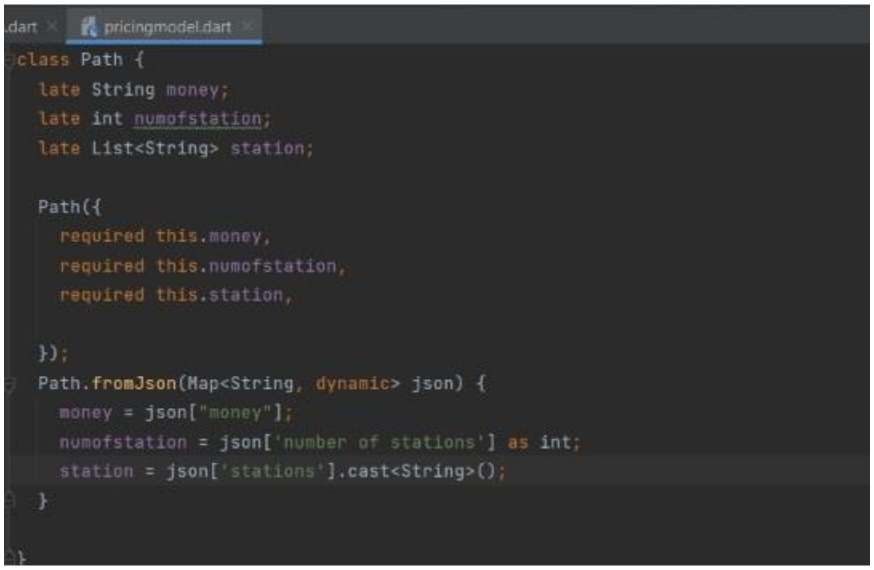
```
decoration: const BoxDecoration(
    color: Colors.white70,
    //borderRadius: BorderRadius.all(Radius.circular(20.0),)
    ),  // BoxDecoration
child: Padding(
  padding: const EdgeInsets.symmetric(vertical: 15,horizontal: 5),
  child: Row(
    children: [
      SizedBox(width: 15,),
      Expanded(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
          Expanded(flex: 1, child: FittedBox(child: Text(month,),),),
          Expanded(flex: 2, child: FittedBox(child: Text(now.day.toString()),),)
        ],),  // Column
      ),  // Expanded
      Expanded(
        flex: 2,
        child: Container(
          decoration: BoxDecoration(
              border: Border.all(color: Colors.blueGrey, width: 1.5),
              color: Colors.white,
              borderRadius: const BorderRadius.all(Radius.circular(15.0))),  // BoxDecoration
          padding: const EdgeInsets.all(25),
          child: const Text(
            'Cairo Metro',
            style: TextStyle(color: Colors.blueGrey, fontSize: 22),
            softWrap: true,
            textAlign: TextAlign.center,
          ),  // Text
```

Here we will find the function of previewing date on the main page and the calling for the animated search bar.



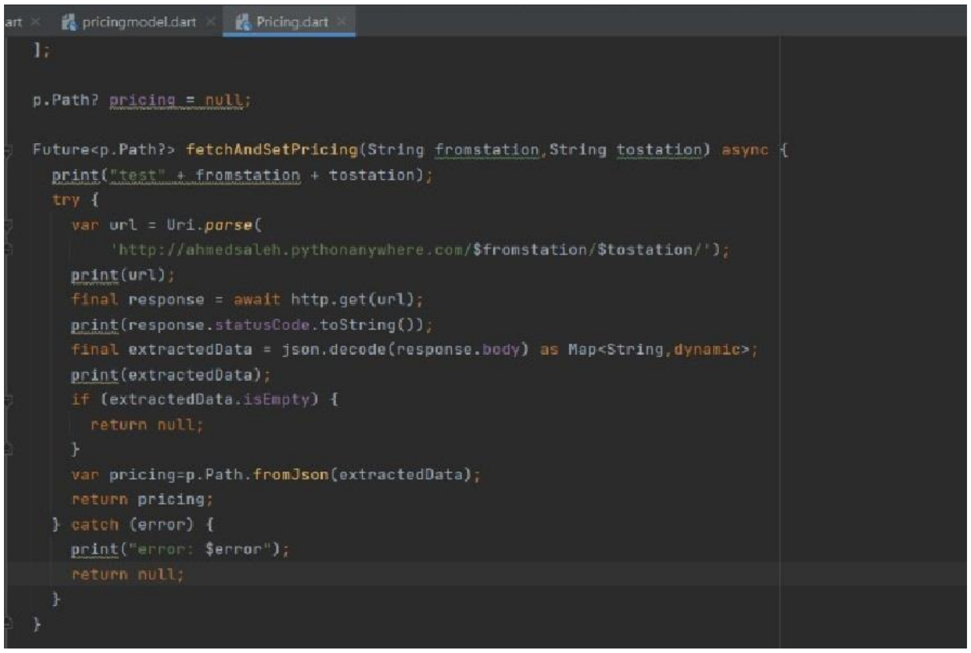
```
class HomePage1 extends State<MyHomePageState> {
  @override
  Widget build(BuildContext context) {
    DateTime now = DateTime.now();
    String month = DateFormat('MMMM').format(now);
    return SafeArea(
      child: Scaffold(
          body: Stack(children: [
        Container(
            decoration: const BoxDecoration(
          image: DecorationImage(
              image: AssetImage("assets/images/iedited.jpg"), fit: BoxFit.fill),  // DecorationImage
        )),  // BoxDecoration, Container
        SingleChildScrollView(
          child: Column(children: [
            const Padding(
              padding: EdgeInsets.all(8.0),
              child:
                  Align(alignment: Alignment.topLeft, child: AnimatedSearchBar()),
            ),  // Padding
            Container(
                height: 120.0,
                width: 400.0,
                decoration: const BoxDecoration(
```

The below picture is retrieving the ticket price to perform the pricing operations.

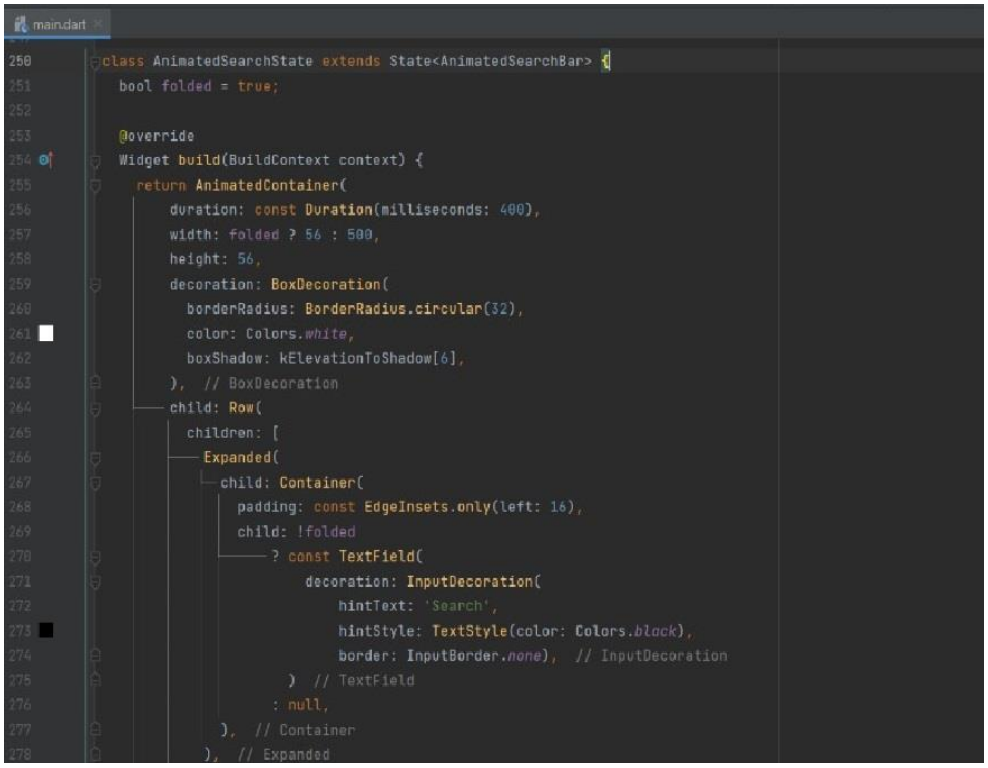


```
class Path {
  late String money;
  late int numofstation;
  late List<String> station;

  Path({
    required this.money,
    required this.numofstation,
    required this.station,

  });
  Path.fromJson(Map<String, dynamic> json) {
    money = json["money"];
    numofstation = json['number of stations'] as int;
    station = json['stations'].cast<String>();
  }

}
```

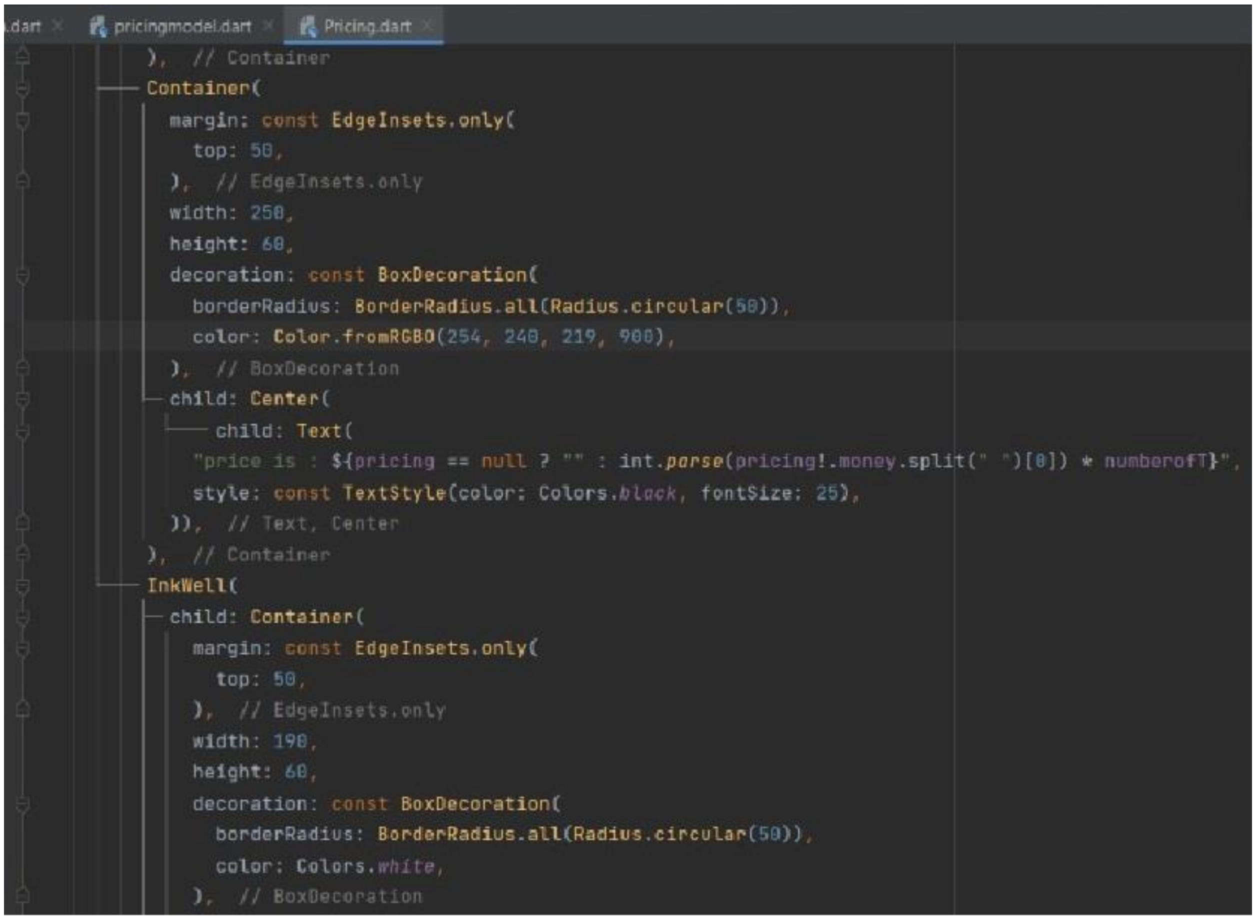
Calling API for performing the pricing operation.


```dart
];

p.Path? pricing = null;

Future<p.Path?> fetchAndSetPricing(String fromstation,String tostation) async {
  print("test" + fromstation + tostation);
  try {
    var url = Uri.parse(
        'http://ahmedsaleh.pythonanywhere.com/$fromstation/$tostation/');
    print(url);
    final response = await http.get(url);
    print(response.statusCode.toString());
    final extractedData = json.decode(response.body) as Map<String,dynamic>;
    print(extractedData);
    if (extractedData.isEmpty) {
      return null;
    }
    var pricing=p.Path.fromJson(extractedData);
    return pricing;
  } catch (error) {
    print("error: $error");
    return null;
  }
}
```

The class of the animated search bar.


```dart
class AnimatedSearchState extends State<AnimatedSearchBar> {
  bool folded = true;

  @override
  Widget build(BuildContext context) {
    return AnimatedContainer(
        duration: const Duration(milliseconds: 400),
        width: folded ? 56 : 500,
        height: 56,
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(32),
          color: Colors.white,
          boxShadow: kElevationToShadow[6],
        ), // BoxDecoration
        child: Row(
          children: [
            Expanded(
              child: Container(
                padding: const EdgeInsets.only(left: 16),
                child: !folded
                    ? const TextField(
                        decoration: InputDecoration(
                            hintText: 'Search',
                            hintStyle: TextStyle(color: Colors.black),
                            border: InputBorder.none), // InputDecoration
                      ) // TextField
                    : null,
              ), // Container
            ), // Expanded
```

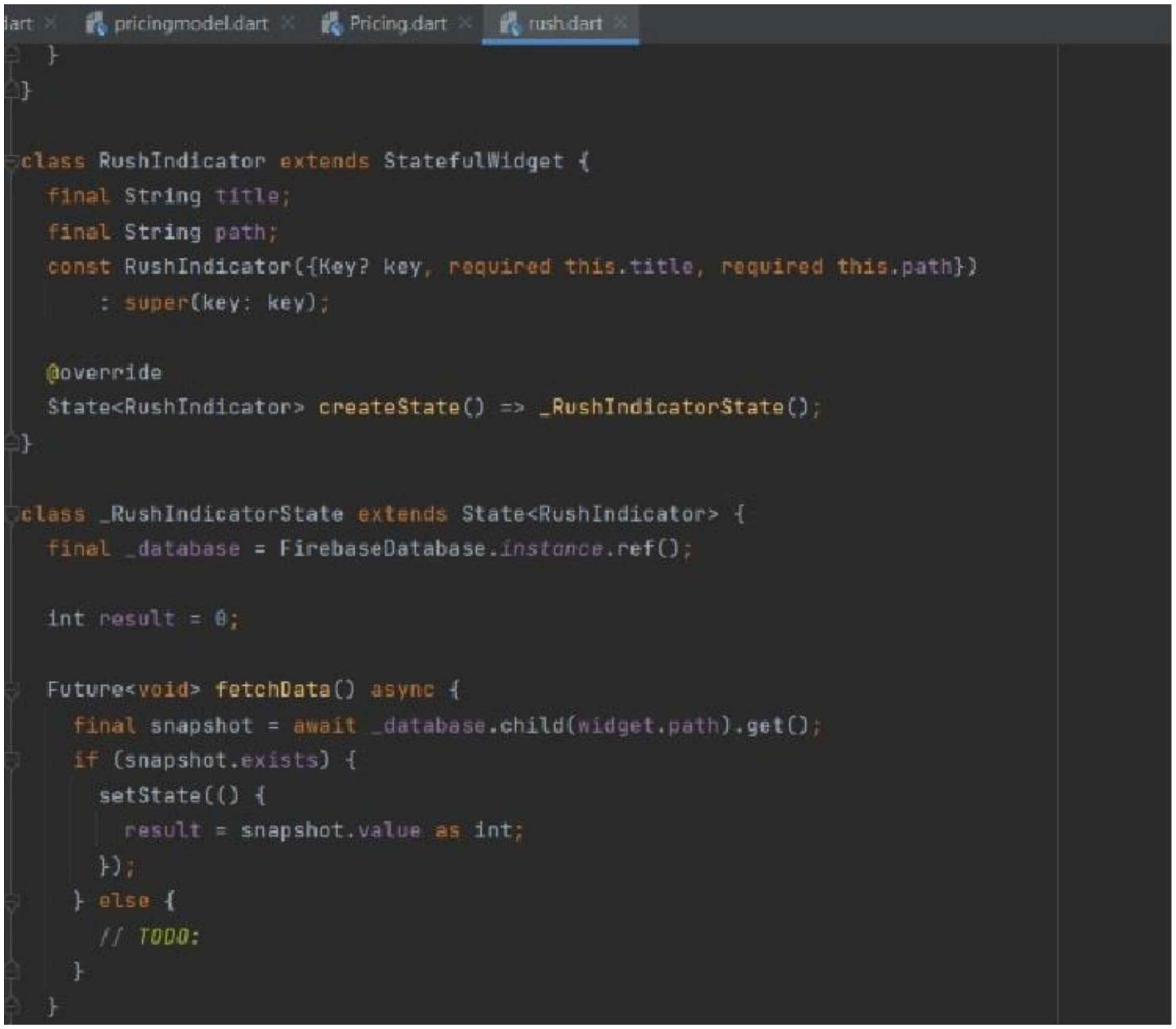
Calculating ticket's price based on the number of stations.



.dart × pricingmodel.dart × Pricing.dart ×

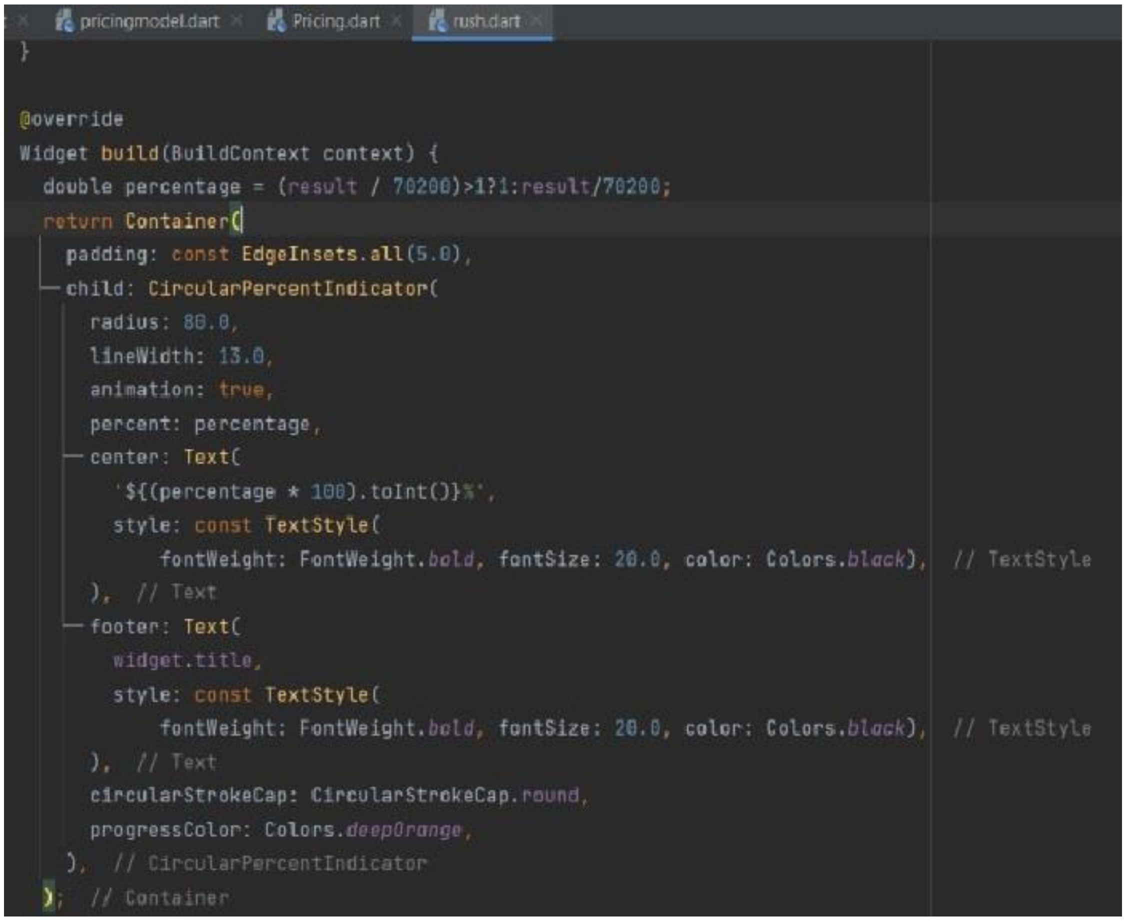
```
      ),  // Container
      Container(
        margin: const EdgeInsets.only(
          top: 50,
        ),  // EdgeInsets.only
        width: 250,
        height: 60,
        decoration: const BoxDecoration(
          borderRadius: BorderRadius.all(Radius.circular(50)),
          color: Color.fromRGBO(254, 240, 219, 900),
        ),  // BoxDecoration
        child: Center(
            child: Text(
          "price is : ${pricing == null ? "" : int.parse(pricing!.money.split(" ")[0]) * numberofT}",
          style: const TextStyle(color: Colors.black, fontSize: 25),
        )),  // Text, Center
      ),  // Container
      InkWell(
        child: Container(
          margin: const EdgeInsets.only(
            top: 50,
          ),  // EdgeInsets.only
          width: 190,
          height: 60,
          decoration: const BoxDecoration(
            borderRadius: BorderRadius.all(Radius.circular(50)),
            color: Colors.white,
          ),  // BoxDecoration
```

The class below is retrieving data from the database to use in rush calculations processes.



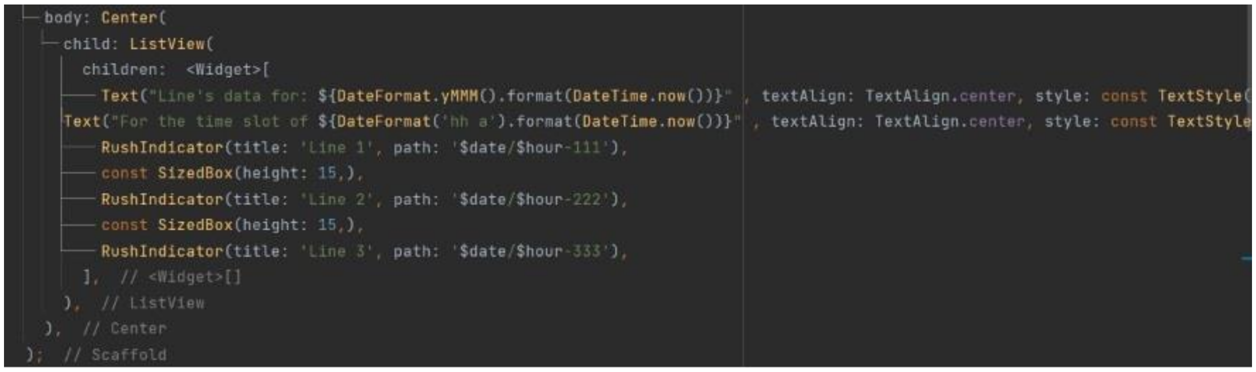
```
  }
}

class RushIndicator extends StatefulWidget {
  final String title;
  final String path;
  const RushIndicator({Key? key, required this.title, required this.path})
      : super(key: key);

  @override
  State<RushIndicator> createState() => _RushIndicatorState();
}

class _RushIndicatorState extends State<RushIndicator> {
  final _database = FirebaseDatabase.instance.ref();

  int result = 0;

  Future<void> fetchData() async {
    final snapshot = await _database.child(widget.path).get();
    if (snapshot.exists) {
      setState(() {
        result = snapshot.value as int;
      });
    } else {
      // TODO:
    }
  }
```

Calculating the line rush based on the data that has been retrieved.



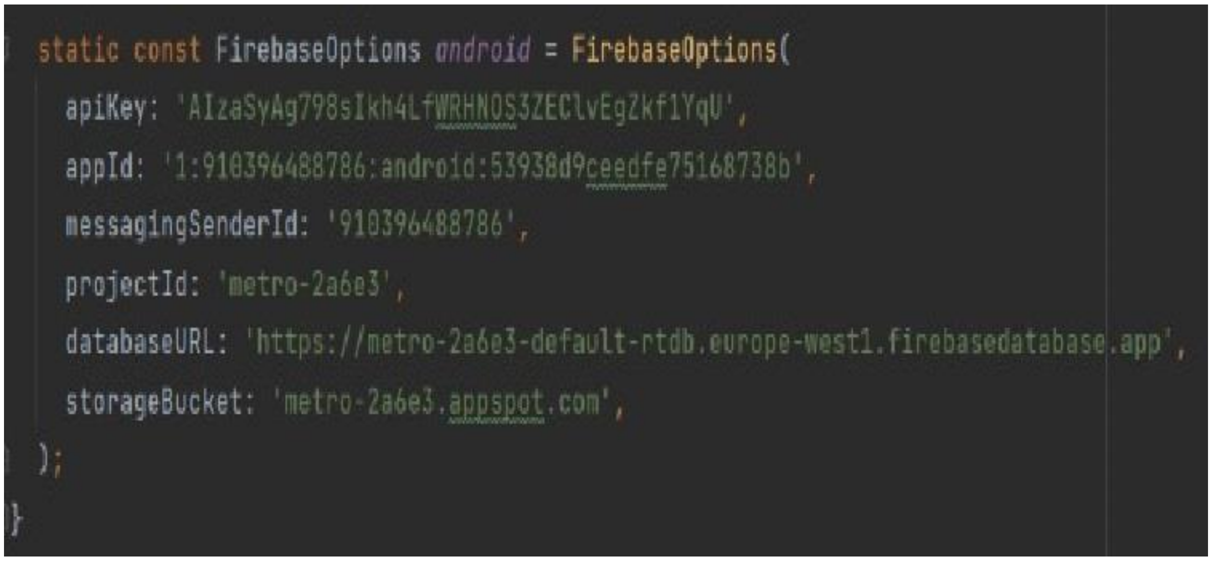
```
}

@override
Widget build(BuildContext context) {
  double percentage = (result / 70200)>1?1:result/70200;
  return Container(
    padding: const EdgeInsets.all(5.0),
    child: CircularPercentIndicator(
      radius: 80.0,
      lineWidth: 13.0,
      animation: true,
      percent: percentage,
      center: Text(
        '${(percentage * 100).toInt()}%',
        style: const TextStyle(
            fontWeight: FontWeight.bold, fontSize: 20.0, color: Colors.black),  // TextStyle
      ),  // Text
      footer: Text(
        widget.title,
        style: const TextStyle(
            fontWeight: FontWeight.bold, fontSize: 20.0, color: Colors.black),  // TextStyle
      ),  // Text
      circularStrokeCap: CircularStrokeCap.round,
      progressColor: Colors.deepOrange,
    ),  // CircularPercentIndicator
  );  // Container
```

Calling the line rush calculator class.



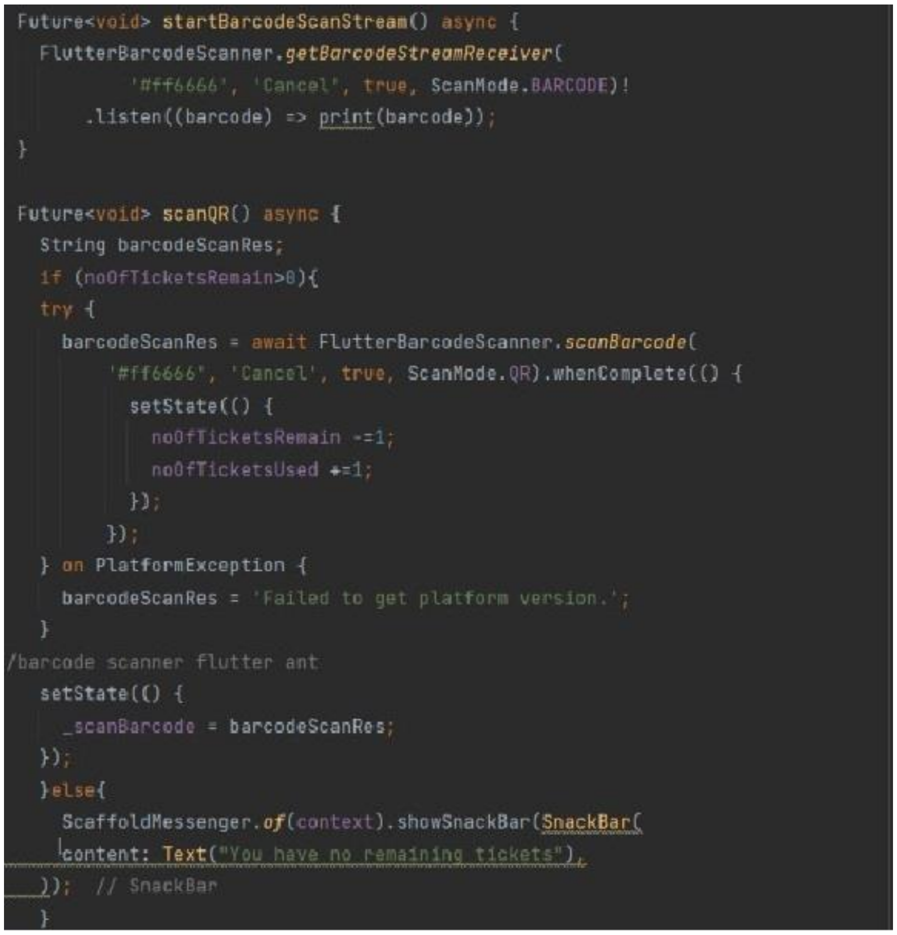
```
  body: Center(
    child: ListView(
      children:  <Widget>[
        Text("Line's data for: ${DateFormat.yMMM().format(DateTime.now())}" , textAlign: TextAlign.center, style: const TextStyle(
      Text("For the time slot of ${DateFormat('hh a').format(DateTime.now())}" , textAlign: TextAlign.center, style: const TextStyle
        RushIndicator(title: 'Line 1', path: '$date/$hour-111'),
        const SizedBox(height: 15,),
        RushIndicator(title: 'Line 2', path: '$date/$hour-222'),
        const SizedBox(height: 15,),
        RushIndicator(title: 'Line 3', path: '$date/$hour-333'),
      ],  // <Widget>[]
    ),  // ListView
  ),  // Center
);  // Scaffold
```

Firebase connection establishing.



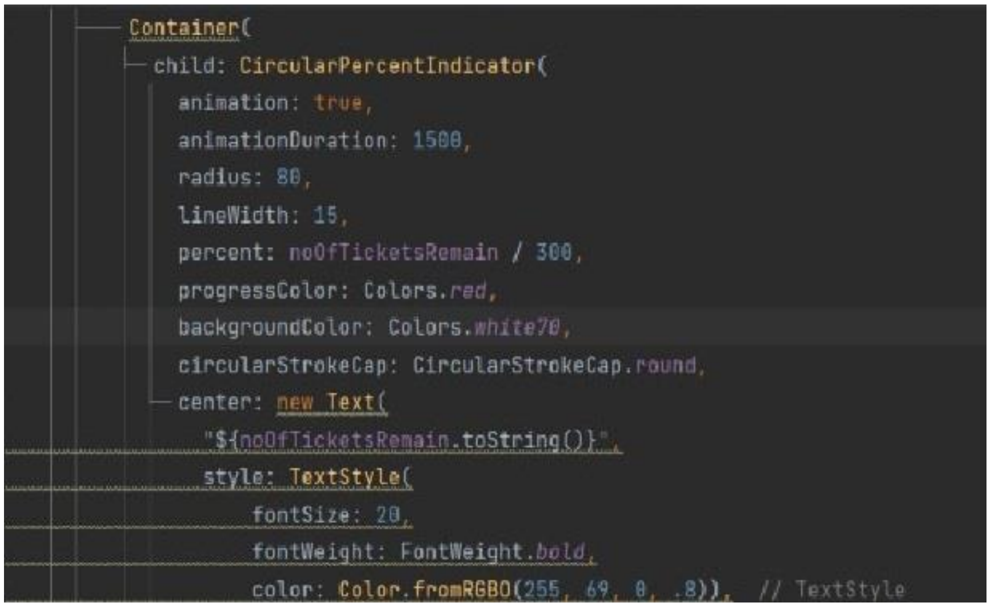
```
  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyAg798sIkh4LfWRHNOS3ZEClvEgZkf1YqU',
    appId: '1:910396488786:android:53938d9ceedfe75168738b',
    messagingSenderId: '910396488786',
    projectId: 'metro-2a6e3',
    databaseURL: 'https://metro-2a6e3-default-rtdb.europe-west1.firebasedatabase.app',
    storageBucket: 'metro-2a6e3.appspot.com',
  );
}
```

The user profile.



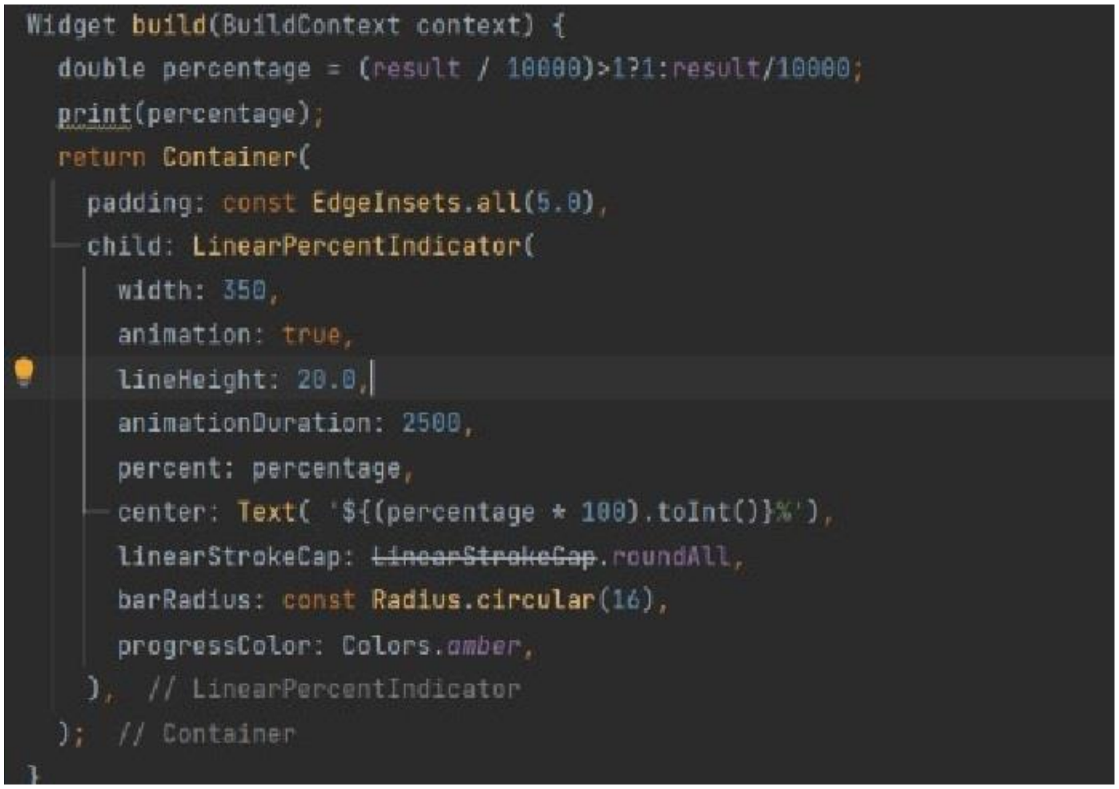
```
Future<void> startBarcodeScanStream() async {
  FlutterBarcodeScanner.getBarcodeStreamReceiver(
          '#ff6666', 'Cancel', true, ScanMode.BARCODE)!
      .listen((barcode) => print(barcode));
}

Future<void> scanQR() async {
  String barcodeScanRes;
  if (noOfTicketsRemain>0){
  try {
    barcodeScanRes = await FlutterBarcodeScanner.scanBarcode(
        '#ff6666', 'Cancel', true, ScanMode.QR).whenComplete(() {
          setState(() {
            noOfTicketsRemain -=1;
            noOfTicketsUsed +=1;
          });
        });
  } on PlatformException {
    barcodeScanRes = 'Failed to get platform version.';
  }
/barcode scanner flutter ant
  setState(() {
    _scanBarcode = barcodeScanRes;
  });
  }else{
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
    content: Text("You have no remaining tickets"),
  )); // SnackBar
  }
```

User balance’s circular.



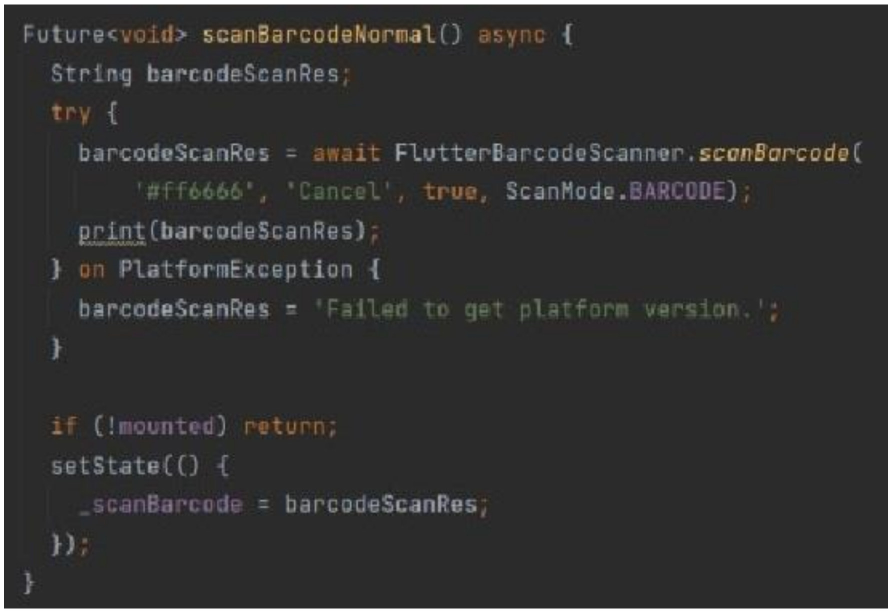
```
Container(
  child: CircularPercentIndicator(
    animation: true,
    animationDuration: 1500,
    radius: 80,
    lineWidth: 15,
    percent: noOfTicketsRemain / 300,
    progressColor: Colors.red,
    backgroundColor: Colors.white70,
    circularStrokeCap: CircularStrokeCap.round,
    center: new Text(
      "${noOfTicketsRemain.toString()}",
      style: TextStyle(
          fontSize: 20,
          fontWeight: FontWeight.bold,
          color: Color.fromRGBO(255, 69, 0, .8)),  // TextStyle
```

The station rush calculation and bar.



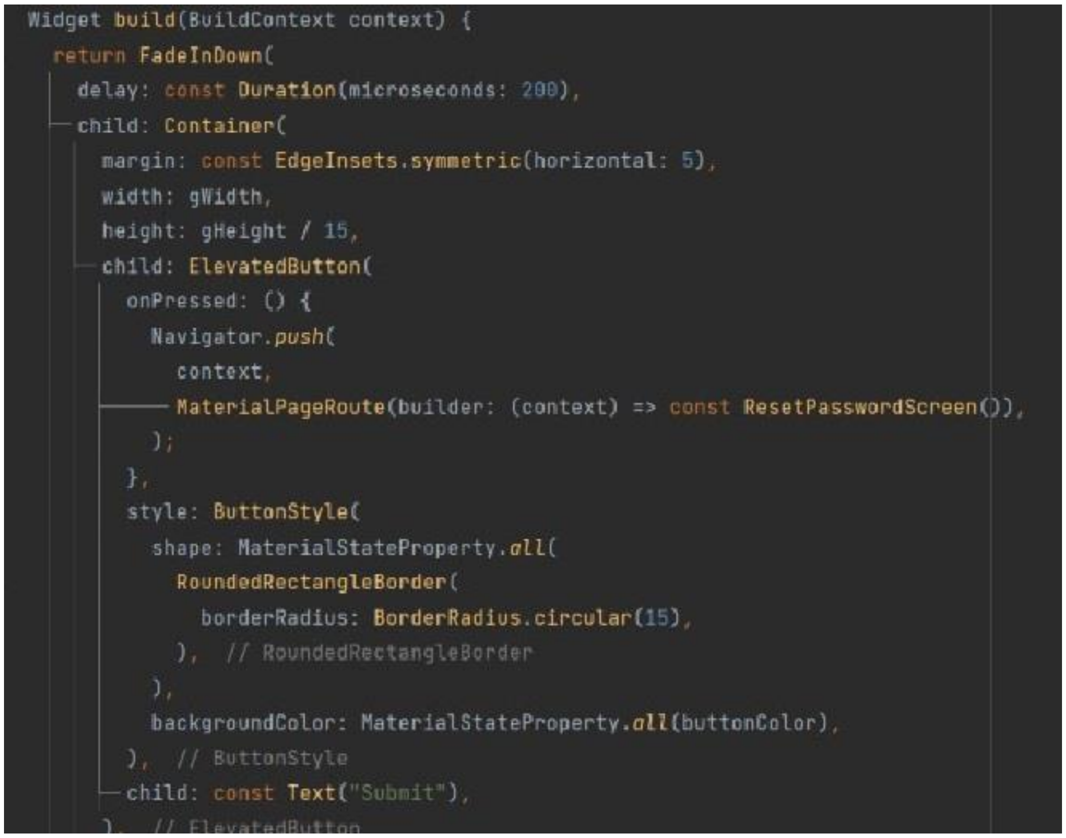
```
Widget build(BuildContext context) {
  double percentage = (result / 10000)>1?1:result/10000;
  print(percentage);
  return Container(
    padding: const EdgeInsets.all(5.0),
    child: LinearPercentIndicator(
      width: 350,
      animation: true,
      lineHeight: 20.0,
      animationDuration: 2500,
      percent: percentage,
      center: Text( '${(percentage * 100).toInt()}%'),
      linearStrokeCap: ~~LinearStrokeCap~~.roundAll,
      barRadius: const Radius.circular(16),
      progressColor: Colors.amber,
    ),  // LinearPercentIndicator
  );  // Container
}
```

The QR code scanner feature.



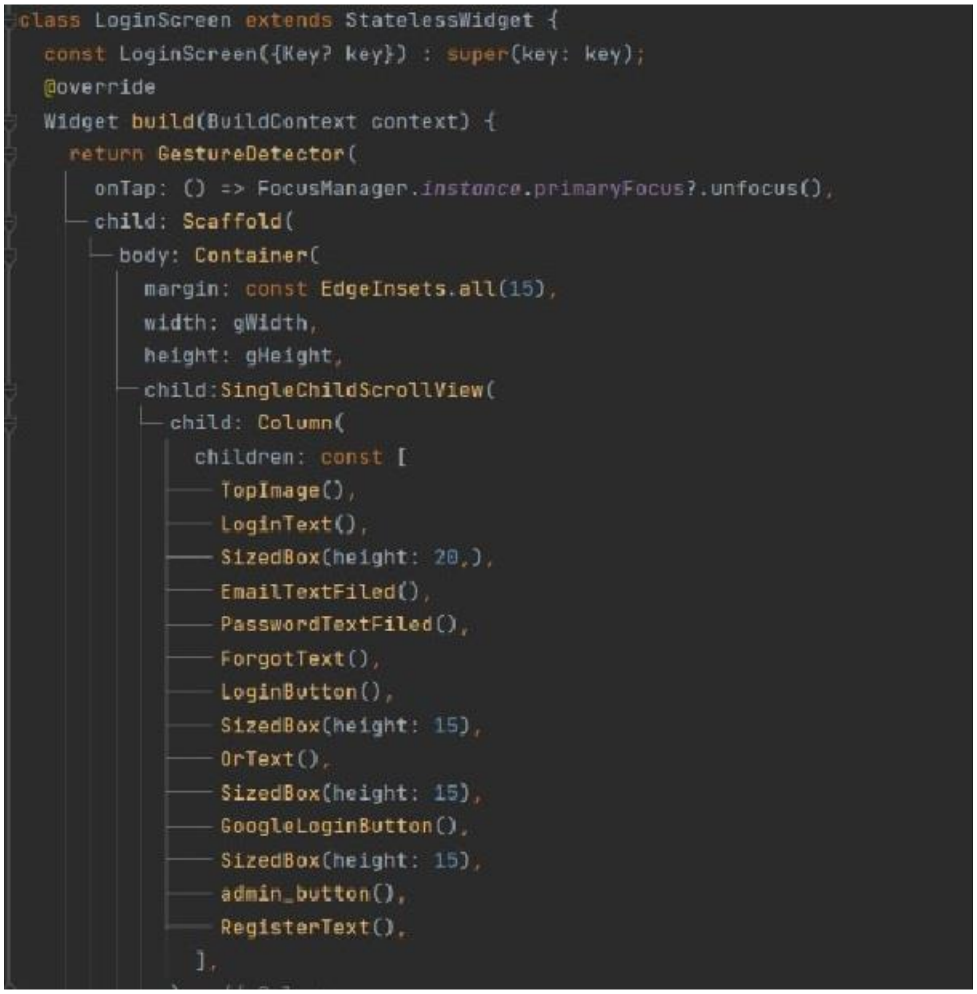
```
Future<void> scanBarcodeNormal() async {
  String barcodeScanRes;
  try {
    barcodeScanRes = await FlutterBarcodeScanner.scanBarcode(
        '#ff6666', 'Cancel', true, ScanMode.BARCODE);
    print(barcodeScanRes);
  } on PlatformException {
    barcodeScanRes = 'Failed to get platform version.';
  }

  if (!mounted) return;
  setState(() {
    _scanBarcode = barcodeScanRes;
  });
}
```

Password resetting.



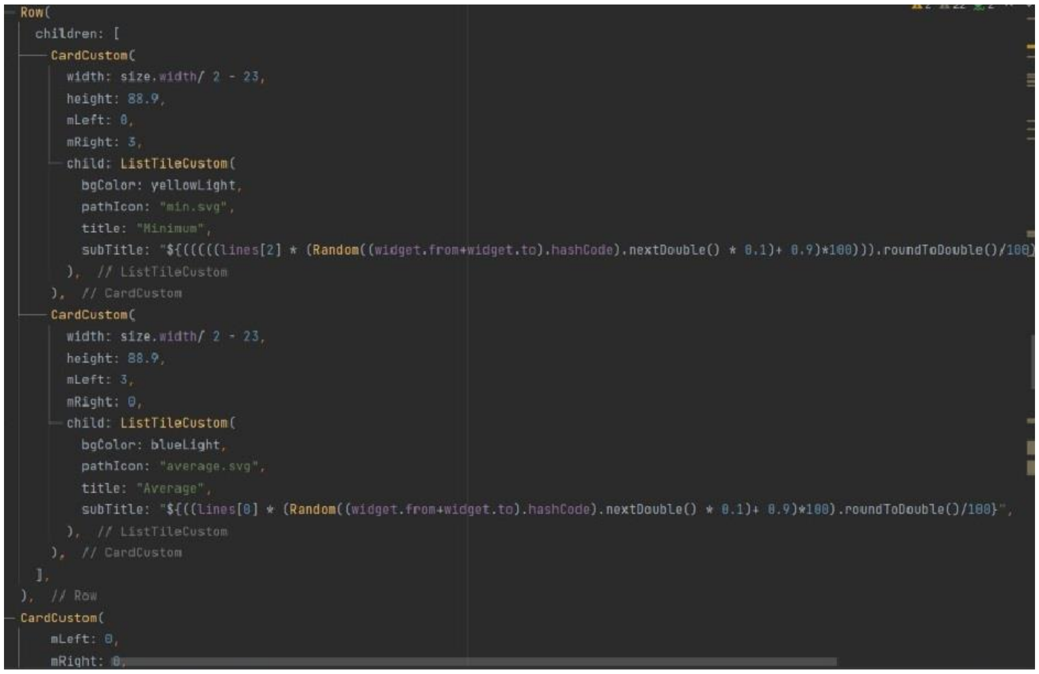
```
Widget build(BuildContext context) {
  return FadeInDown(
    delay: const Duration(microseconds: 200),
    child: Container(
      margin: const EdgeInsets.symmetric(horizontal: 5),
      width: gWidth,
      height: gHeight / 15,
      child: ElevatedButton(
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => const ResetPasswordScreen()),
          );
        },
        style: ButtonStyle(
          shape: MaterialStateProperty.all(
            RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(15),
            ), // RoundedRectangleBorder
          ),
          backgroundColor: MaterialStateProperty.all(buttonColor),
        ), // ButtonStyle
        child: const Text("Submit"),
      ), // ElevatedButton
```

The log-in screen.



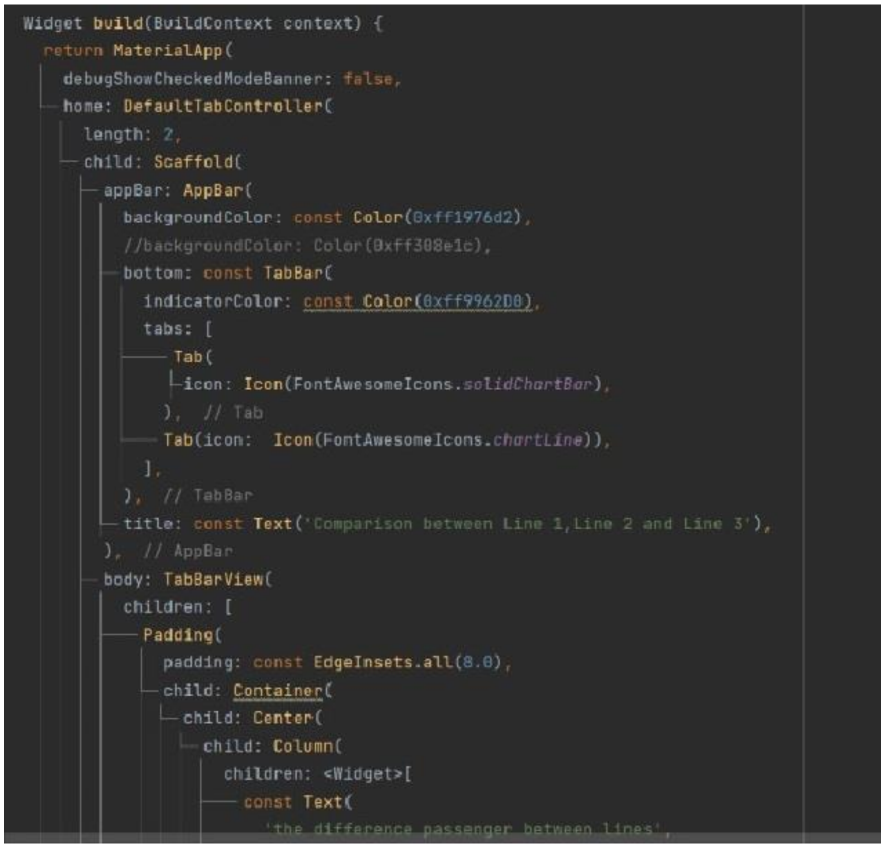
```
class LoginScreen extends StatelessWidget {
  const LoginScreen({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () => FocusManager.instance.primaryFocus?.unfocus(),
      child: Scaffold(
        body: Container(
          margin: const EdgeInsets.all(15),
          width: gWidth,
          height: gHeight,
          child:SingleChildScrollView(
            child: Column(
              children: const [
                TopImage(),
                LoginText(),
                SizedBox(height: 20,),
                EmailTextFiled(),
                PasswordTextFiled(),
                ForgotText(),
                LoginButton(),
                SizedBox(height: 15),
                OrText(),
                SizedBox(height: 15),
                GoogleLoginButton(),
                SizedBox(height: 15),
                admin_button(),
                RegisterText(),
              ],
```

The analysis operations (Average, and MIN).



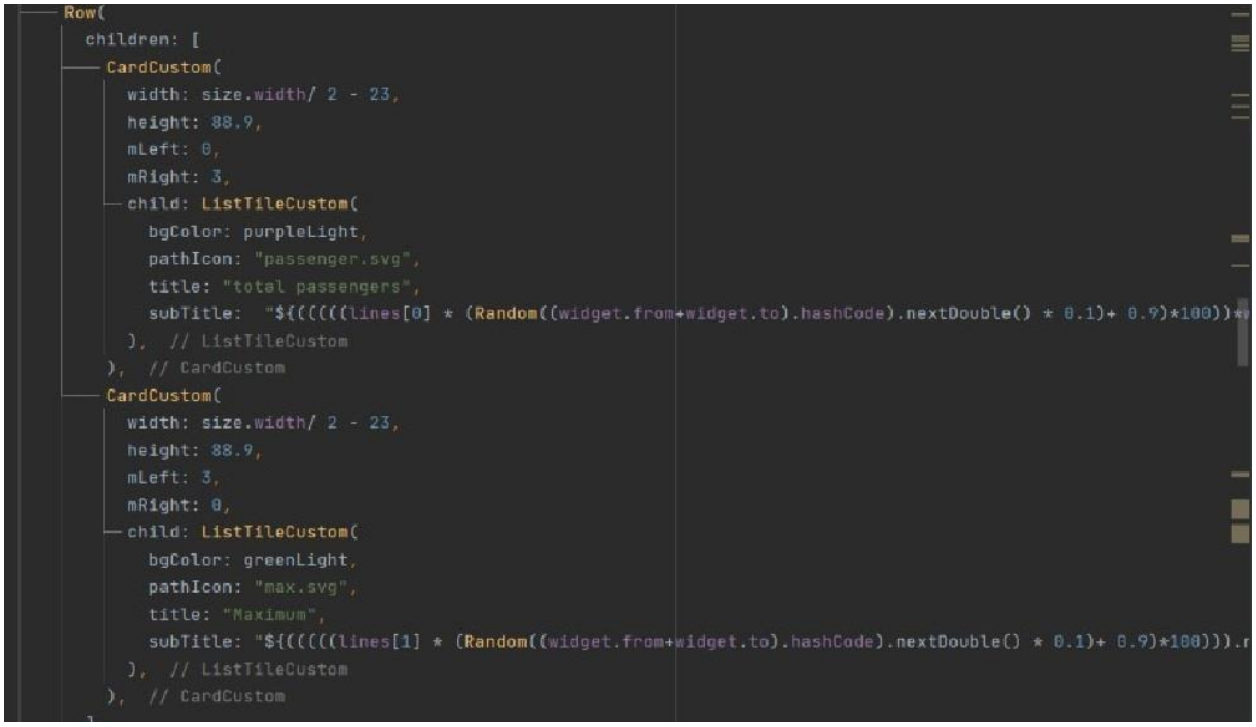
```
Row(
  children: [
    CardCustom(
      width: size.width/ 2 - 23,
      height: 88.9,
      mLeft: 0,
      mRight: 3,
      child: ListTileCustom(
        bgColor: yellowLight,
        pathIcon: "min.svg",
        title: "Minimum",
        subTitle: "${((((((lines[2] * (Random((widget.from+widget.to).hashCode).nextDouble() * 0.1)+ 0.9)*100))).roundToDouble()/100
      ),  // ListTileCustom
    ),  // CardCustom
    CardCustom(
      width: size.width/ 2 - 23,
      height: 88.9,
      mLeft: 3,
      mRight: 0,
      child: ListTileCustom(
        bgColor: blueLight,
        pathIcon: "average.svg",
        title: "Average",
        subTitle: "${((lines[0] * (Random((widget.from+widget.to).hashCode).nextDouble() * 0.1)+ 0.9)*100).roundToDouble()/100}",
      ),  // ListTileCustom
    ),  // CardCustom
  ],
),  // Row
CardCustom(
    mLeft: 0,
    mRight: 0,
```

Comparison between lines in analysis



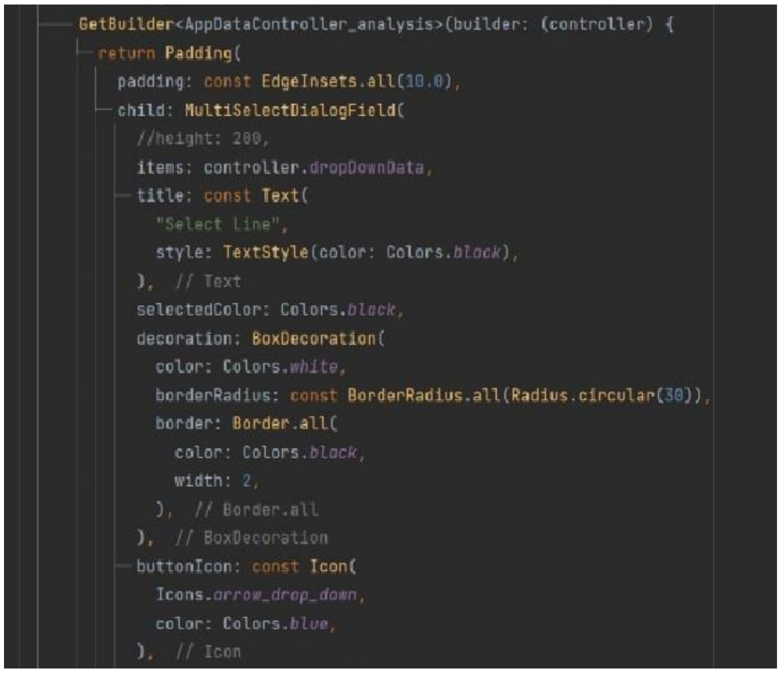
```
Widget build(BuildContext context) {
  return MaterialApp(
    debugShowCheckedModeBanner: false,
    home: DefaultTabController(
      length: 2,
      child: Scaffold(
        appBar: AppBar(
          backgroundColor: const Color(0xff1976d2),
          //backgroundColor: Color(0xff308e1c),
          bottom: const TabBar(
            indicatorColor: const Color(0xff9962D0),
            tabs: [
               Tab(
                icon: Icon(FontAwesomeIcons.solidChartBar),
              ),  // Tab
              Tab(icon:  Icon(FontAwesomeIcons.chartLine)),
            ],
          ),  // TabBar
          title: const Text('Comparison between Line 1,Line 2 and Line 3'),
        ),  // AppBar
        body: TabBarView(
          children: [
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Container(
                child: Center(
                  child: Column(
                    children: <Widget>[
                      const Text(
                        'the difference passenger between lines',
```

The analysis operations (MAX, and SUM).



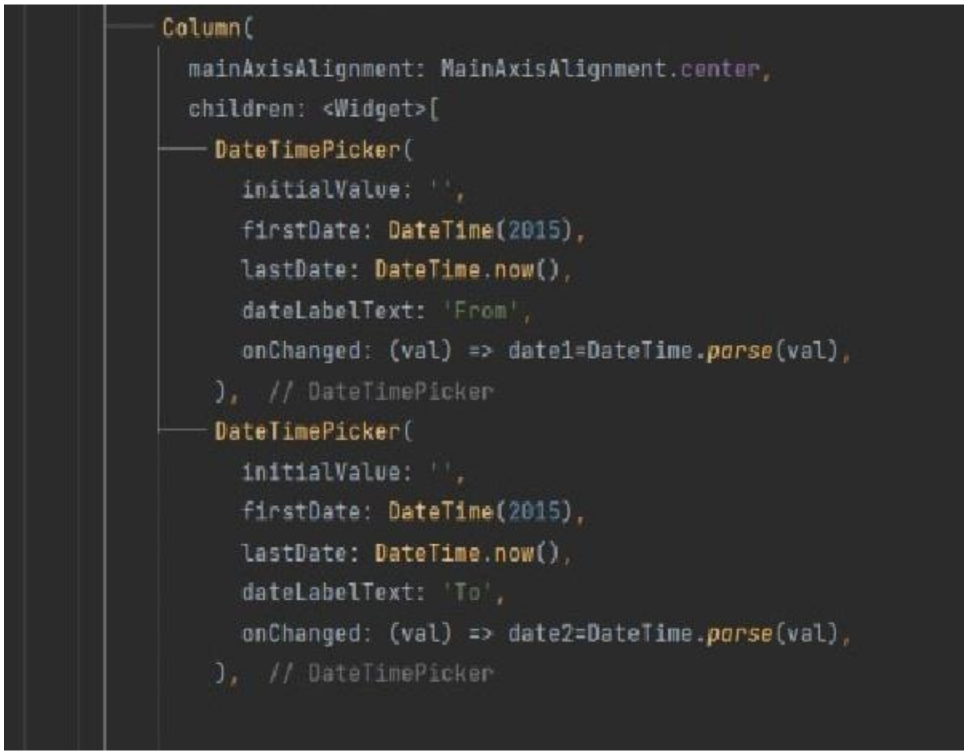
```
Row(
  children: [
  CardCustom(
    width: size.width/ 2 - 23,
    height: 88.9,
    mLeft: 0,
    mRight: 3,
    child: ListTileCustom(
      bgColor: purpleLight,
      pathIcon: "passenger.svg",
      title: "total passengers",
      subTitle:  "${((((((lines[0] * (Random((widget.from+widget.to).hashCode).nextDouble() * 0.1)+ 0.9)*100))*
    ),  // ListTileCustom
  ),  // CardCustom
  CardCustom(
    width: size.width/ 2 - 23,
    height: 88.9,
    mLeft: 3,
    mRight: 0,
    child: ListTileCustom(
      bgColor: greenLight,
      pathIcon: "max.svg",
      title: "Maximum",
      subTitle: "${((((((lines[1] * (Random((widget.from+widget.to).hashCode).nextDouble() * 0.1)+ 0.9)*100))).r
    ),  // ListTileCustom
  ),  // CardCustom
```

Choosing a line (Analysis, and station determining)



```
GetBuilder<AppDataController_analysis>(builder: (controller) {
  return Padding(
    padding: const EdgeInsets.all(10.0),
    child: MultiSelectDialogField(
      //height: 200,
      items: controller.dropDownData,
      title: const Text(
        "Select Line",
        style: TextStyle(color: Colors.black),
      ),  // Text
      selectedColor: Colors.black,
      decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: const BorderRadius.all(Radius.circular(30)),
        border: Border.all(
          color: Colors.black,
          width: 2,
        ),  // Border.all
      ),  // BoxDecoration
      buttonIcon: const Icon(
        Icons.arrow_drop_down,
        color: Colors.blue,
      ),  // Icon
```

Date and time picker button that has been used in Analysis



```
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
  DateTimePicker(
    initialValue: '',
    firstDate: DateTime(2015),
    lastDate: DateTime.now(),
    dateLabelText: 'From',
    onChanged: (val) => date1=DateTime.parse(val),
  ),  // DateTimePicker
  DateTimePicker(
    initialValue: '',
    firstDate: DateTime(2015),
    lastDate: DateTime.now(),
    dateLabelText: 'To',
    onChanged: (val) => date2=DateTime.parse(val),
  ),  // DateTimePicker
```

REFERENCES

- https://guides.libraries.uc.edu/MatlabForEngineers
- https://en.wikipedia.org/wiki/Mathematical_optimization
- https://www.engati.com/glossary/mathematical-optimization
- https://web.stanford.edu/group/sisl/k12/optimization/#!index.md
- https://firebase.google.com/docs
- https://firebase.google.com/docs/database
- https://firebase.google.com/docs/guides
- https://firebase.google.com/docs/flutter/setup?platform=android
- https://ieeexplore.ieee.org/abstract/document/4632123
- https://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/
- https://flutter.dev/
- https://pub.dev/
- https://app.asana.com/0/1202611093044518/timeline
- https://en.wikipedia.org/wiki/List_of_Cairo_Metro_stations
- https://online.visual-paradigm.com/app/diagrams/#diagram:proj=0&type=SequenceDiagram&width=11&height=8.5&unit=inch