Cairo University

Faculty of Computers and Artificial Intelligence

Computer Science Department

Metro Tickets Reservation

Implemented By

Ahmed Ibrahim Mohamed Kassab	20170008
Ahmed Mostafa Elsayed	20170033
Mohamed Bakr Abdelhafez	20170224
Mohamed Sameh Omar	20170232
Mohamed Mohsen Abdelsallam	20170253

Supervised By

Dr. Cherry Ahmed
TA. Ashraf Mohey

Graduation Project
Academic Year 2020-2021

Final Documentation

Project Documentation

Metro Ticket Reservation

Supervised By:

Dr. Cherry Ahmed

TA. Ashraf Mohey

Table of Content:

List of Figures 4

CHAPTER 1 : INTRODUCTION 6

1.1 Motivation 6

1.2 Background 7

1.2.1 Large number of Metro Passengers 7

1.2.2 The prevalence of mobile applications 7

1.2.3 QR Code Technology 7

1.3 Problem Definition 8

1.4 Project Objective 8

1.5 Gantt Chart of Project Time Plan 9

1.5.1 Analysis gantt chart 9

1.5.2 Implementation gantt chart 10

1.6 Project Development Methodology 11

1.7 Software and Hardware Tools 11

1.7.1 Software tools 11

1.7.2 Hardware tools 11

1.8 Report Organization 12

CHAPTER 2 : RELATED WORK 13

CHAPTER 3 : SYSTEM ANALYSIS 15

3.1 Project Specification 15

3.1.1 Functional requirements 15

3.1.2 Non-functional requirements 16

3.1.2.1 Performance Requirements 16

3.1.2.2 Security Requirements 16

3.1.2.3 Software Quality Attributes 17

3.2 Use Case Diagram 18

CHAPTER 4 : SYSTEM DESIGN **19**
4.1 System Architecture 19
4.2 System Component Diagram 20
4.3 System Class Diagram 21
4.4 Sequence Diagrams 22
4.4.1 Buy ticket sequence diagram 22
4.4.2 Charge wallet sequence diagram 23
4.4.3 Get closest station sequence diagram 24
4.4.4 Use subscription sequence diagram 25
4.4.5 Make subscription sequence diagram 26
4.4.6 Use ticket sequence diagram 27
4.5 Entity Relationship Diagram (ERD) 28
4.5 System Gui Design 29
4.5.1 Mobile app GUI 29
4.5.2 Web app GUI 33

CHAPTER 5 : IMPLEMENTATION AND TESTING **37**
5.1 Implementation 37
5.1.1 Subsystems for user mobile app and their main functions 38
5.1.1.1 Account and user 38
5.1.1.2 Payment 38
5.1.1.3 Ticket 39
5.1.1.4 Subscription 39
5.1.1.5 Station 40
5.1.1.6 Trip 40
5.1.1.7 Machine 41
5.1.2 Subsystems for admin web and their functions 42
5.1.2.1 Admin 42
5.1.2.2 Basic ticket 42
5.1.2.3 Basic subscription 42
5.1.2.4 Basic station 42

5.1.3 User mobile application 42
5.2 Testing 43

References **50**

List of Figures

- Figure 1. Analysis gantt chart
- Figure 2. Implementation gantt chart
- Figure 3. Use case diagram
- Figure 4. Component diagram
- Figure 5. Class diagram
- Figure 6. Buy a ticket sequence diagram
- Figure 7. Charge wallet sequence diagram
- Figure 8. Get closest station sequence diagram
- Figure 9. Use subscription sequence diagram
- Figure 10. Make subscription sequence diagram
- Figure 11. Use ticket sequence diagram
- Figure 12. ERD

- Figure 13. Login mobile page
- Figure 14. Setting mobile page
- Figure 15. Buy tickets
- Figure 16. My tickets
- Figure 17. Get nearest station before
- Figure 18. Get nearest station after

- Figure 19. Login web page
- Figure 20. Home web page
- Figure 21. Ticket web page
- Figure 22. Subscription web page
- Figure 23. Station web page

- Figure 24. Add station page
- Figure 25. Edit station page

- Figure 26. Sign up input
- Figure 27. Sign up output
- Figure 28. Login input
- Figure 29. Login output
- Figure 30. Charge wallet input
- Figure 31. Charge wallet output
- Figure 32. Subscription input
- Figure 33. Subscription output
- Figure 34. Buy ticket input
- Figure 35. But ticket output
- Figure 36. Get route input
- Figure 37. Get route output

CHAPTER 1 : INTRODUCTION

1.1 Motivation

With this large number of Metro passengers, stations become extremely crowded, especially in times previous and after a heavy work day for employees or students in school, as a result of that each person becomes bound to stay at a too long queue in the metro station in front of the ticket window.

In addition to the delays of their journeys, in days like we are living now (a period of serious illness) The infection becomes more common in this crowded queue, as it happens in this period, since these lines cannot be controlled effectively enough. Our goal is to solve the extreme crowding of metro stations by handling all the process of booking a ticket by the mobile app.

Besides what is already mentioned, there are many people who often take the Metro at long intervals ,so they usually don't know the route to their destination and often they have problems with the sequence of their stations.

So we provide the users of our application with a lot of privileges to help him decide what metro stations he/she should take and what is the route to them. As well as that we provide our users the sequence of the stations he/she should follow to reach their destination.

1.2 Background

1.2.1 Large number of Metro Passengers

In a comprehensive report issued by the Egyptian Council of Ministers Information Centre in 2019, which includes numerous figures and details, the Centre indicated in its report published at Facebook, that the number of passengers served by the subway amounts to 3.5 million per day by completing the implementation of the third line of the metro, as well as 1662 trips per day in 2019, compared to 1544 trips per day in 2014.

1.2.2 The prevalence of mobile applications

Mobile Applications, or apps, have taken over in terms of user reach. Here are some stats to support our argument: An annual report on mobility from Ericsson states that smartphones are poised to reach 6.1 billion users by 2020.

Gartner predicts that by the end of 2017, mobile apps downloads will exceed 268 billion times which will generate revenue somewhere over $77 billion.

As a result, organizations or companies are heading to develop mobile applications to make its services easier for users, solve some problems they face or to get more profit by Sponsorship Agreements, given the many users of this application.

1.2.3 QR Code Technology

QR i.e. "Quick Response" code is a 2D matrix code that is designed by keeping two points under consideration. It stores large amounts of data as compared to 1D barcodes and it is decoded at high speed using any handheld device like phones.

QR code provides high data storage capacity, fast scanning, omnidirectional readability, and many other advantages including, error-correction (so that damaged code can also be read successfully) and different types of versions.

Nowadays, a QR code is applied in different application streams related to marketing, security, academics etc. and gain popularity at a really high pace. Day by day more people are getting aware of this technology and use it accordingly. The popularity of QR code grows rapidly with the growth of smartphone users and thus the QR code is rapidly arriving at high levels of acceptance worldwide.

1.3 Problem Definition

The problem lies in the effort that people make to buy metro tickets. Besides that, Many people who do not usually use the Metro regularly have issues when they enter the Metro station, they don't know which line and which direction they should follow to reach their destination station.

In addition to all that mentioned, People usually stuck in routine queues in order to create a metro subscription or to renew it.

1.4 Project Objective

handle all the process of reserving a ticket by the mobile app. Use mobile phones instead of Metro smart cards for subscriptions and handle the process of activating or renewing the subscription by the mobile application.

Allow users to find the station they want and provide a described sequence of stations he/she should follow from the start station to their final station.

Let the users easily charge his balance using a secured payment method. Handle the ticket or subscription validation process by using QR Code technology instead of normal tickets.

Develop some features to make the application better and more usable. Our Mobile Application is implemented by Flutter framework to be accessible to every smartphone which supports either Android or IOS Operating Systems.

1.5 Gantt Chart of Project Time Plan

Project gantt chart is divided into two parts to be more clear and these two parts are (analysis and implementation)

1.5.1 Analysis gantt chart

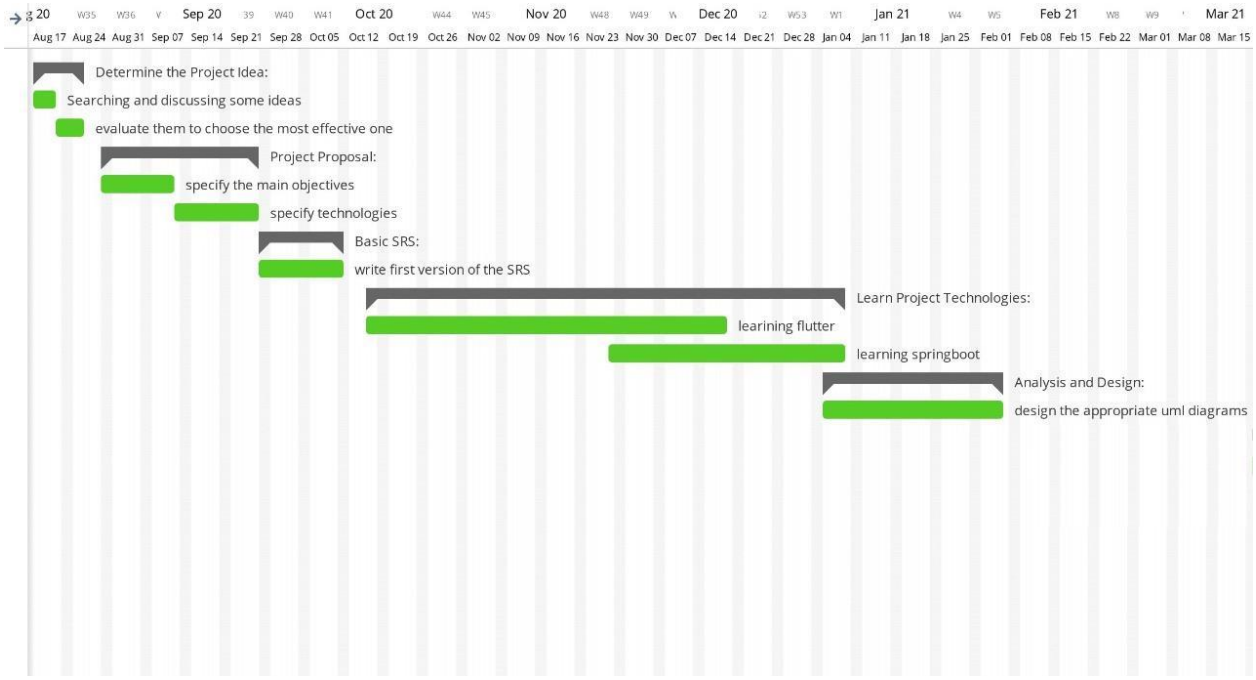


Figure 1.

1.5.2 Implementation gantt chart

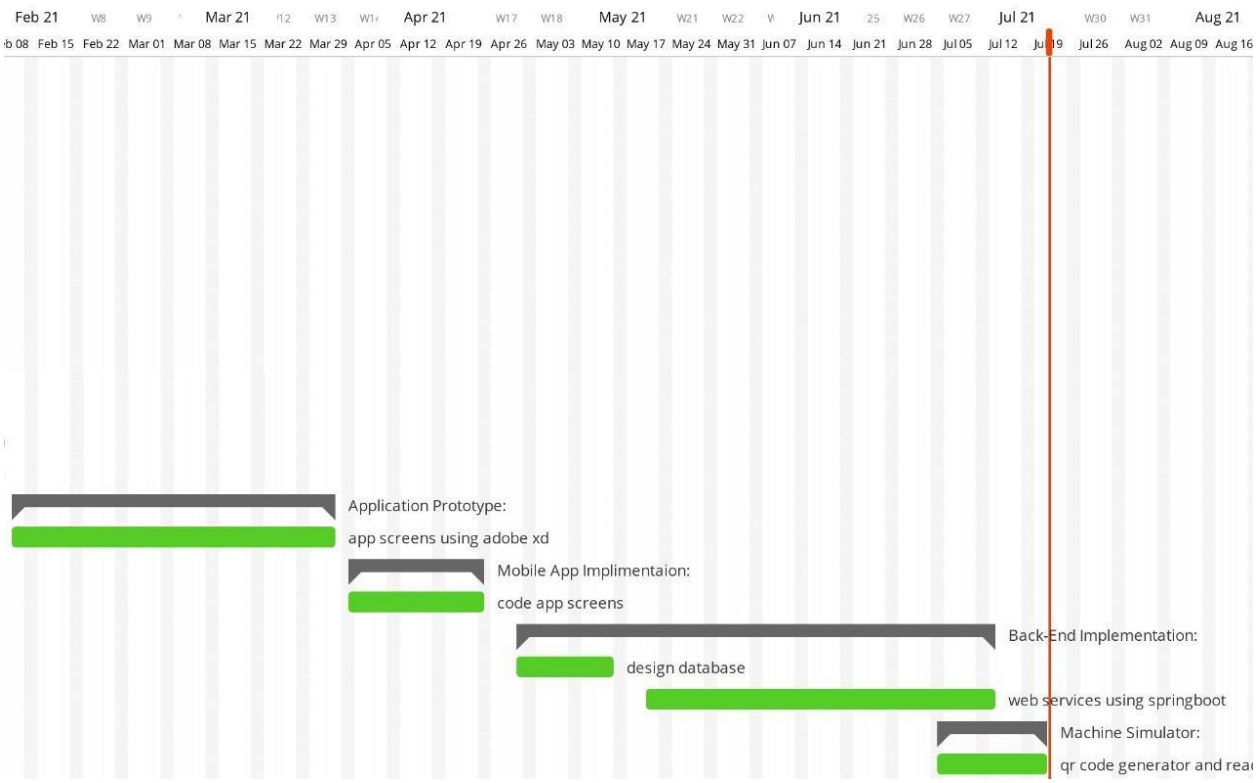


Figure 2.

1.6 Project Development Methodology

We work in a project with a combination of iterative and incremental process models with a focus on process adaptability and stakeholder satisfaction by rapid delivery of working software products, and apply (Agile Software Development Methodology).

1.7 Software and Hardware Tools

1.7.1 Software tools

In this project we have used many software tools to make this project reliable with high performance such as we have used Adobe XD to build prototype screens for the application. Beside that, we have used Dart programming language in Flutter framework to implement the actual application screens and logic.

We have built our database by Mysql via Mysql workbench.And we developed the back-end APIs by java using Spring Boot framework.

Moreover, We have used HTML, CSS, Jquery and Bootstrap to create web services for the application admin.

1.7.2 Hardware tools

We were planning to make our ticket reader like the real metro reader machine. So, We have tried to communicate with the Cairo Metro organization to gain some information about how ticket reader machines read and write on the ticket but we didn’t get any suitable answer.

In order to do that, we didn’t use any hardware tools but we have developed a simulation app to act like a reader/writer machine.

1.8 Report Organization

In the following chapters we will be talking about

- Some related works to our project:

  Examples of matching applications and the differences between us and them.

- System analysis :

  We will talk about functional, nonfunctional requirements and Use Case Diagrams.

- System design :

  UML Diagrams like Component Diagram, Class Diagram, Sequence Diagrams & ERD.

- Implementation and testing:

  System running and samples of the applied test cases.

CHAPTER 2 : RELATED WORK

There are some Applications that similar to our App, for example:

- Q Ticketing: Metro is the Metropolitan Transit Authority of Harris County, serving the Greater Houston, Texas region with safe, clean, reliable, accessible and friendly public transportation services.

  Link for the app:
  https://play.google.com/store/apps/details?id=org.ridemetro.qticketing&hl=ar&gl=US

- Metrolink: With the Metrolink App, which serves Southern California, you can securely purchase Metrolink One-Way, Round-Trip tickets, 7-Day, Monthly and Weekend Day Passes in seconds. Purchasing a ticket is easy: select your trip & ticket type, enter your payment card information, activate your ticket prior to boarding and your device is your ticket.

  Link for the app:

  Metrolink - التطبيقات على Google Play

- Ridlr App: Ridlr is a public transport ticketing and commuting app that's apt for your daily intra-city travel needs which serve Delhi & Mumbai, India.

  Link for the app:

  Metro (Delhi & Mumbai) and Bus Tickets & Passes - التطبيقات على Google Play

Our Application is similar with these Applications in having common Features like:

- Handling all the process of booking a ticket by the mobile app.
- Secure ticket purchasing.
- User device is his ticket.
- Easy to select User's origin and destination to purchase.

In Addition to other features will be added in our application like:

- Handling all the process of making a subscription for users with specific duration and number of trips.
- Some other features to make the application more usable by users like (get the nearest station and calculate estimated time for a trip).

CHAPTER 3 : SYSTEM ANALYSIS

3.1 Project Specification

3.1.1 Functional requirements

The following table illustrates functional requirements where a functional requirement defines a function of a software system or its component. Functional requirements may be technical details, data manipulation.

No.	**Functional Requirement**
1	The system shall allow users to register in the Application.
2	The system shall allow users to login in the Application.
3	The system shall allow users to charge their wallet using a payment method.
4	The system shall allow users to buy one ticket or more.
5	The system shall allow users to determine the price of a ticket using source and destination.
6	The system shall allow users to determine the closest station from a specific location.
7	The system shall allow users to determine estimated time for the entire trip (from Specific Station to another).
8	The system shall allow users to get full directions (path) to go from one subway to another.

9	The system shall allow users to apply for or renew a normal subscription.
10	The system shall allow Admins to add, update and delete basic tickets if any changes are needed.
11	There is a map showing all the subway lines.

Table 1.

3.1.2 Non-functional requirements

3.1.2.1 Performance Requirements

Using cloud server storage for our database will improve scalability as we will have the ability to increase or decrease IT resources as needed to meet changing demand.

This also will make the login information be verified within less seconds. Response time of the system will not take a long time, almost in a few seconds. The system works 24 hours per day 7 days a week. The passengers' information must be saved in the database in a few minutes after the end of registration.

3.1.2.2 Security Requirements

Using a firewall that can help protect your network by filtering traffic and blocking outsiders from gaining unauthorized access to the private data which will protect the integrity and avoid changes or access by unauthorized users.

Consideration of the security of the system has a great advantage for this system, because the database should be secured from unauthorized users. Only authorized users can get access to the database. To prevent unauthorized users,

the user should have their username and password that help them to login to the system. Additionally, the users should have to take care of their own username and password. They should have to keep it a secret manner.

3.1.2.3 Software Quality Attributes

Should be easily maintainable, using the MVC design pattern will make the system easy to upgrade and make adjustments as it is known, every system needs to be maintained and modified, so the code will be readable.

The system will check user inputs to the system to handle errors. It handles and shows errors by displaying the error message when the user enters invalid input. Our system describes the logical characteristics of each interface between the system and the users.

This may include any graphical user interface (GUI) standards or product family style guides, screen layout constraints, standard buttons and functions that will appear on every screen, error message display standards, and so on.

So, our system does all these functions in an easy and efficient way. In other words, the system is user interactive.

3.2 Use Case Diagram

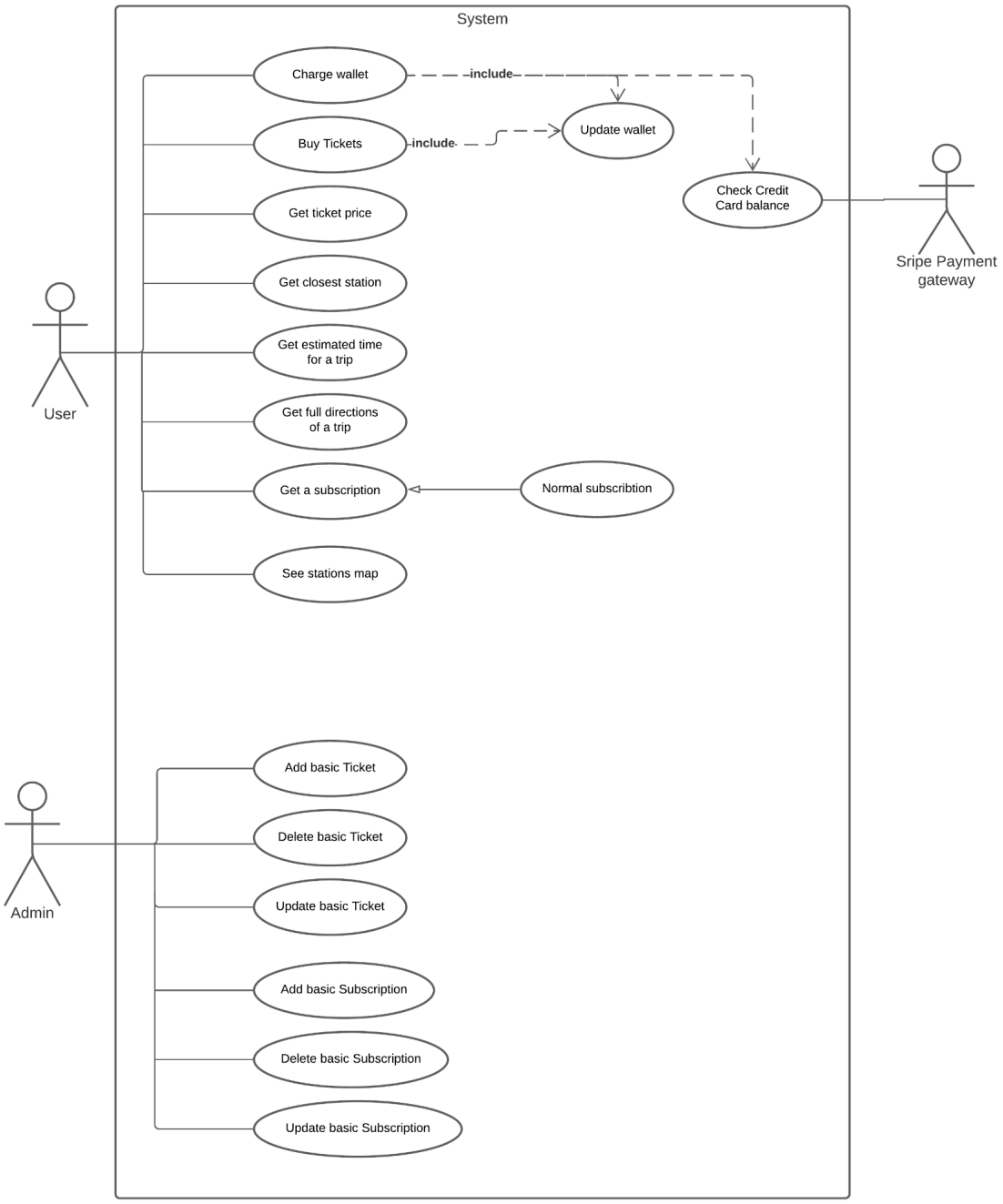
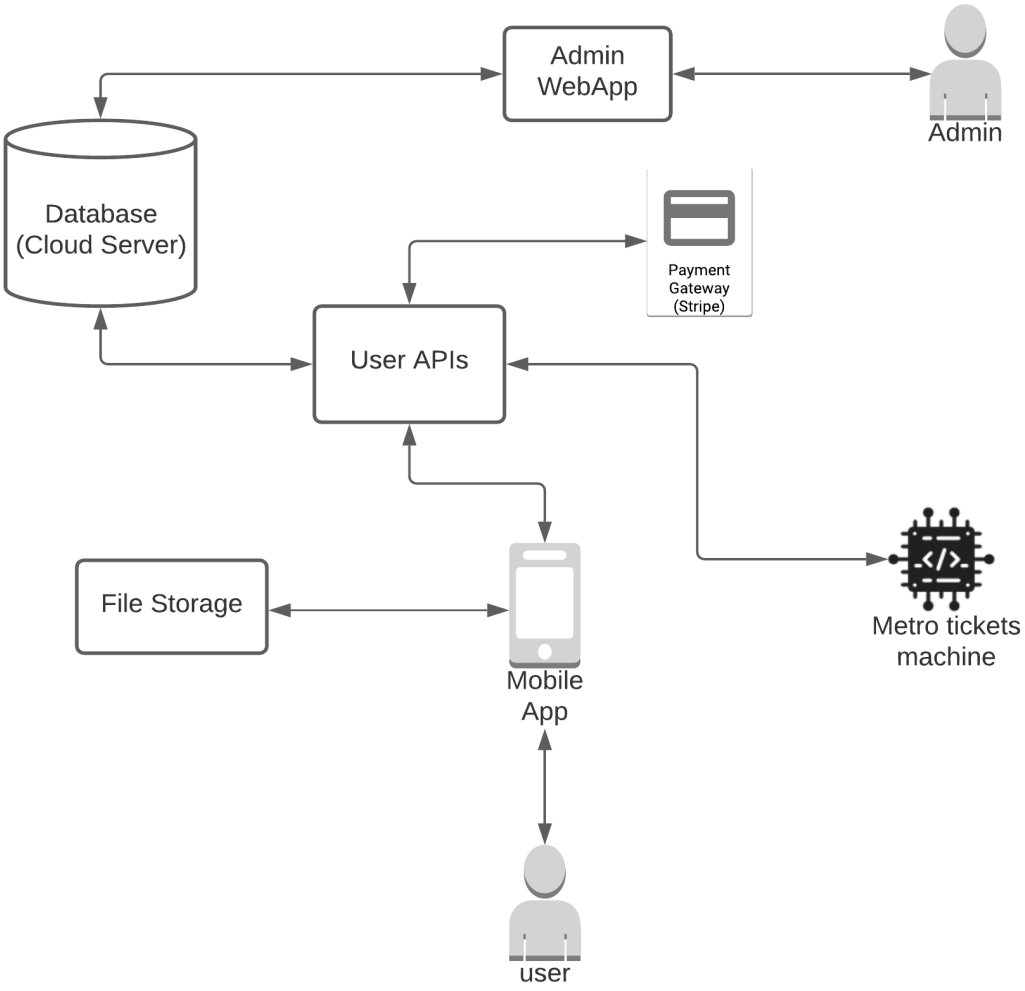


Figure 3.

CHAPTER 4 : SYSTEM DESIGN

4.1 System Architecture



4.2 System Component Diagram

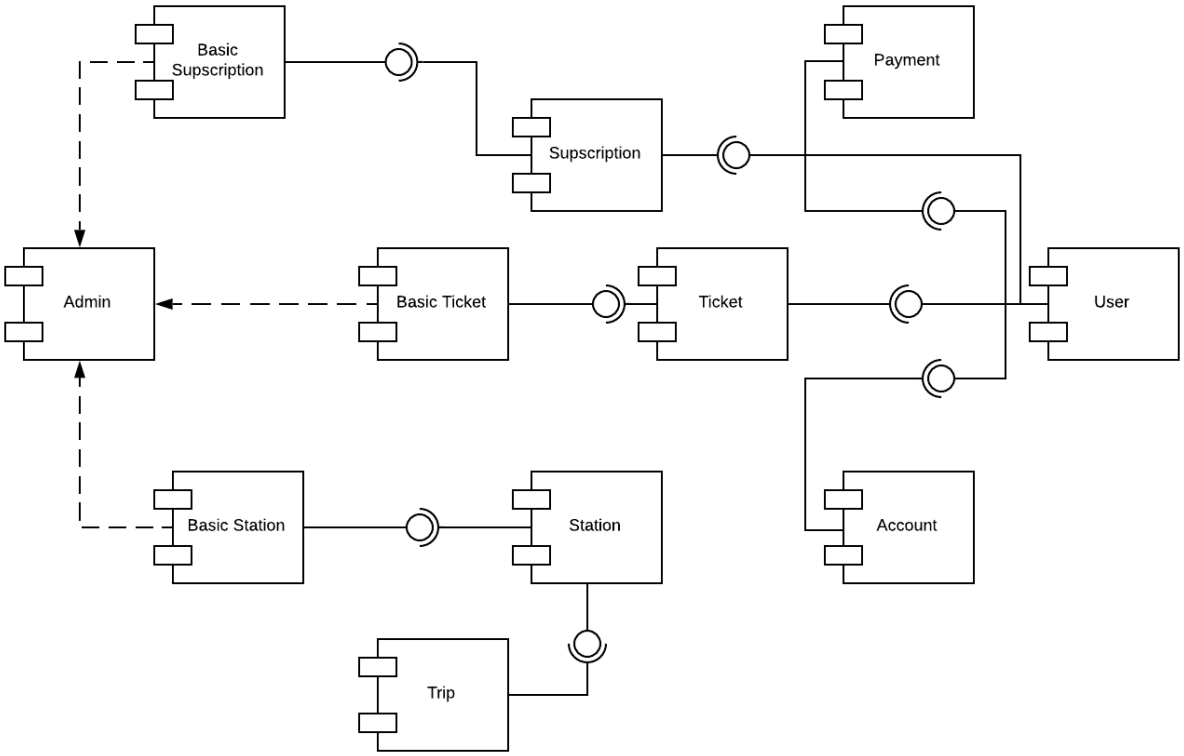


Figure 4.

4.3 System Class Diagram

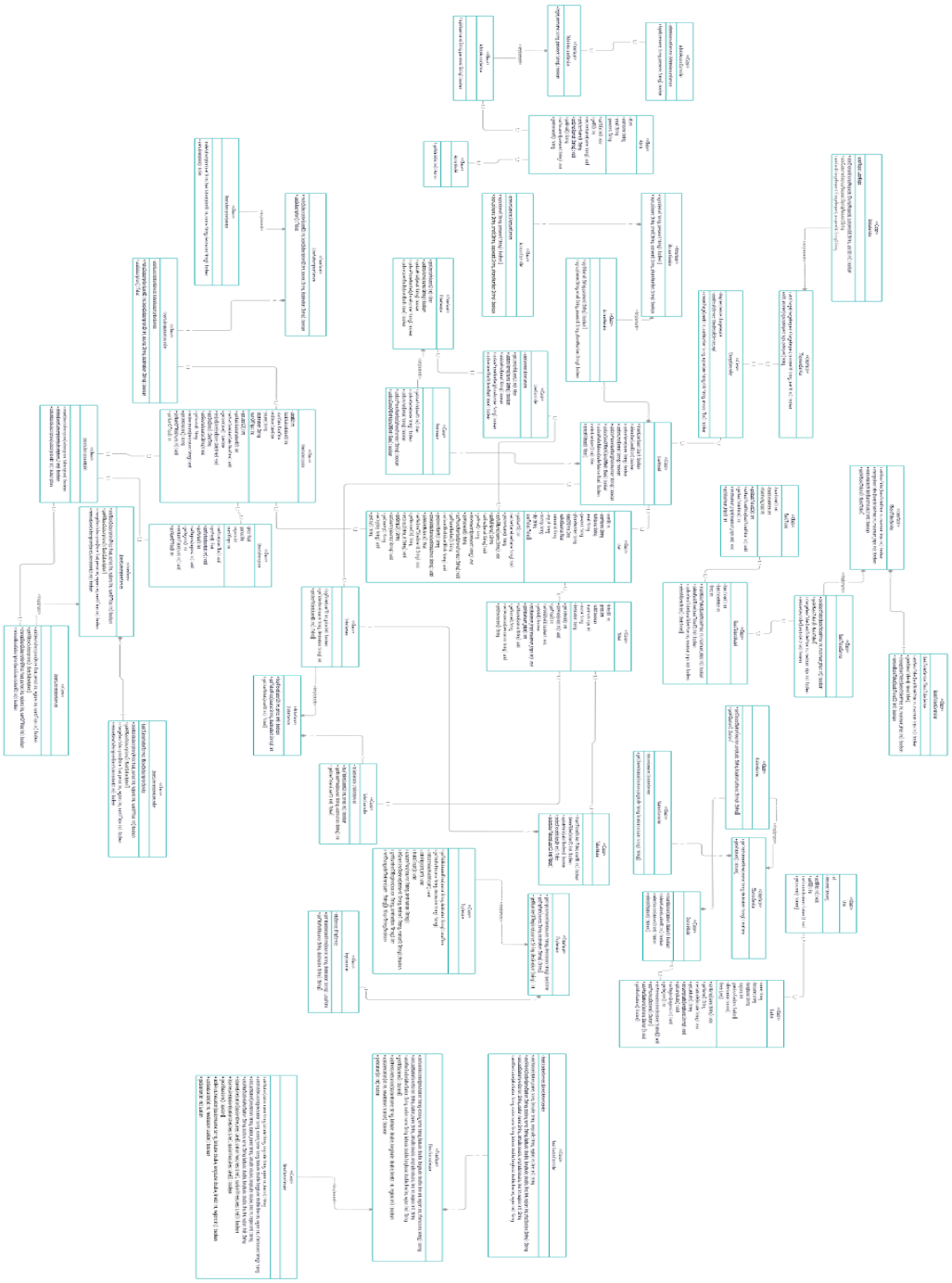


Figure 5.

Because there are many details in the diagram. here is a link to see the diagram in a better way: https://imgur.com/XxylOwL

4.4 Sequence Diagrams

4.4.1 Buy ticket sequence diagram

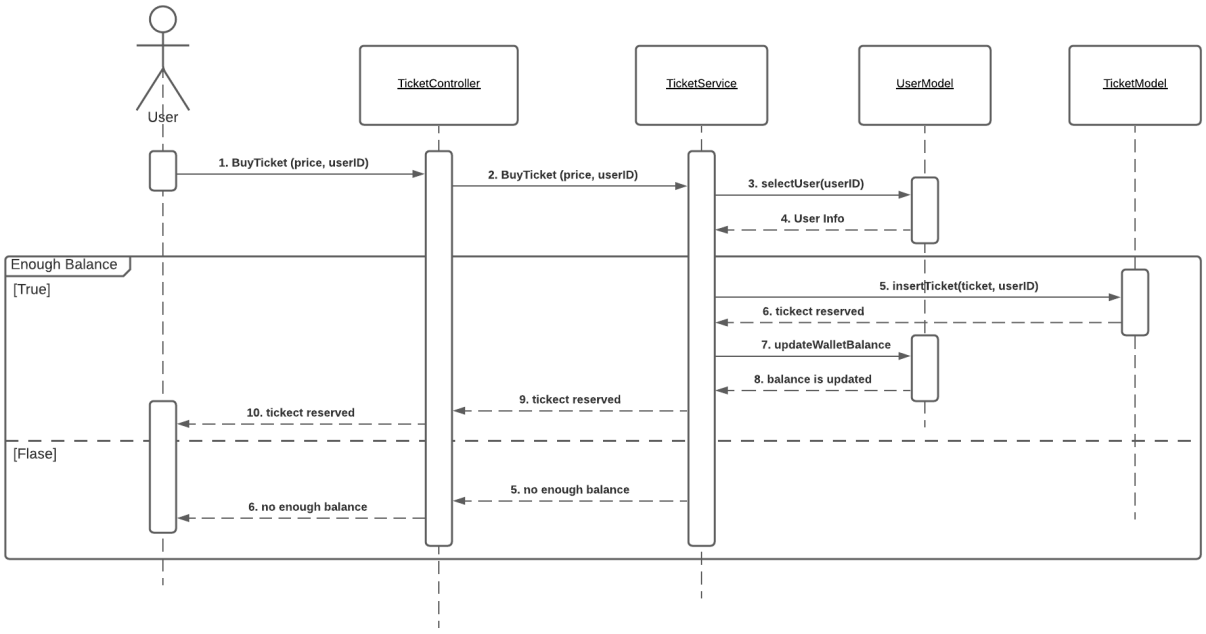


Figure 6.

4.4.2 Charge wallet sequence diagram

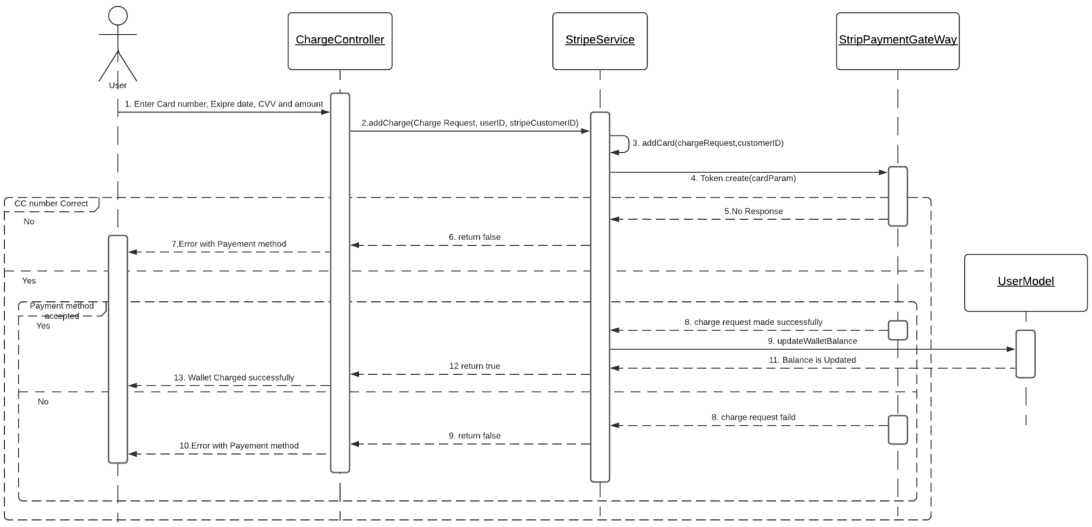


Figure 7.

4.4.3 Get closest station sequence diagram

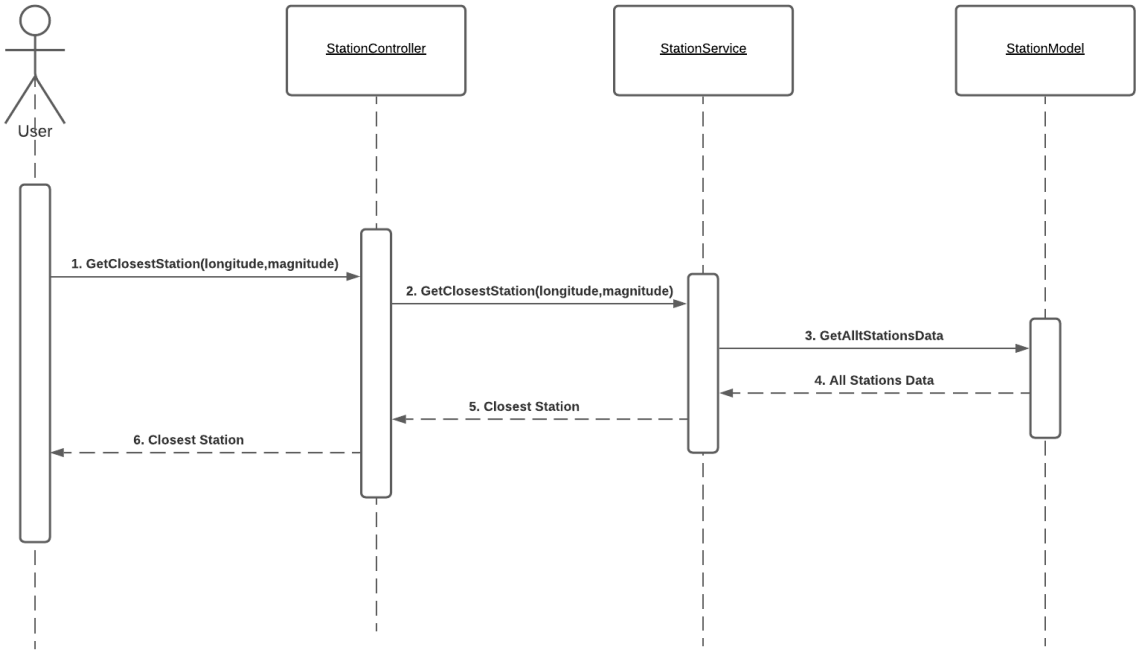


Figure 8.

4.4.4 Use subscription sequence diagram

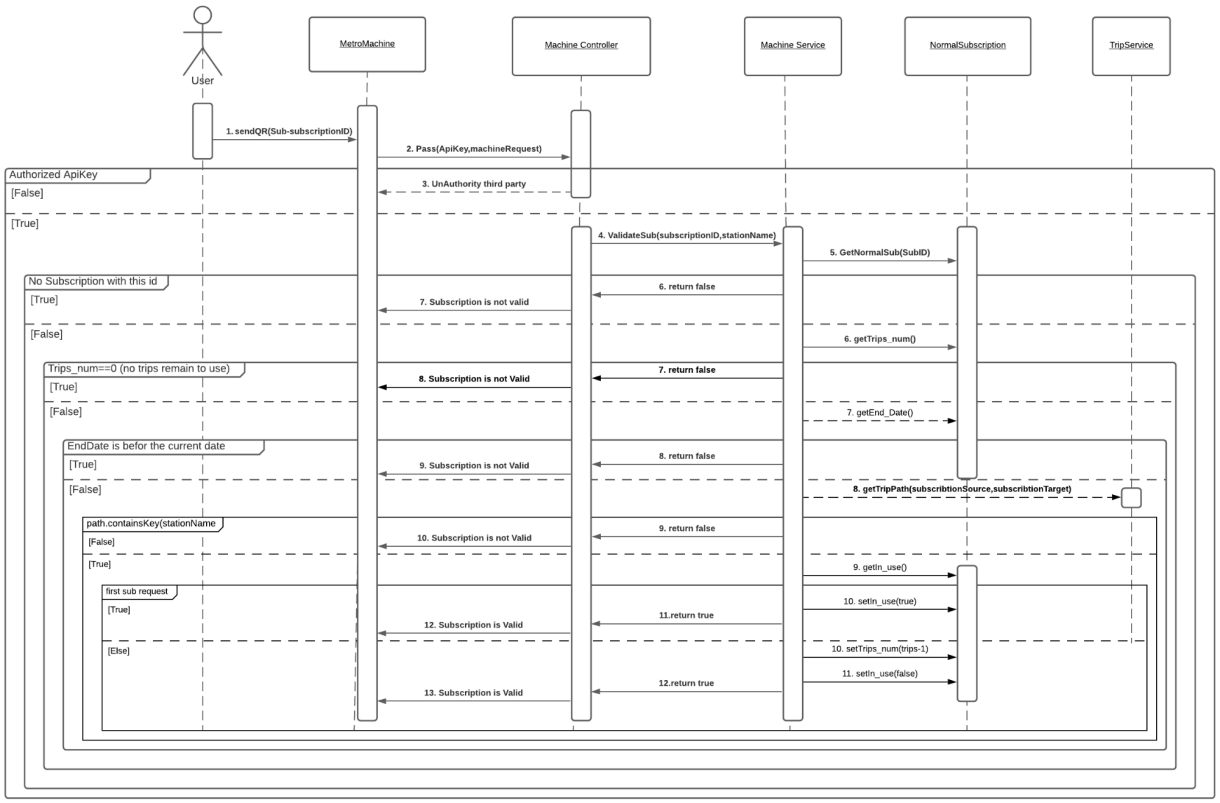


Figure 9.

4.4.5 Make subscription sequence diagram

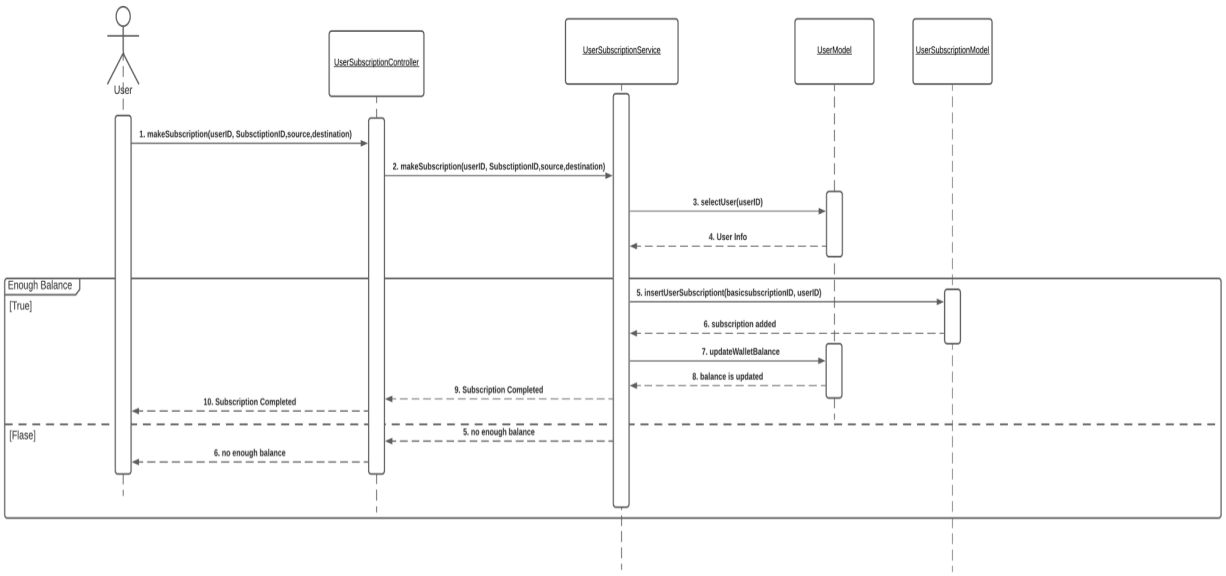


Figure 10.

4.4.6 Use ticket sequence diagram

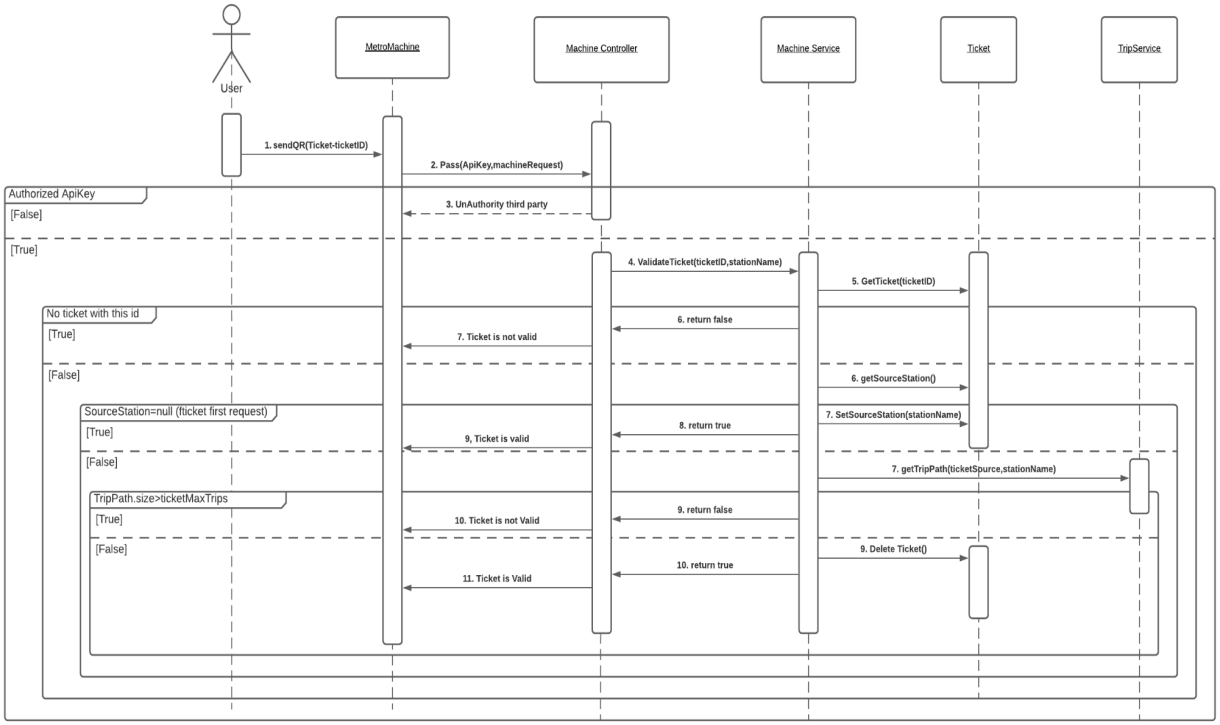


Figure 11.

4.5 Entity Relationship Diagram (ERD)

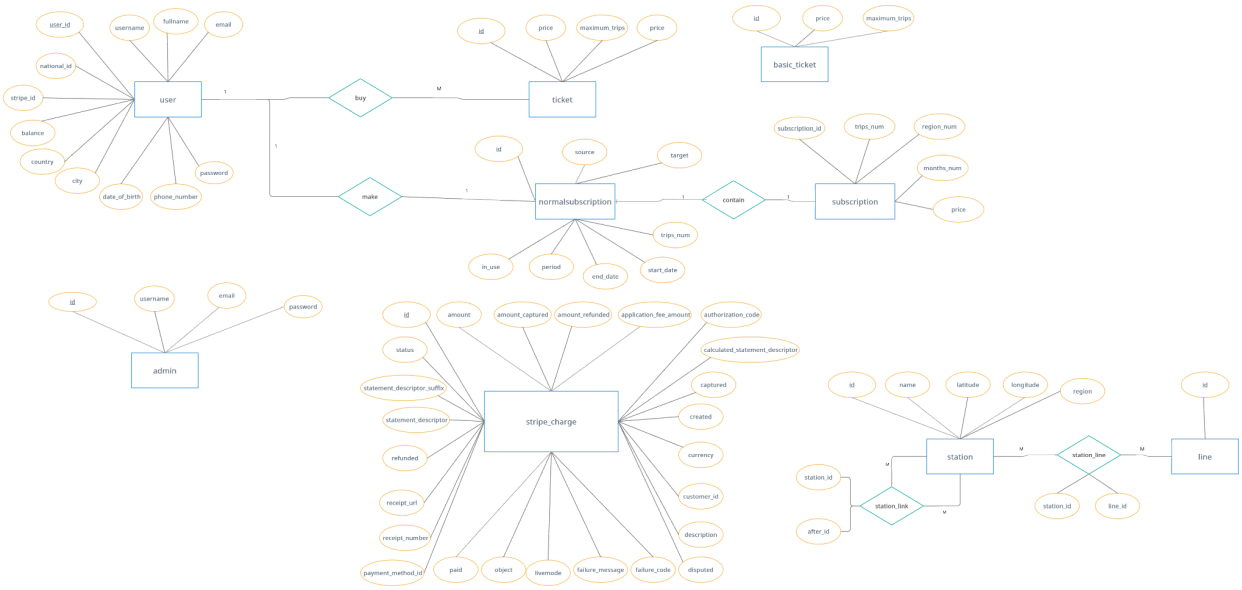


Figure 12

4.5 System Gui Design

4.5.1 Mobile app GUI

Home Page:

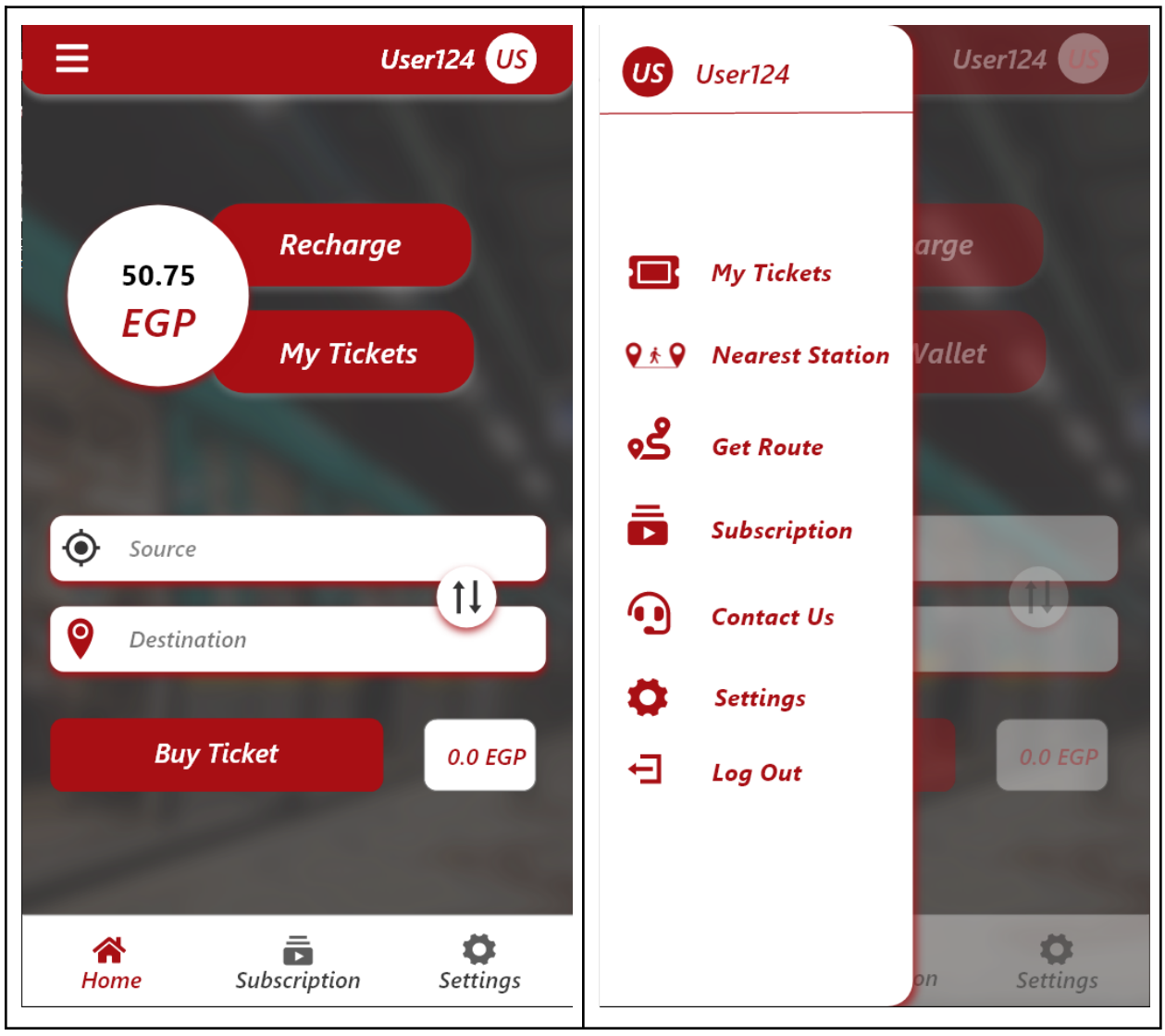


Figure 12

Login

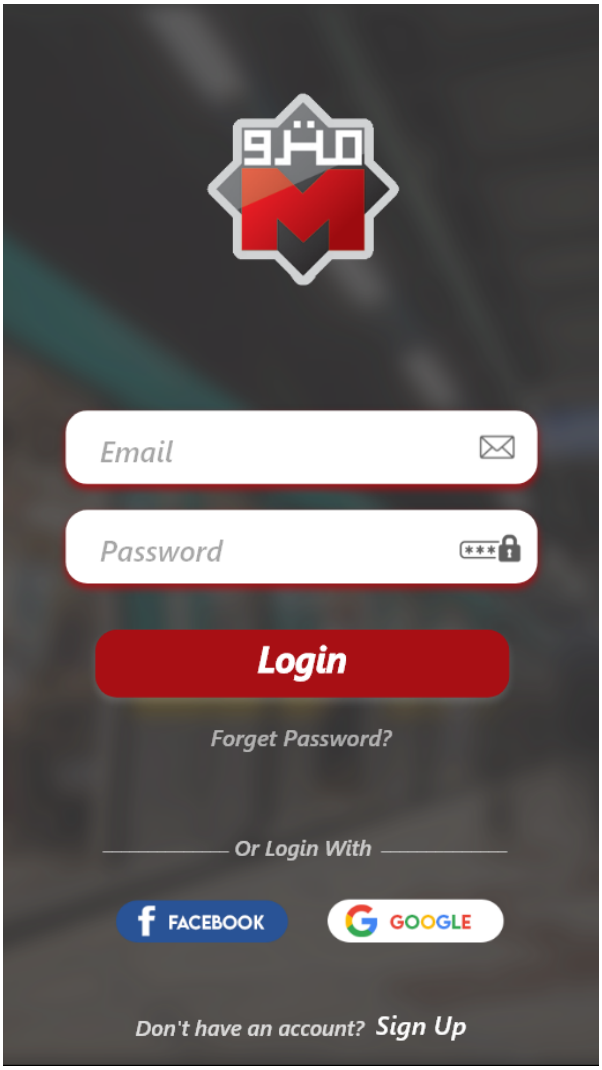


Figure 13

Settings

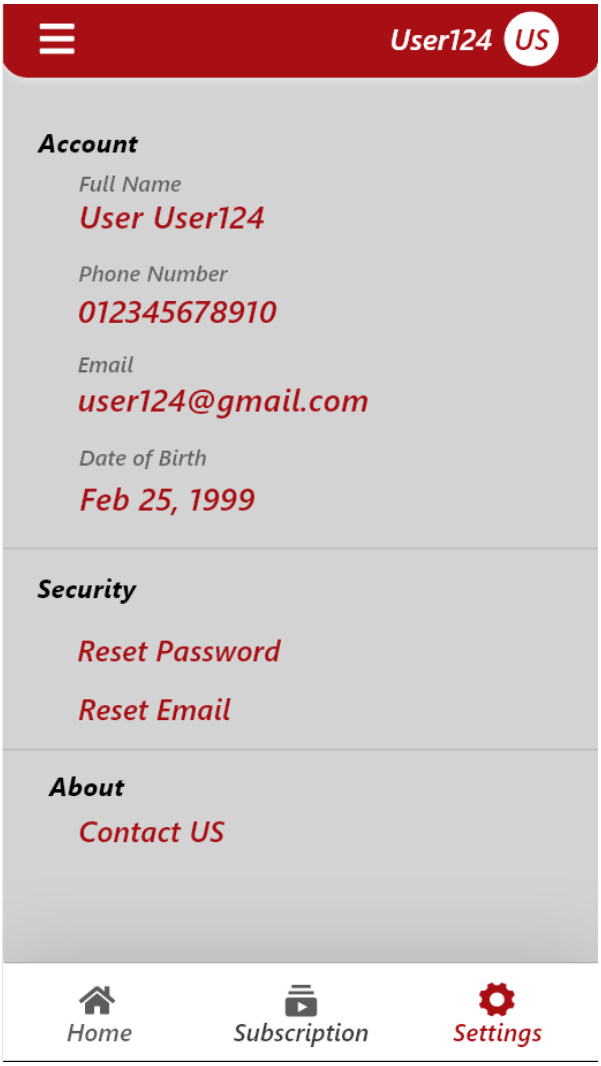


Figure 14

Buy Ticket

Figure 15

My Tickets

Figure 16

Get Nearest Station:

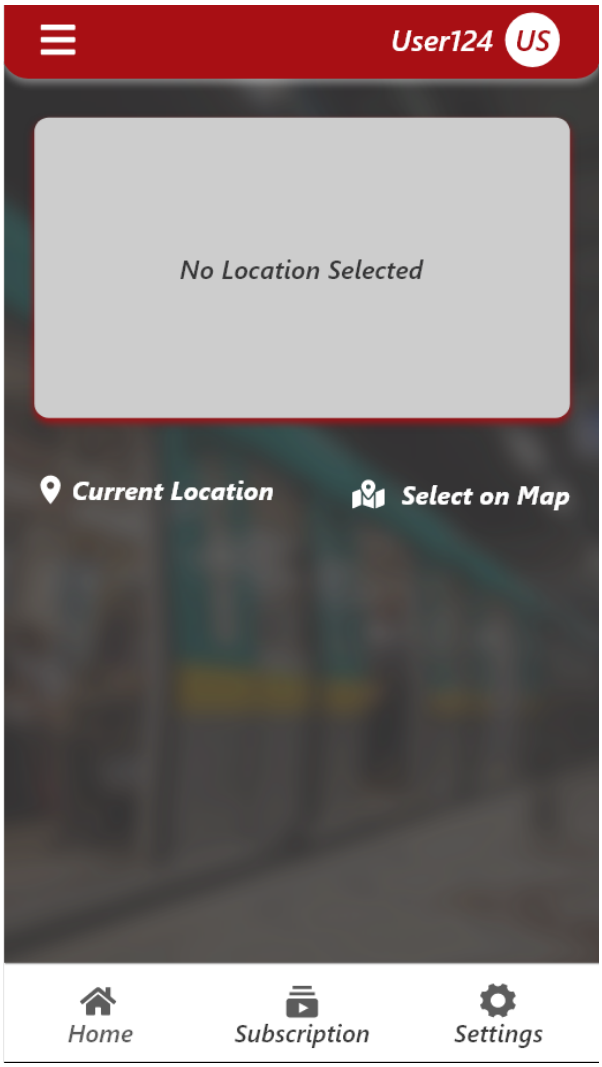


Figure 17

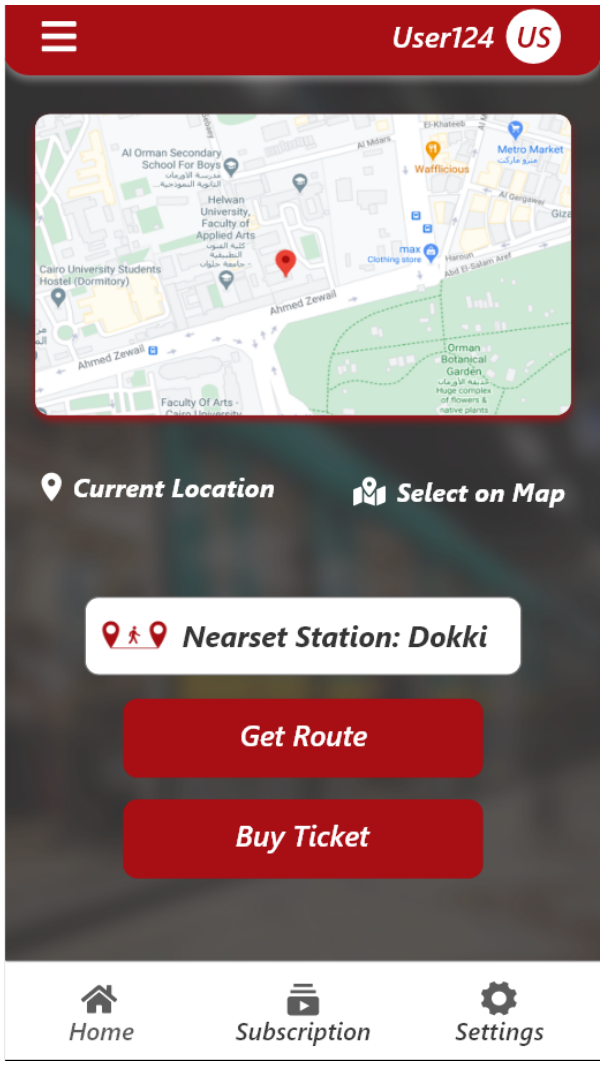


Figure 18

4.5.2 Web app GUI

Login web page:

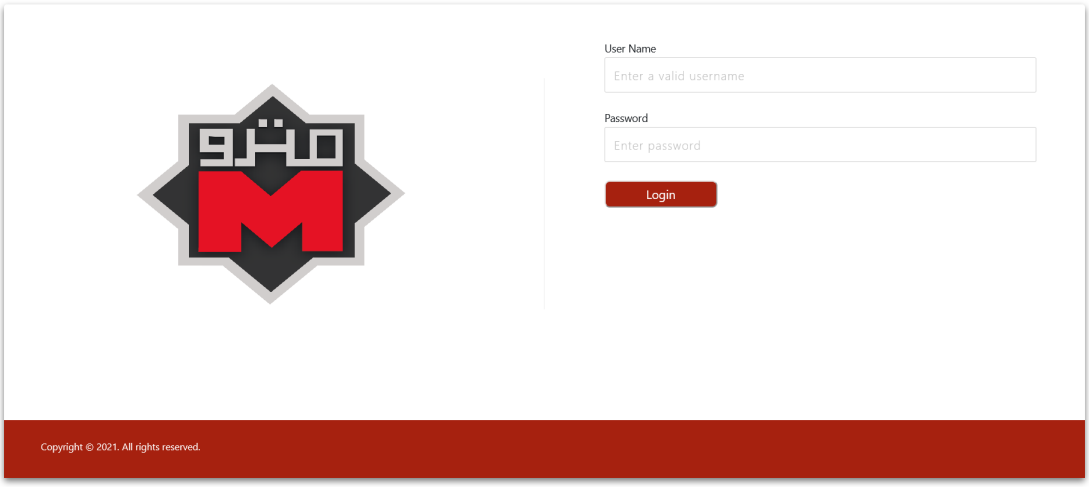


Figure 19.

Home web page:

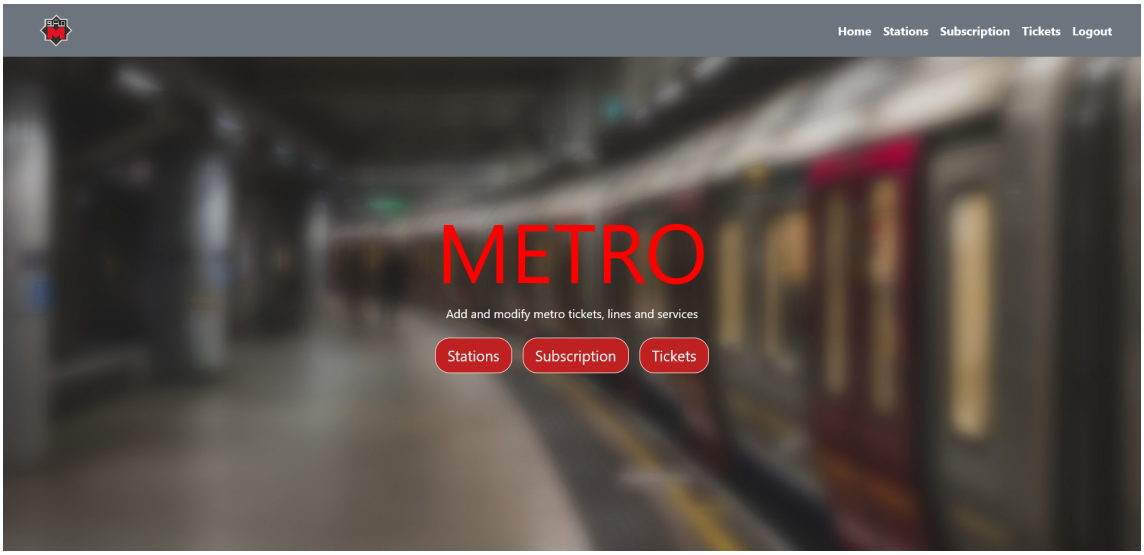


Figure 20.

Ticket web page:

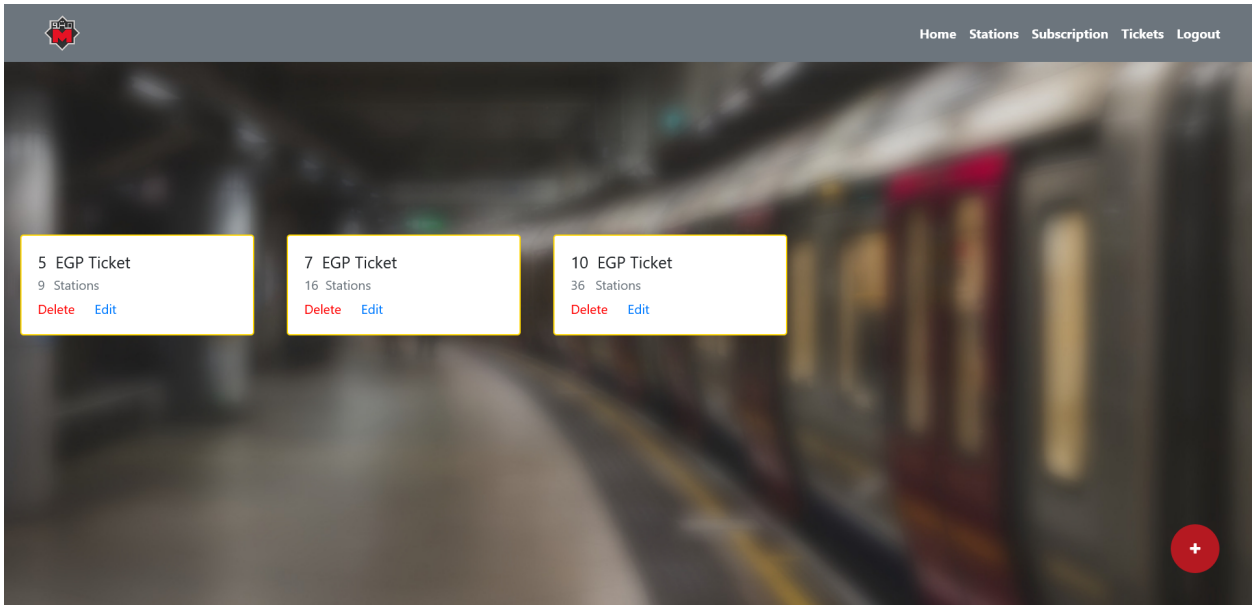


Figure 21.

Subscription web page:

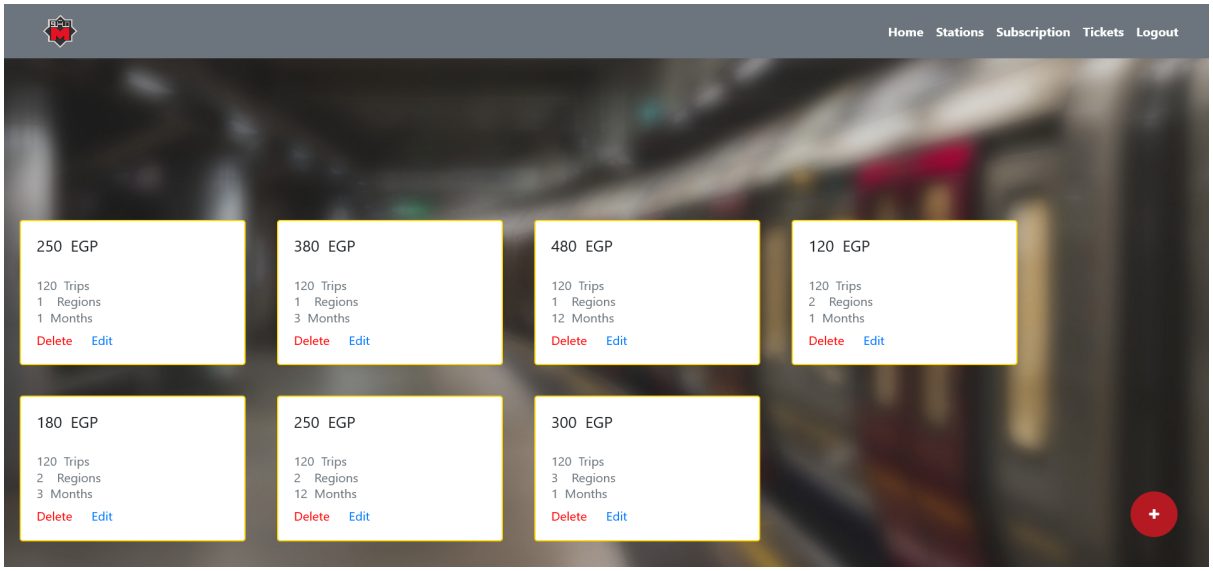


Figure 22.

Station web page:

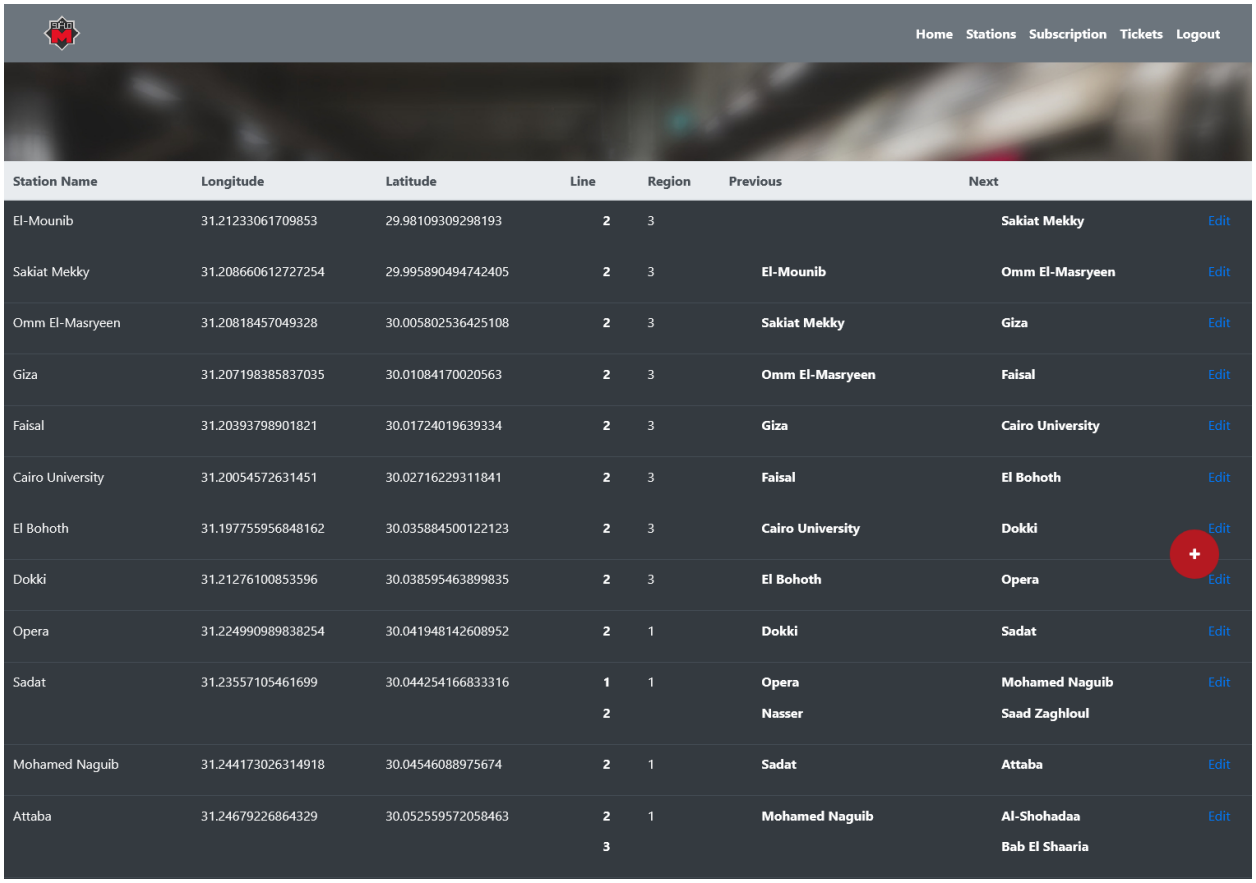


Figure 23.

Add station page:

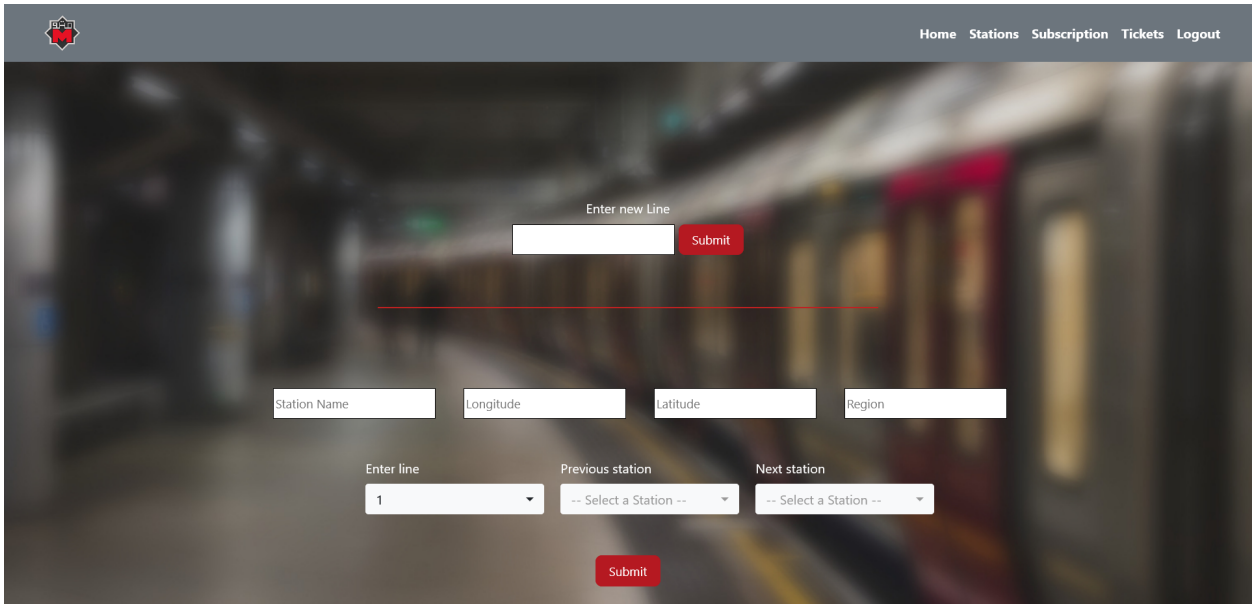


Figure 24.

Edit station page:

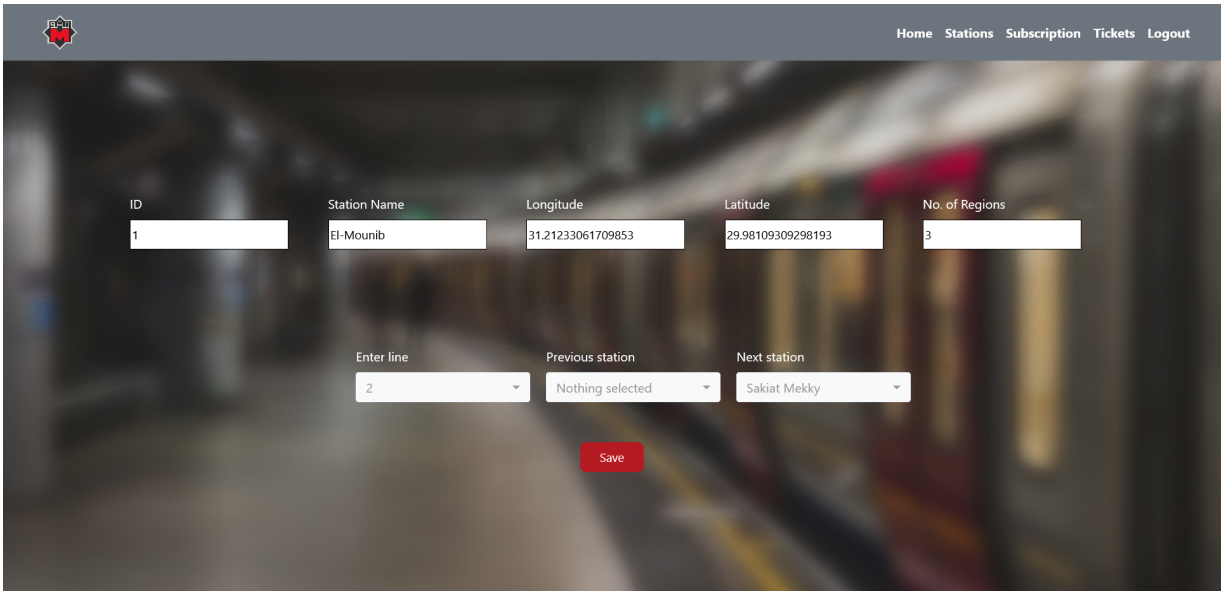


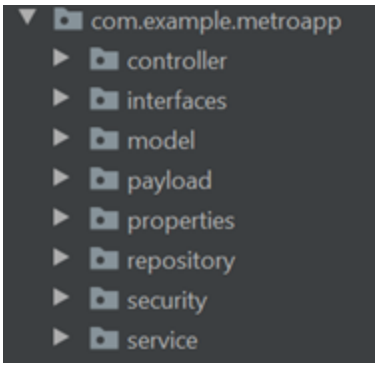
Figure 25.

CHAPTER 5 : IMPLEMENTATION AND TESTING

5.1 Implementation

Using Spring Boot framework which has been built by some design patterns like (Dependency injection and MVC Design Pattern) for developing our Back end allows our system structure to be more clarified and organized with low coupling and high cohesion within its subsystems.

As shown below:



Our system contains some packages , each package contains some classes with the same role and each one of them applies this role in its different subsystem.

Each subsystem contains from a Controller which be called by user in order to use this subsystem, An Interface for the services done by this subsystem and be called by the subsystem controller, A Service class which implement the interface and overwrite its functions, A Model represents an Entity in the database and A Repository responsible for this model in the database.

And there are also other packages used in the whole system (security and payload).

5.1.1 Subsystems for user mobile app and their main functions

5.1.1.1 Account and user

- signUP() :

This function is responsible for creating user account, it takes a signUpRequest object which contains user data needed to sign up as a parameter and check if username or email is already exist and in this case it returns BAD_REQUEST, it also creates a new customer in stripe (Payment Gateway) and store its customerID which will be used if user want to charge his wallet, then it save this account in the Database and returns HttpStatus.OK.

- login():

This function is responsible for user login, it takes a loginRequest object which contains username and password then it authenticates this user and returns user JWT token for this user or returns BAD_REQUEST if there is an error with login (wrong username or password).

- changeUserPassword(), changeUserEmail(), changeUserPhoneNum() :

Those functions are responsible for changing user data and require authentication so they take the user 's current password and the new data user wants to change , and if it is the correct password they return HttpStatus.OK , else : they return BAD_REQUEST.

5.1.1.2 Payment

- createCharge():

This function is responsible for charging user wallet, it takes chargeRequest object which contains credit card data (cardNum-Date-CVV-Amount) and create a payment using the payment gateway (Stripe is used in our project as a simulation method) and if payment is created successfully it update the user wallet and returns HttpStatus.OK , else it returns BAD_REQUEST.

5.1.1.3 Ticket

- buyTicket():

This function is responsible for buying a new ticket for user, it takes ticket price as a parameter it checks if user balance is greater or equal the ticket price and then create a new ticket for this user, insert it into Database , update user balance and then returns HttpStatus.OK , or returns BAD_REQUEST if something wrong.

5.1.1.4 Subscription

- addSubscripe():

This function is responsible for adding a new subscription for user, it takes subscription needed data as a parameter it checks if user balance

is greater or equal the subscription price and if user have not any other subscriptions and then create a new Subscription for this user, insert it into Database , update user balance and then returns HttpStatus.OK , or returns BAD_REQUEST if something wrong.

- GetSubscriptionType():

This function is responsible for getting the suitable subscription type based on source and destination stations and the period selected by the user, it takes those as parameters and checks how many regions in the path between two stations in order to return the suitable subscription type.

5.1.1.5 Station

- getClosestStation():

This function is responsible for getting the closest station for the user, it takes the latitude and longitude as parameters then checks the Data Base for all the stations and returns the closest one for him.

5.1.1.6 Trip

- initializeAndBuildGraph():

This function is responsible for creating a graph of stations (metro map) in order to get the shortest path and so on, it access the stations,line and station_line tables in Database and make the graph based on it.

- getTripPath():

This function is responsible for getting the shortest path between two stations, it takes two stations as parameters and returns a Map<String, Bool> with the station name and Boolean to check if this station is a change one (from line to other).

- getNumberOfRegions():

This function is responsible for getting the number of regions for the shortest path between two stations. It takes two stations as parameters, calculates the shortest path and then returns the total number of different regions in this path.

5.1.1.7 Machine

- Pass():

This function is responsible for Metro Machines requests, it needs an ApiKey in order to authorize this request and takes machineRequest object contains the request type (Ticket-Subscription), this request ID (ticketID-subID) and a station name where this machine exist then it call one of two functions:

1. **ValidateTicket():**

   This function is responsible for checking ticket request, it takes ticketID and stationName as parameters it checks if ticket id is right and exist in Database,

   its check this is the first request for this ticket (user getting into source station) then add current station into this ticket record and return true, if this is the second request for this ticket (user getting out destination station) then checks if the shortest path between source station and current one is greater than or equal to the maximum stations allowed then it delete the ticket record and return true, Otherwise it returns false.

2. **ValidateSub():**

   This function is responsible for checking subscription request, it takes subscriptionID and stationName as parameters it check if subscription id is right and exist in Database, it checks if the current station exist in the shortest path between subSourceStation and subDestinationStation, it checks if the endDate for this subscription is after current Date (subscription is not expired),

   it checks if the number of remaining trips is greater than zero then it checks if this is the first request for this sub (user getting into source station) then

   update this sub record (in_use = true) and return true, if this is the second request for this sub (user getting out destination station) then update this sub record (in_use = false) and decrease the number of remaining trips and return true, otherwise it returns false.

If the called function returns true: then the machine will let the user to pass Otherwise: the machine won't let the user to pass.

5.1.2 Subsystems for admin web and their functions

5.1.2.1 Admin

Responsible for Admin login and authorization before performing any function.

5.1.2.2 Basic ticket

Responsible for adding, updating and deleting ticket types (Ticket Price, Max number of Trips).

5.1.2.3 Basic subscription

Responsible for adding, updating and deleting subscription types (Subscription Price, Max number of Trips, Duration, Number Of Regions).

5.1.2.4 Basic station

Responsible for adding, updating and deleting stations.

5.1.3 User mobile application

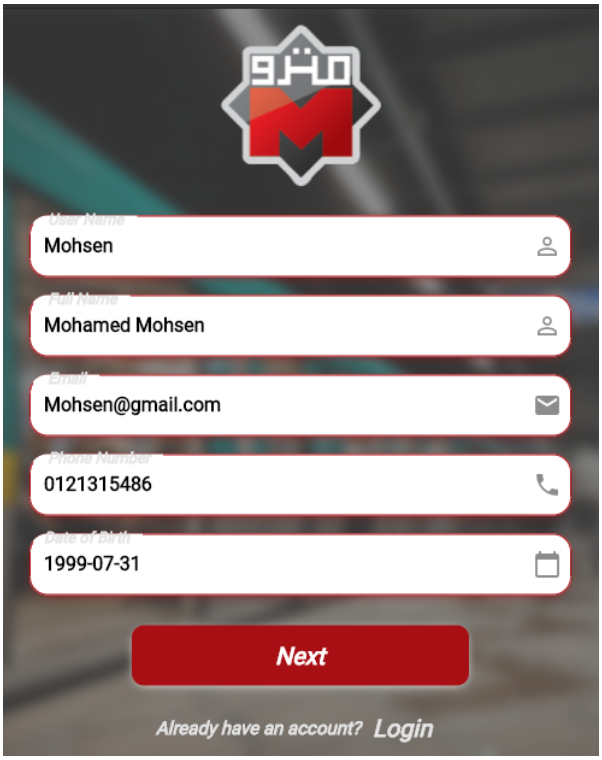
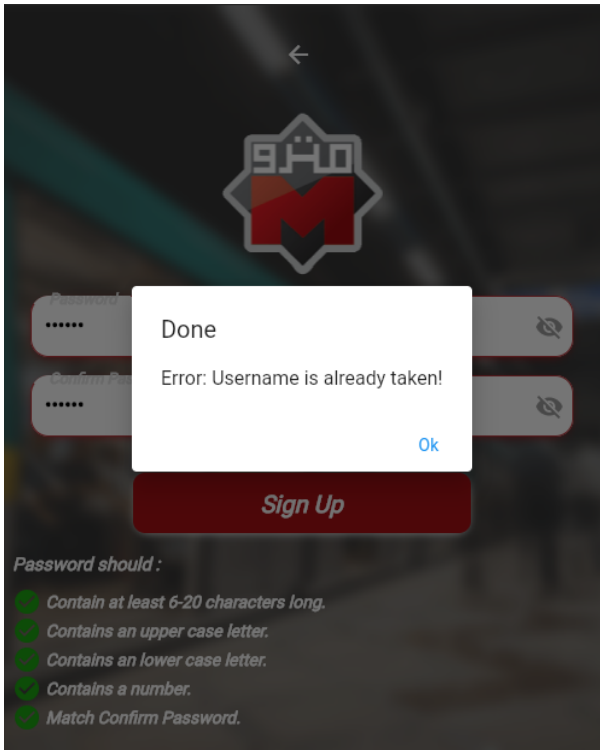
A mobile application front end developed by flutter which enables developing an app for both operating systems using the same codebase, makes the development process faster and more efficient and perform at a level compatible with native apps and are winning over other cross-platform technologies.

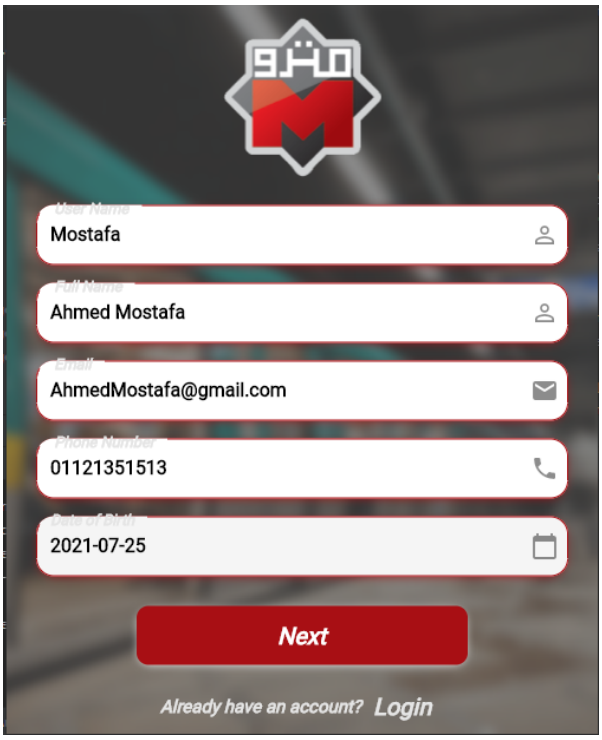
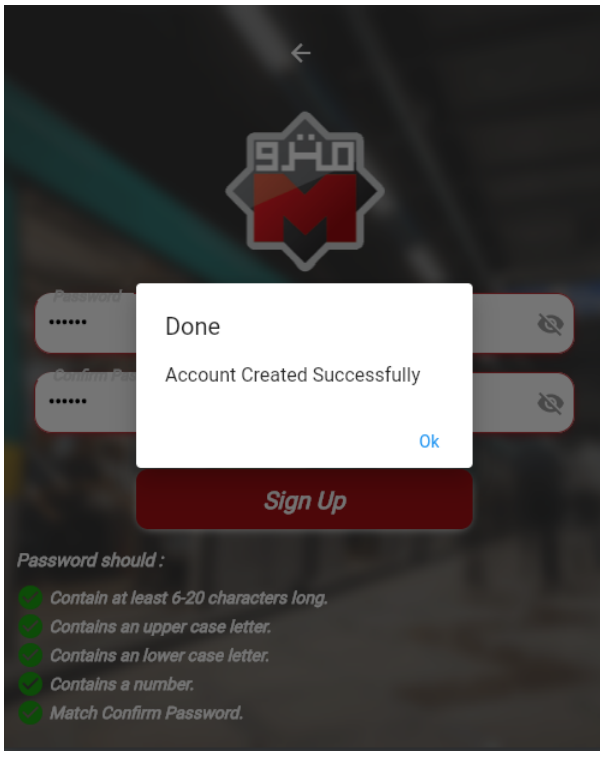
The mobile application receives user actions and sends requests to Back End Rest APIs after authorizing this user through a login request, it saves the returned JWT Token and then passes it as a header for any other request to the APIs in order to perform high security.

5.2 Testing

The following tables will show how our system interact with different test cases in details, every test case below has

simple description for both input and output and expected output screen if the input data is valid and correct.

Sign Up Test Case	
Input	**Output**
Sign up form, User should enter a unique username,his full name, valid email, valid phone number and date of birth. Then click on the “next” button and enter his password. Then click on the “Sign Up” button.	Since the username is not unique and already exists in the database, the application will return this error page.
	

Sign Up Test Case	
Input	**Output**
Sign up form, User should enter a unique username,his full name, valid email, valid phone number and date of birth. Then click on the “next” button and enter his password. Then click on the “Sign Up” button.	The system checks if the user enters correct data and then the user account will be created successfully and return this page.
 *Figure 26*	 *Figure 27*

Login Test Case	
Input	**Output**
Login form, User should his username and password Then click on the “Login” button.	The system checks if the user enters correct data and then moves him into his home page, if he enters wrong input it will show a page like the below.

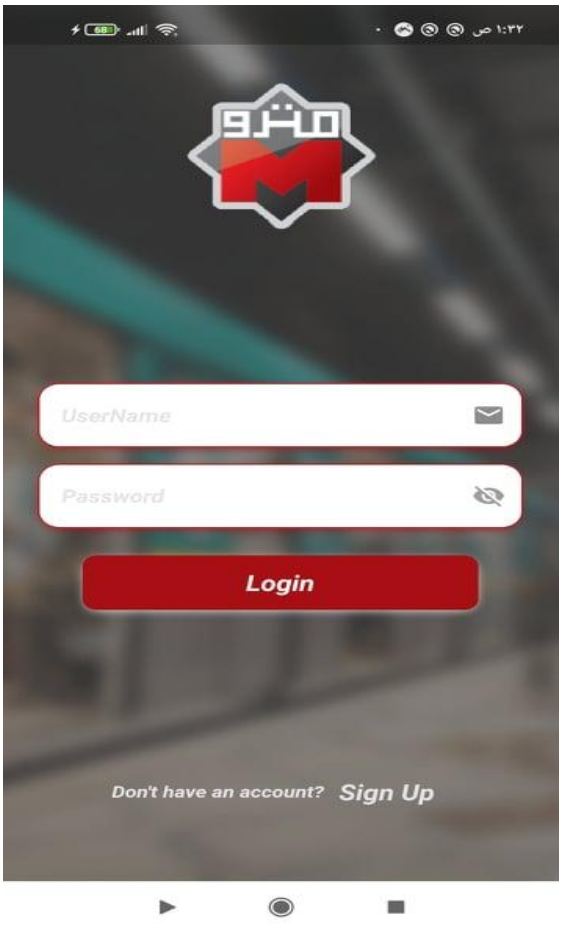
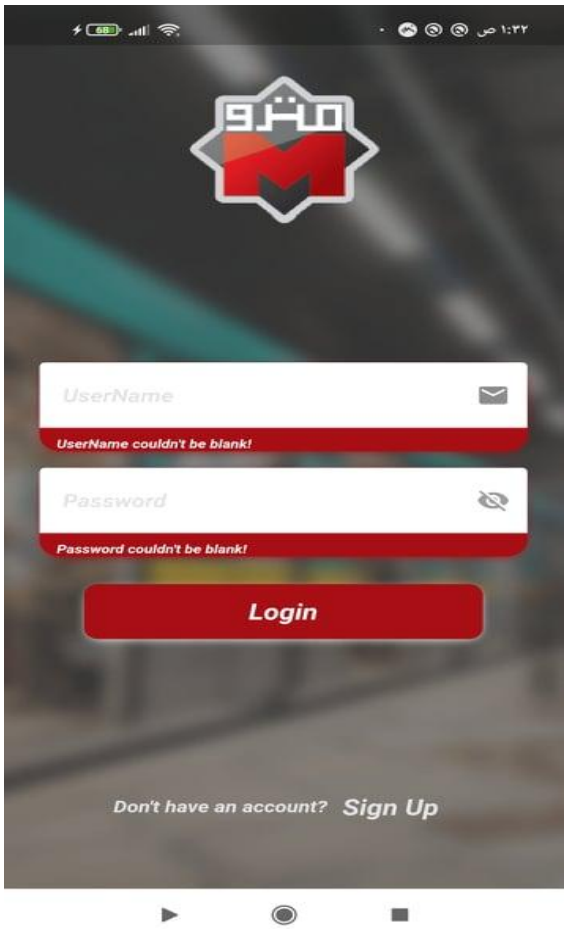

	

Figure 28	*Figure 29*

Charge Wallet Test Case	
Input	**Output**
Credit Card Details form, User should enter the amount he wants to charge his wallet with, valid credit card number, CVV number and expiration date for this card. Then click on the “Charge” button.	The system checks if the user enters correct data and then the charge will be created successfully, the user's balance will be updated and return this page.

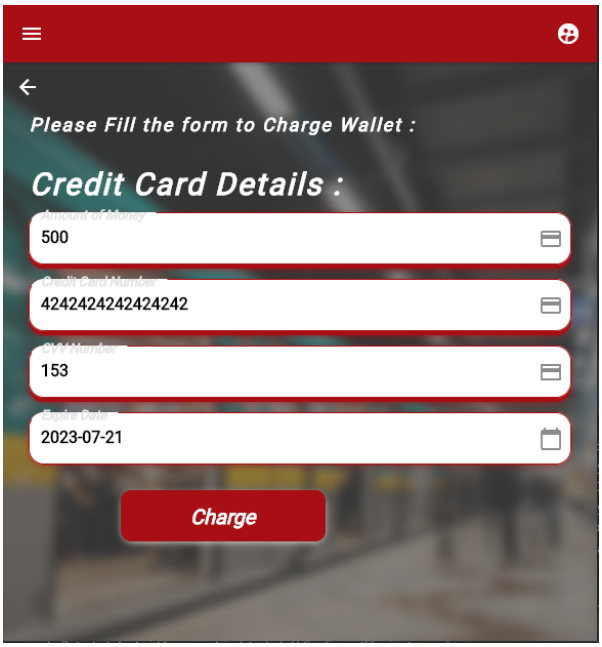
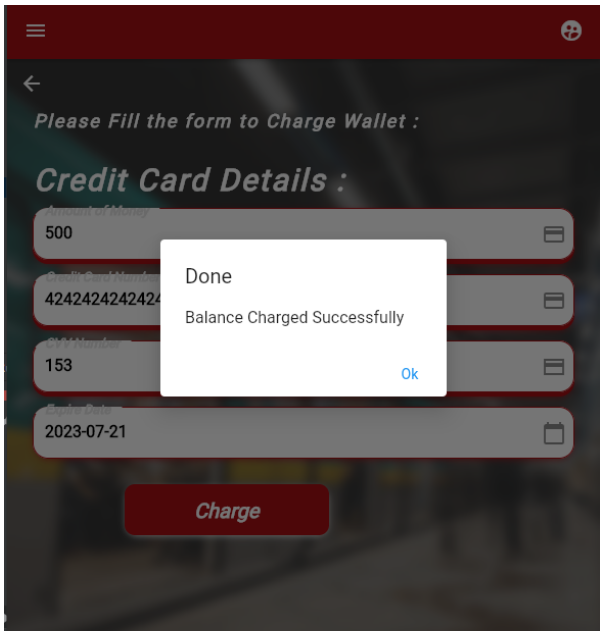

	

Figure 30	*Figure 31*

Add Subscription Test Case	
Input	**Output**
Add Subscription form, User should enter his full name, valid email, phone number, national_ID and city. Then click on the “Subscribe” button.	The system checks if the user enters correct data,he hasn't any other subscriptions and his balance is enough to make this subscription then the subscription will be added successfully for him, the user's balance will be updated and return this page.

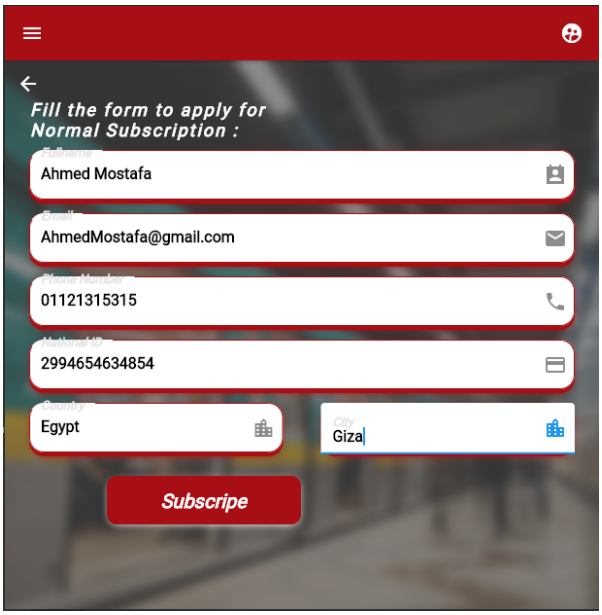
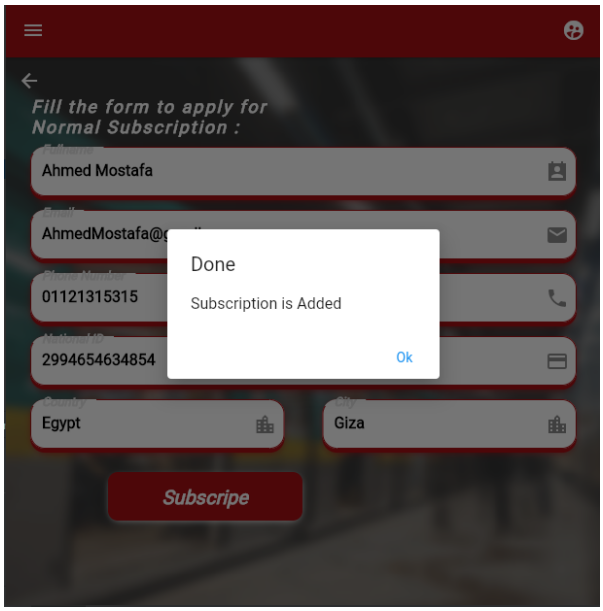

	

Figure 32	*Figure 33*

Buy Ticket Test Case	
Input	**Output**
Buy Ticket Page,User should select one of ticket types. Then click on the “Buy” button and confirm this process.	The system checks if the user's balance is enough to buy this ticket then the ticket will be added successfully for him, the user's balance will be updated then move to this page.

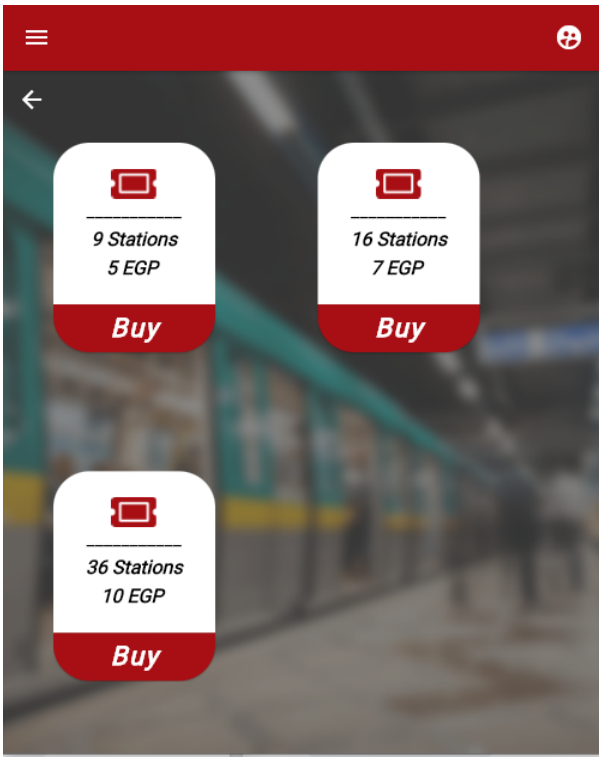
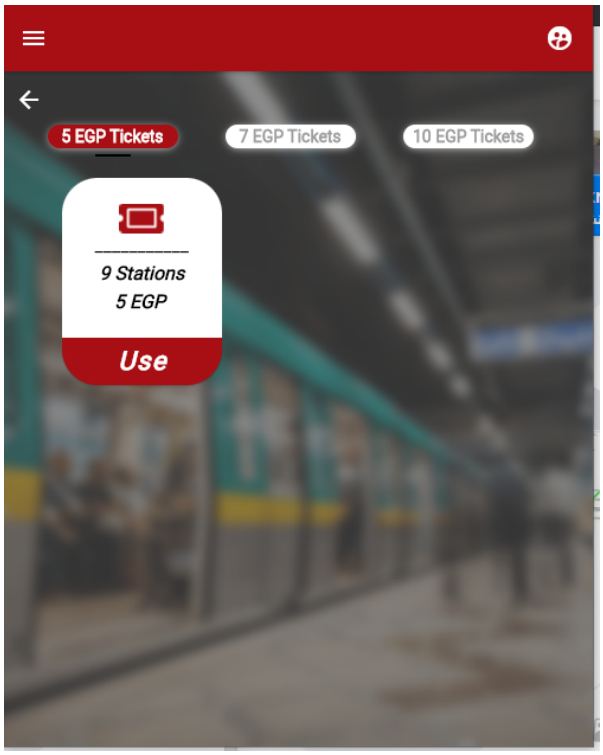

	

Figure 34	*Figure 35*

Get routeTest Case	
Input	**Output**
Get Route Page,User should select any two stations from list of stations. Then click on the “Get Route” button.	The system will compute the shortest route between those two stations and will return the first station , change line station (if it exists) and the destination one. (user also can view the full route between those stations).

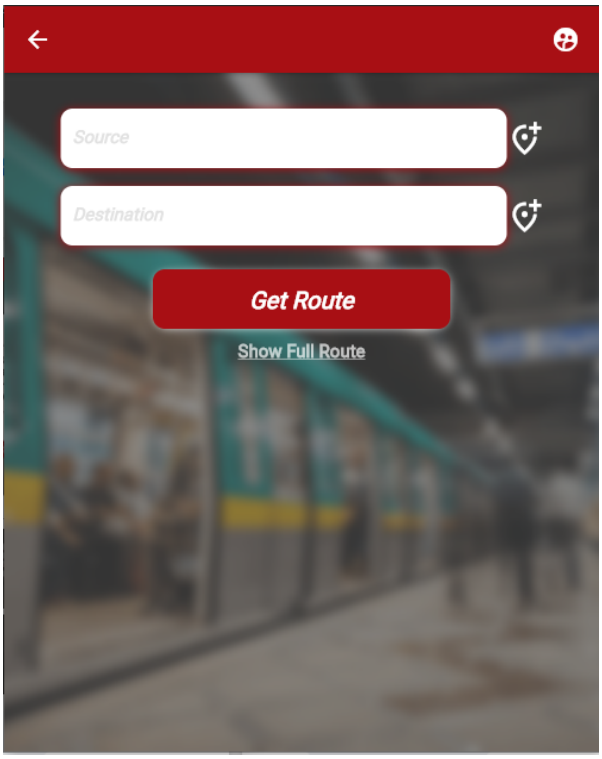
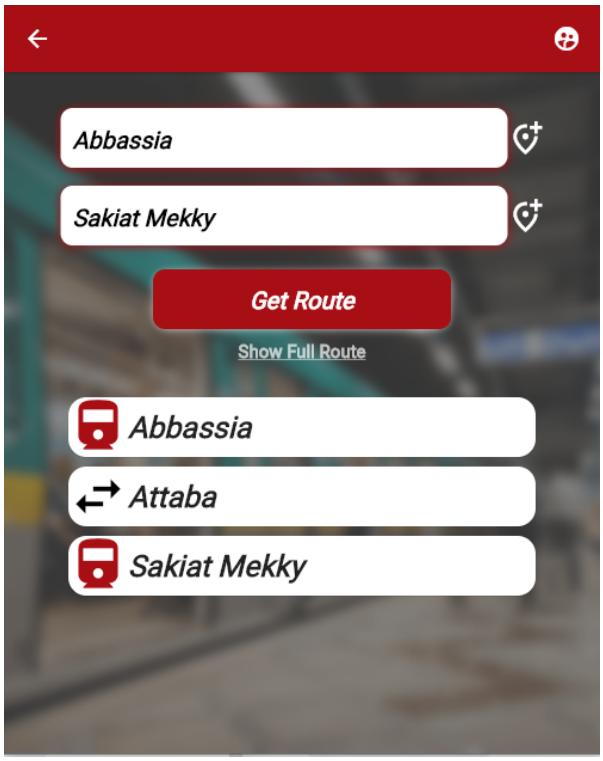

	

Figure 36	*Figure 37*

References

- **Statistics resources**

  https://www.bankmycell.com/blog/how-many-phones-are-in-the-world#:~:text=March%202021%20Mobile%20User%20Statistics,world's%20population%20owning%20a%20smartphone.

  https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast

  https://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/

- **Flutter Tutorial**

  https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/

- **Spring Boot Tutorials**

  https://www.tutorialspoint.com/spring_boot/index.htm

- **Spring Boot JWT Tokens**

  https://www.bezkoder.com/spring-boot-jwt-authentication/

- **Stripe Docs**

  https://stripe.com/docs

- **Tickets and Subscriptions Informations**

  https://cairometro.gov.eg/ar/bookings/4

- **Migrate MySQL DataBase on Azure Cloud Service**

  https://www.youtube.com/watch?v=uxSDpZnFa18