POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Trenord Improvements App Design Document

Author(s): **Poli Dimieva - 11078140**

Gagan J H - 10881210

Academic Year: 2024-2025
Version: 1.0
Release date: 04/02/2025

Contents

1 | Introduction

1.1. Purpose

The purpose of this document is to present a detailed design description of the Trenord Improvements App. We are a team of two students from Politecnico di Milano who developed this application as a proposal for additional functionalities to enhance the existing Trenord app. Our goal is to improve the user experience by introducing features that facilitate real-time train tracking, location-based advertisements, and emergency contact options.

1.2. Scope

The Trenord Improvements App is designed to enhance the existing Trenord application by introducing additional functionalities that improve the travel experience for commuters. The system aims to provide real-time train tracking, seamless digital ticket management, and an integrated emergency contact feature. By leveraging live updates on train schedules, platform changes, and delays, the app ensures that users have access to accurate and up-to-date travel information. To mock the current state of the Trenord App this version also enables the storage and retrieval of digital tickets, making ticket validation and journey management more efficient. The addition to this is the inclusion of an emergency contact system allows users to quickly share their live location with a designated contact in case of need. The design outlined in this document establishes a solid foundation for implementing these features while adhering to user requirements and improving overall commuter satisfaction.

1.3. Definition, Acronyms, Abbreviations

Acronyms	Definition
DD	Design Document
User	All people who downloaded Trenord app
API	Application Programming Interface
RQX	Requirement X

Table 1.1: Acronyms used in the document.

1.4. System Requirements

Requirement ID	Description
RQ1	The system shall support user registration.
RQ2	The system shall support user authentication via email confirmation.
RQ3	The system shall provide a login functionality.
RQ4	The system shall provide a logout functionality.
RQ5	The system shall allow users to contact an emergency contact through the app.
RQ6	The system shall enable users to send an emergency SMS with their live location.
RQ7	The system shall allow users to send emergency messages via WhatsApp.
RQ8	The system shall mock the functionality of users purchasing train tickets directly from the app.
RQ9	The system shall provide a digital wallet for storing purchased tickets.
RQ10	The system shall mock the functionality of purchasing carnet tickets through the application.
RQ11	The system shall allow users to view their ticket purchase history.
RQ12	The system shall send real-time push notifications in case of train delays.

RQ13	The system shall send real-time push notifications when a train platform changes.
RQ14	The system shall send real-time push notifications when a train is canceled.
RQ15	The system shall send location-based push notifications with relevant advertising deals for places near their current location.
RQ16	The system shall send welcoming push notifications when users arrive at a train station before their departing train.
RQ17	The system shall allow the user to see an overview of all important notifications sent to them regarding trip udates.
RQ18	The system shall provide an admin interface for managing train schedules.
RQ19	The system shall allow admins to update train departure times.
RQ20	The system shall allow admins to update train platform assignments.
RQ21	The system shall allow admins to cancel a train trip when necessary.
RQ22	The system shall allow the user to get directions to the closest station to their current location through Google maps.

1.5. Features Implemented

The functionalities implemented in the Trenord Improvements App, structured in an order reflecting user interaction, are as follows:

1.5.1. Login and Registration

Users can sign up using an email and password, with an email confirmation step to ensure secure authentication. The login functionality allows returning users to access their accounts securely.

1.5.2. Emergency Contact System

Users can store emergency contact details within the app. In case of an emergency, the app allows users to send an alert via SMS or WhatsApp, including their real-time location,

to their predefined emergency contact.

1.5.3. Ticket Purchase and Management

Users can purchase train tickets directly within the app by selecting desired destinations, date and time.

1.5.4. Ticket History and Wallet

The app provides a dedicated wallet feature where users can view a summary of their purchased tickets and their travel history. Users can quickly access ticket details and track their past journeys from here.

1.5.5. Real-Time Travel Notifications

Users receive push notifications in real time regarding:

- Train delays.
- Platform changes.
- Train cancellations.

These alerts ensure that passengers stay informed about their travel plans without needing to check external sources manually.

1.5.6. Location-Based Notifications

The app includes a location-based notification system that provides:

- Promotional deals and recommendations for places to visit near the user's current location.
- Welcome messages upon arrival at major train stations, enhancing the passenger experience.

1.5.7. Admin Trip Management

A dedicated admin panel allows authorized personnel to manage train schedules efficiently. Admins can:

- Update train departure times.

- Modify platform assignments.
- Cancel train trips when necessary.

This feature ensures that train schedule changes are reflected instantly in user notifications.

1.5.8. Location-Based Directions

The app includes integration with Google Maps to show directions to the user to the closest station where they could use Trenord's services.

Each of these features has been designed to improve the overall user experience, ensuring efficiency, reliability, and ease of use for both passengers and Trenord administrators.

1.6. Document Structure

The document is structured into seven sections, each detailing key aspects of the Trenord Improvements App.

Introduction. This section outlines the purpose of the document, defines key terms, and describes the scope of the Trenord Improvements App. It provides an overview of the system's intended functionalities and the motivation behind its development.

Architectural Design. The second section describes the system's architecture, detailing its core components and their interactions. It discusses the design choices, data management, and data flow within the application.

User Interface Design. This section presents the app's user interface, including mockups of the main screens and explanations of their functionalities. It ensures that the design aligns with user expectations and enhances the user's travelling experience.

Implementation, Integration, and Test Plan. This section describes the implementation process, the integration of different system modules, and the testing strategies used to validate the system's performance and reliability.

Effort Spent. The sixth section documents the distribution of work between the two developers, detailing the effort put into different aspects of the project, including research, design, and implementation.

References. The final section provides a list of all the sources and reference materials consulted during the development of this document.

2 | Architectural Design

2.1. Architectural Design

The app is developed using React Native, an open-source framework developed by Facebook, with Expo, a framework that provides file-based routing. Its unique quality that allows developers to build native-style applications for iOS and Android using JavaScript helps designing a platform-specific user-friendly design from a single source code. The application is designed to look similar to the existing Trenord application with a more focus on adding functionality to it.

2.1.1. Overview

A number of services are used for the development of the application. They can be mainly categories into

- System Services
- Storage Services
- React-Native Services
- Expo Services
- External Services

Data Model

Data managed by the application is structured around the following conceptual interfaces of Typescript.

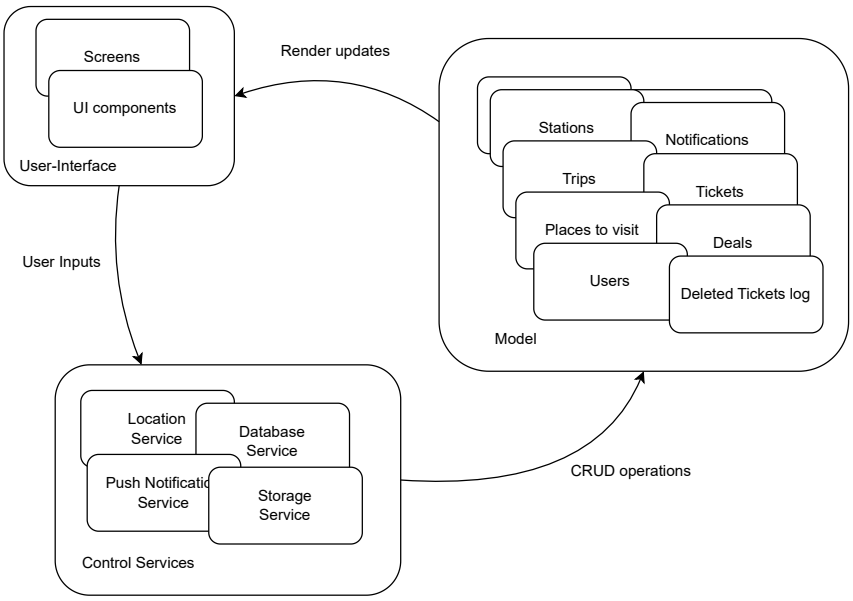

Figure 2.1: The data model adopted in the application.

- User - includes the information of the user. It contains fields like email, First and Last name, Gender, Location, etc.
- Trips - includes information about the routes traversed by the train, the time of departure and arrival.
- Stations - includes information about the stations, their geo-location.
- Deals - includes details about the deals specific for each station
- Tickets - includes information about the tickets purchased, the user that purchased it, the departure and destination details.
- Places to visit - contains the name description and of attractions.

Data is fetched from the Database and it is parsed to the app for rendering the information in the application. The data is stored in the form of tables in the and created, read, updated, and deleted using queries from the application.

Data Management

Supabase is an open-source alternative to Firebase. It is built on top of Postgres, a powerful and reliable open-source relational database. The data store in supabase is organized as tables with rows and columns.

Users

This table is used to store data about the user. A new row is created every time a user registers in the app. The details about each column of the Database table is given in table 2.1

Name	Type	Description
id*†	uuid	Primary Key
email	text	Unique Email
created_at	timestamptz	Creation Timestamp
subId	text	Subscription ID
isAdmin	bool	Admin Status
first_name	varchar	First Name
last_name	varchar	Last Name
phone_number	varchar	Phone Number
birth_date	date	Date of Birth
gender	varchar	Gender
location	geography	User Location

*Primary Key, †Foreign Key

Table 2.1: User table

Tickets

This table is used to store the data about the tickets bought by the users. The data from this table is retrieved by the app to show the user the tickets they have purchased in the wallet screen. The details about the fields is given in table 2.2

Name	Type	Description
id*	uuid	Primary Key
user_id†	uuid	Foreign Key (users.id)
trip_id†	uuid	Foreign Key (trips.id)

*Primary Key, †Foreign Key

Table 2.2: Tickets table

Trips

This table is used to store the details about the tickets. The details about the routes the trains travel, and their timings. The details is provided in table 2.3

Name	Type	Description
id*	uuid	Primary Key
train_id	uuid	Train Identifier
from	text	Departure Location
to	text	Arrival Location
departure	timestamptz	Departure Time
arrival	timestamptz	Arrival Time
platform	int8	Platform Number

*Primary Key

Table 2.3: Trips Table

User Locations

The user's location needs to be updated regularly to keep track of their distance from the station. The data from this table is also used to post ads to the user based on how close the user is to the location offering the benefit.The details of the fields are given in table 2.4

Name	Type	Description
user_id*†	uuid	Foreign Key (users.id)
location	geography	User Location
updated_at	timestamp	Last Updated

*Primary Key, †Foreign Key

Table 2.4: User locations table

Stations

This table has a list of the stations that are functional and offering service to the users. Refer to table 2.5 for details of the fields.

Name	Type	Description
id*	uuid	Primary Key
name	text	Station Name
location	geography	Station Location

*Primary Key

Table 2.5: Stations

Deals

This table store the information about all the deals that are offered and the location at which they are offered. The ads are fetched from this table when the user is close to their

locations. More details of the fields is given in table 2.6

Name	Type	Description
id*	uuid	Primary Key
shop_name	text	Store Name
description	text	Deal Description
discount_percentage	int4	Discount Amount
location	geography	Store Location

*Primary Key

Table 2.6: Deals Table

2.1.2. External Services

Supabase Authentication

The backend services to authenticate the user to the app is provided by Supabase. It is used to authenticate the user with their email and password. The email and password of the user are stored in a database with a unique ID attached to the user when they complete the registration.

Notifications Implementation

Push notifications are implemented in the app with the help of various services, namely, OneSignal, Firebase Cloud Messaging (FCM), and Supabase webhooks. Upon a webhooks trigger, the backend, i.e. Supabase, uses the OneSignal API to send notifications via FCM, for an Android device, or Apple Push Notifications service (APNs) for an Apple device. The app then receives the notification through the OneSignal SDK.

OneSignal

OneSignal makes it easier to use push notifications for React Native apps. The app integrates the OneSignal SDK, registering the device, and handling notifications. OneSignal's platform manages device tokens and routes notifications via APNs/FCM. Supabase interacts with the OneSignal API to trigger push notifications. The app then receives the push notification.

Deep Linking

Deep Linking is a technique in React-Native that allows you to open a specific screen or content using an URL. This technique has been to used to link the app to WhatsApp,

while the user wishes to share their location with a contact through Whatsapp. It has also been used to link the app to Google Maps, if the user wishes to get the directions to the nearest station to them.

Dependencies

The packages used by application and for the development of the application is listed in Table 2.7

Package	Main Purpose
react-native	Core React Native framework.
react	The core React library.
expo	The Expo SDK.
supabase-js	Supabase database interaction.
react-native-onesignal	OneSignal SDK for push notifications.
typescript	TypeScript.
react-navigation	Core navigation functionality.
onesignal-expo-plugin	OneSignal plugin for Expo.
expo-location	Location services.
expo-contacts	Contacts access.
expo-sms	SMS functionality.
expo-haptics	Haptic feedback.
expo-linking	Deep linking.
expo-router	File-system based routing.
moment	Date/time manipulation.
jest	Testing framework.
jest-expo	Jest integration with Expo.
react-test-renderer	React testing utils.

Table 2.7: The packages used by application and for the development of the application.

2.1.3. Sequence Diagrams

Sequence diagram is used for visualization of the behaviors and interactions of the components and methods of within the application. A detailed sequence diagram can be very complex and defeats the purpose of using them for easier visualization, so a condensed sequence diagram with the most relevant operations of the application can be found in the following subsections.

Registration/Login

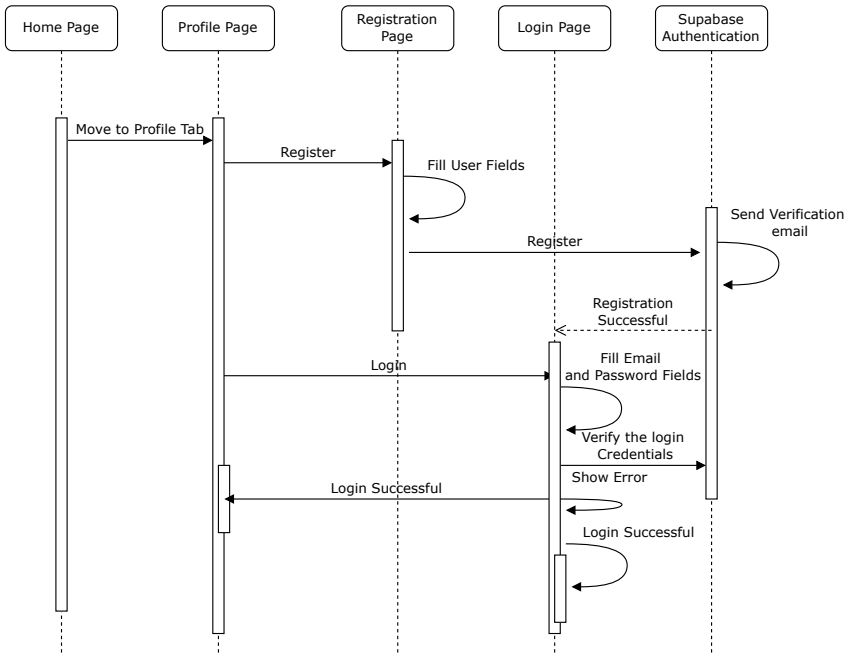

Figure 2.2: Sequence Diagram: Registration/Login.

Buy Integrated/Normal Ticket

Wallet

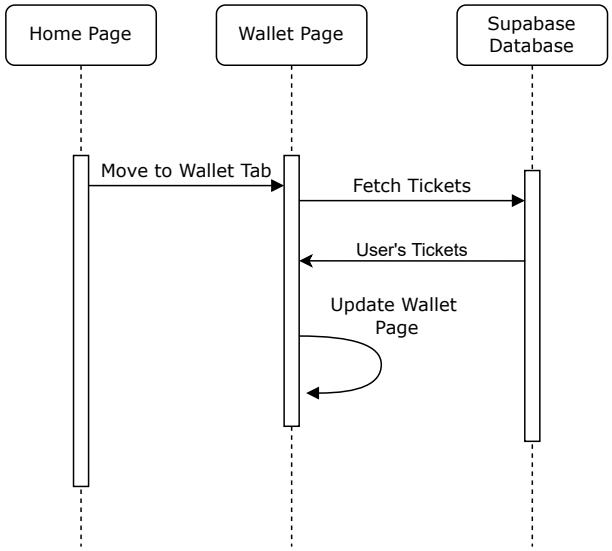

Figure 2.3: Sequence Diagram: Wallet.

S

Push Notifications

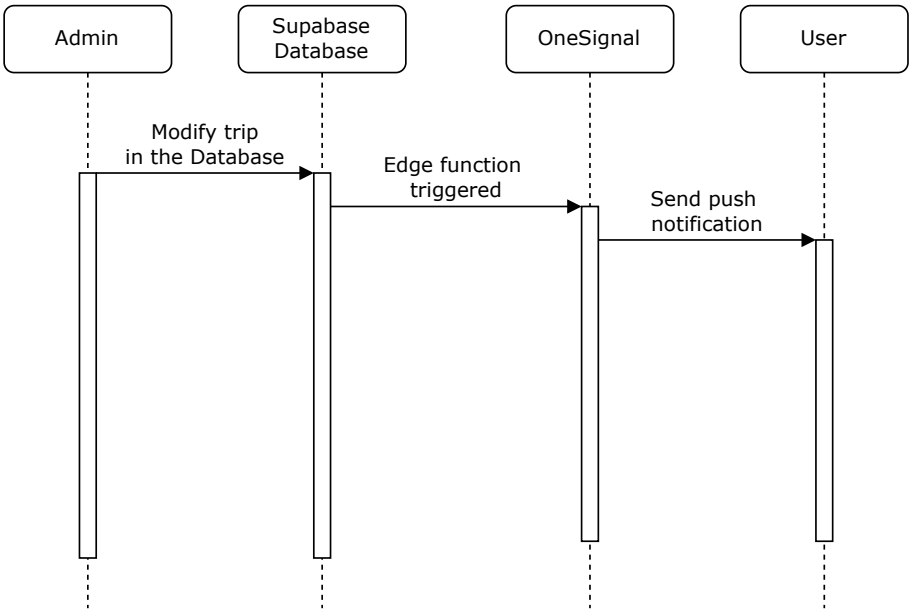

Figure 2.4: Sequence Diagram: Push notification.

Location Based Operations

Directions to Nearest Station

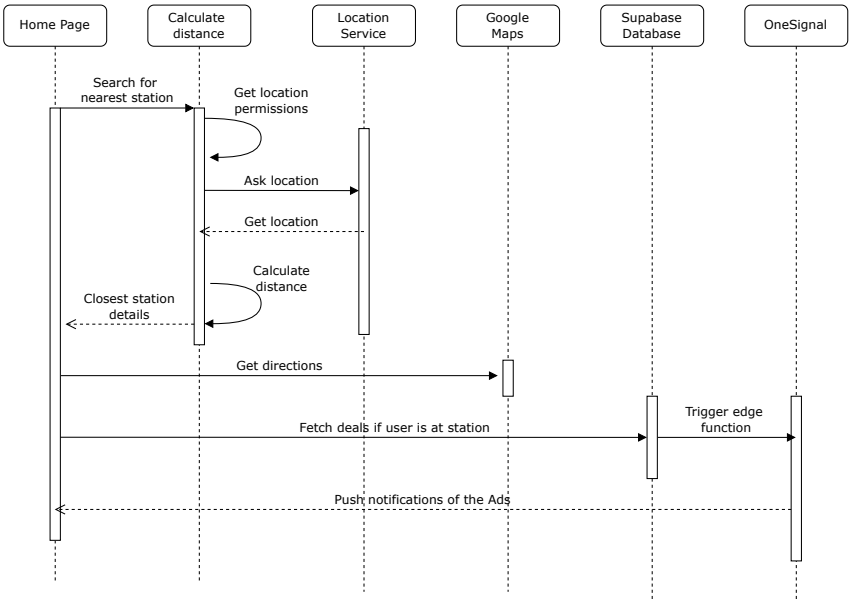

Figure 2.5: Sequence Diagram: Directions to nearest station.

Emergency Contact

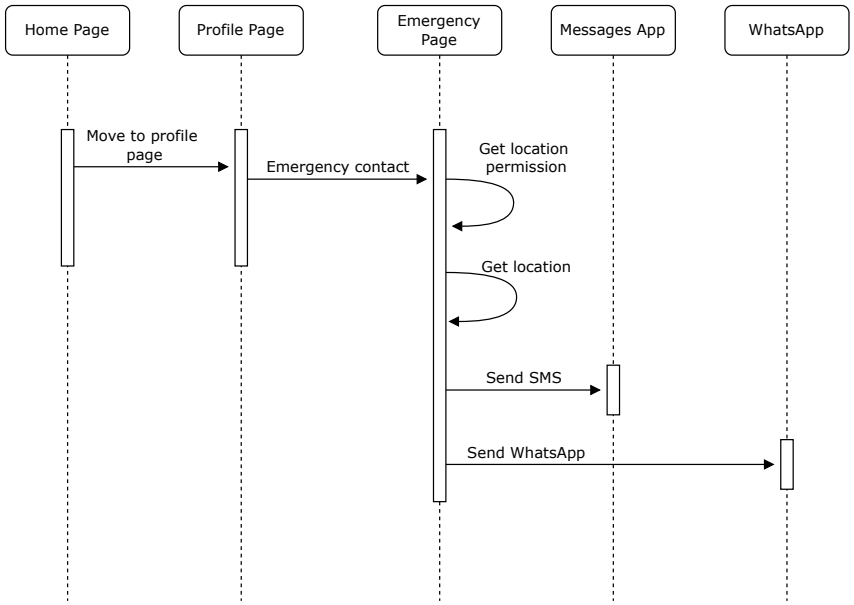

Figure 2.6: Sequence Diagram: Send location details to emergency contact.

Location-Based Notifications

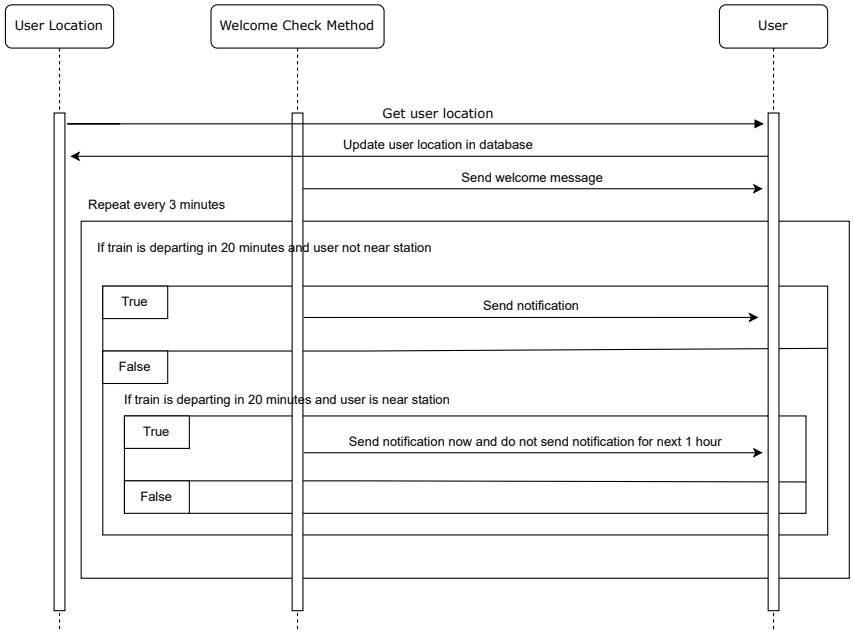

Figure 2.7: Sequence Diagram: Location-based notification system.

3 | User Interface Design

This section describes the User Interface of the Trenord Improvements app, providing an overview of the various screens that make up the app. The design of the app is similar to the official Trenord app.

3.1. User Interfaces

UI1. Home Screen and Bottom Tabs

The Home screen consists of five tabs placed near the bottom of the screen for easier access. This bottom tab provides the user the access to Home, Purchase, Wallet, Train Trips, and Profile screens. Some of these screens are accessible only after the user has logged into the app. The screen that are inaccessible without login are Wallet, Train Trips and Profile Screens. The Home , Purchase, and Train Trips screen is shown in Figure 3.1.

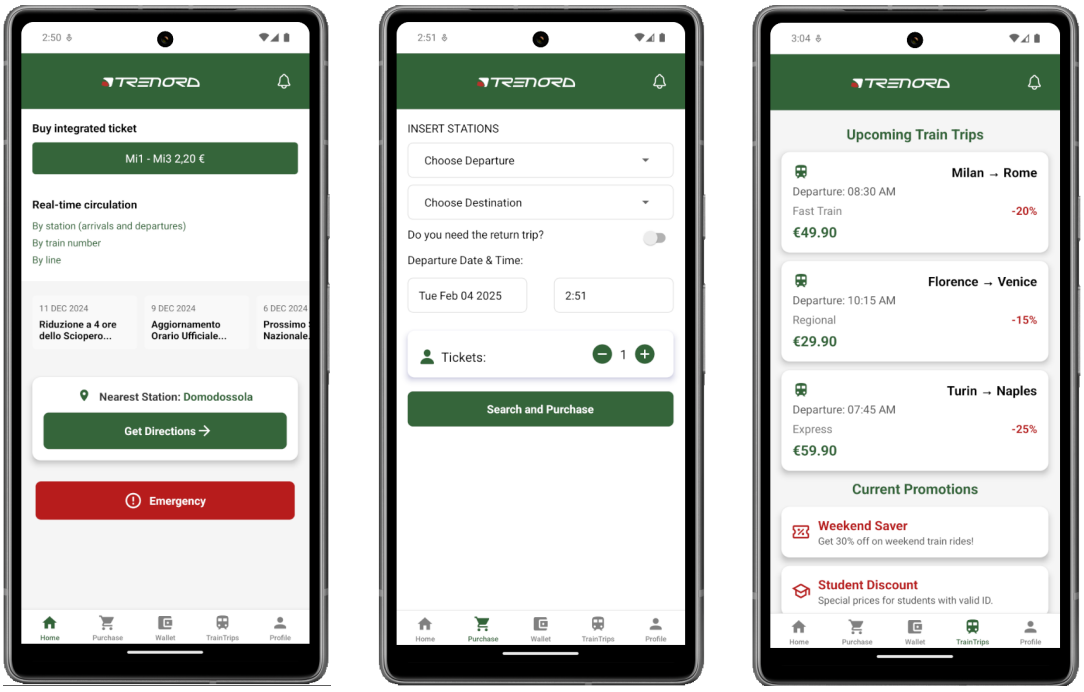

Figure 3.1: Home Screen, Purchase Screen and Train Trips Screens of the app

The Wallet and Profile screens prompts the user to Login, if the user hasn't already logged in. After user login, we can see the tickets purchased by the user in the wallet screen and the user details along with ticket history and emergency contact screen on the Profile Screen. The Wallet screen before and after user login is shown in Figure 3.2, respectively. The Profile screen before and after user login is shown in Figure 3.3, respectively.

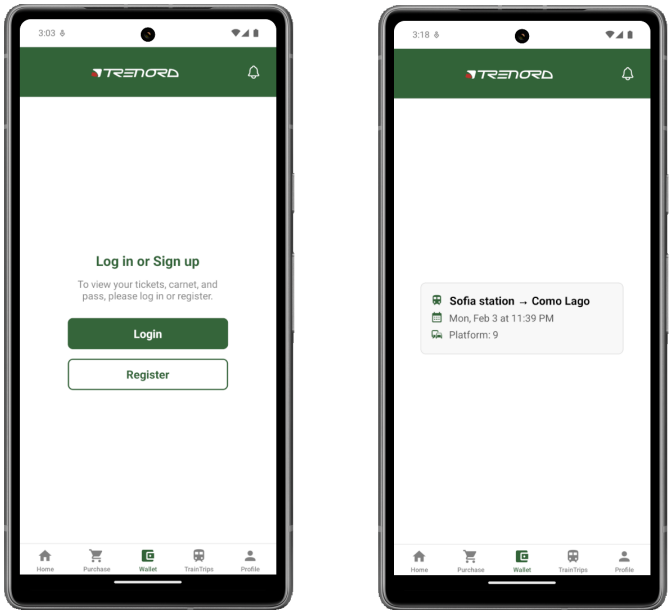

Figure 3.2: Wallet screen before and after user login, respectively.

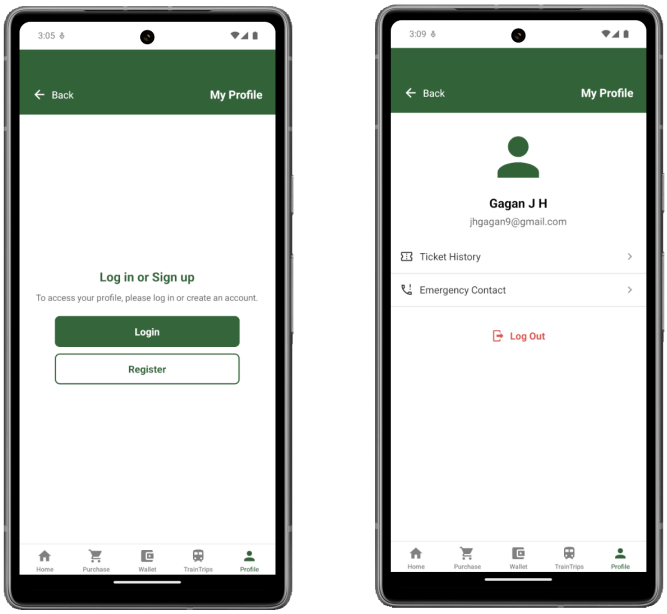

Figure 3.3: Profile Screens before and after user login, respectively.

UI2. Purchasing Integrated Ticket

The user can purchase the integrated ticket for travel within different zones of Milan from the Home Screen. In the Home Screen the user can find the button to initiate the integrated ticket purchase but pressing the 'Mi1-Mi3 2,20' Button. They can also find the information on the nearest station and a shortcut that has a deep link with Google Maps with the directions to the nearest station. There is also the Emergency button that the user can press in case of Danger or Emergency to share their location with the necessary emergency contacts and/or appropriate authorities.

Upon pressing the 'Mi1-Mi3 2,20' Button, the user is taken to the STIBM ticket purchase screen. The User can then select the regions they wish to travel in and the number of tickets in the by interacting with the slider, the '+', and '-' buttons, respectively. Pressing on the 'Mi1-Mi3 2,20' Button then takes the user to the Final Summary Page with all the important details of their selections. Which includes, Ticket Category, Tarrif Zones selected, Validity of individual ticket, Number of passengers selected, Total Price to be paid. The user can confirm the order by pressing the 'PURCHASE' button. The screens during the above mentioned stages is shown in Figure 3.4

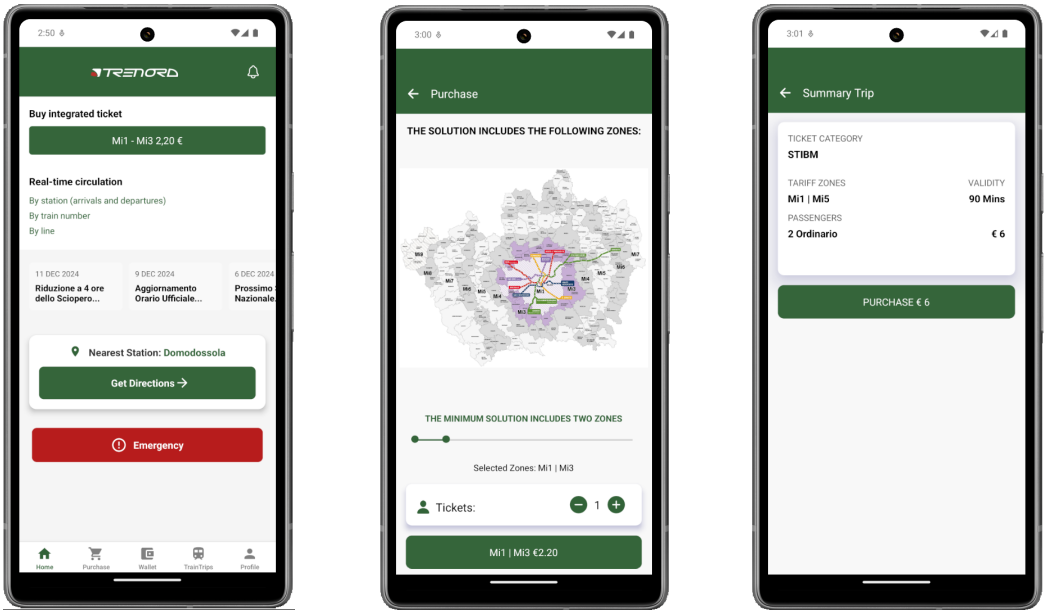

Figure 3.4: Tickets purchase to travel to destinations within Milan's different zones.

UI3. Station Based Ticket Purchase

In Purchase tab, the user is presented with the following options,

- Departure Station

- Destination Station
- Return trip ticket requirement
- Departure Date and Time
- Return Date and Time if return ticket is selected
- Number of tickets being purchased

Once the user has selected, chosen, and filled the above options they can move to the next stage by pressing 'Search and Purchase' button. The User is then presented with the Summary Trip screen containing the information about the ticket type, Departure station, Destination Station, Validity of the ticket once activated, number of passengers, and the price to be paid. Once the user is certain they can purchase the ticket by pressing 'Purchase' Button. Upon which they are presented with a Modal screen, which then redirects them to the Home Page. The screens presented to the user while selecting different options is shown in Figure 3.6. The screens shown during different stages of the purchase is shown in Figure 3.5

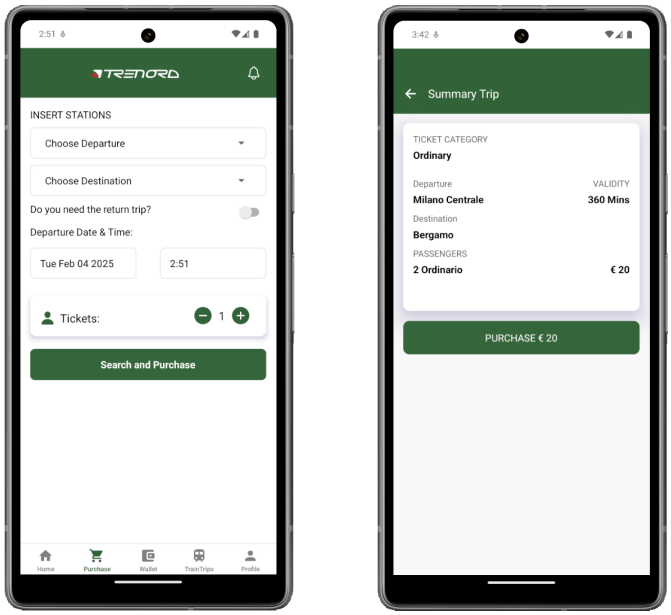

Figure 3.5: Tickets purchase to travel to destinations outside the Milan Zones.

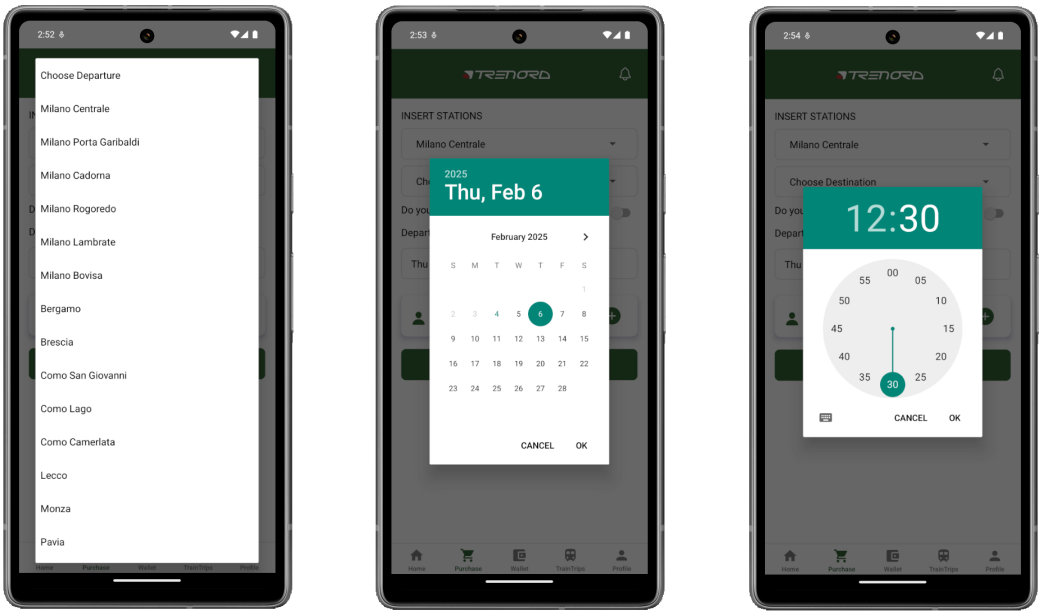

Figure 3.6: Options selections while purchasing the ticket.

UI4. Wallet Screen

The user can't access the Wallet Screen unless they login. Once the user login is completed, they can find the tickets that have been booked in the wallet screen. They booked tickets along with the trip details, such as the departure, destination, date and time of departure and the platform assigned to the train will can be found here. A view of the screen is provided in Figure 3.7.

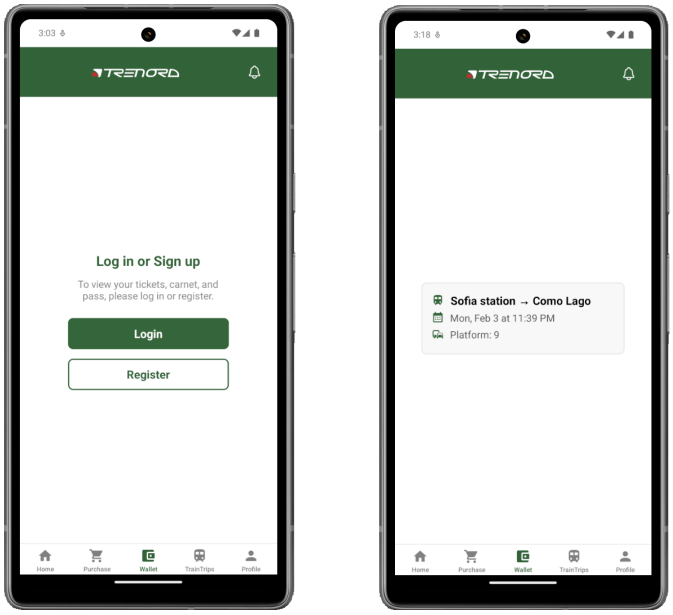

Figure 3.7: Wallet screen and the information of the upcoming trip.

UI5. Train Trips Screen

This Screen provides the user with the information on the latest offers on various routes. This offers and ads on this screens varies based on the location details of the user. It is designed to provide the most relevant offers to the user based on the nearest station to the user. A view of the screen is provided in Figure 3.8

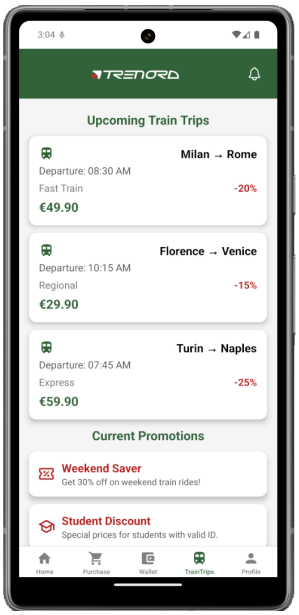

Figure 3.8: Offers and ads on the Train Trips screen.

UI6. Profile Screen

Before user login, they are prompted with a screen that has three buttons, namely, Login, Register, and Emergency button. If the user is new they can choose to register and create a profile but pressing on the 'Register' button. If they are a returning user they can login by pressing on the 'Login' button. In case of emergency the user can press on the 'Emergency' button to share their location. The profile screen at all of the above mentioned stages can be found in Figure 3.9.

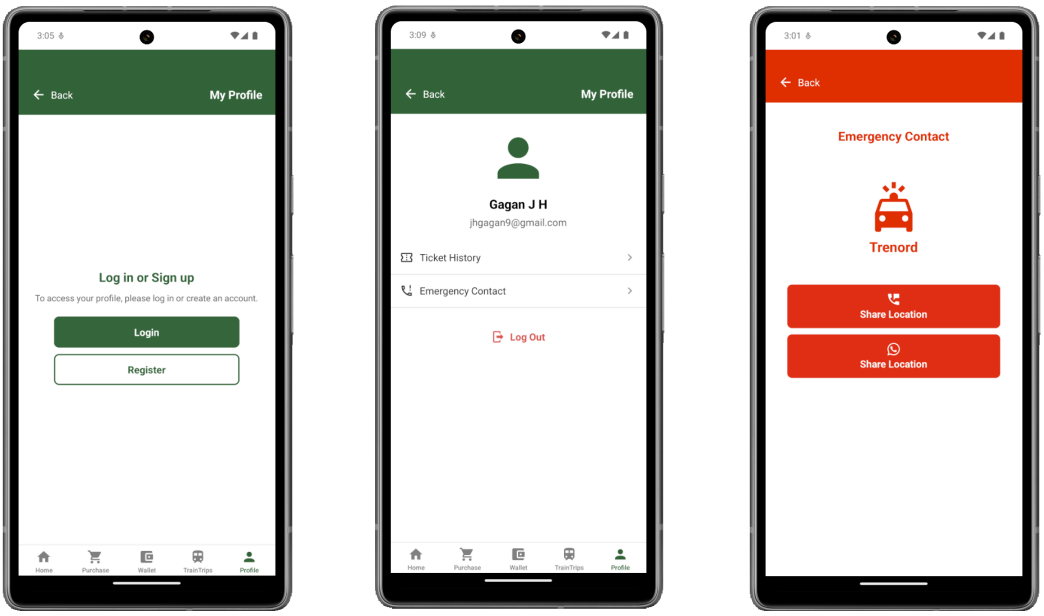

Figure 3.9: Profile screen at different stages.

UI7. Register/Login Screen

A new user can register to the app by filling the details requested. If the user inputs details of an existing account the user will be altered. If they input unique values for email ID a new account is created by creating a row in the users table in the Supabase Database.

A returning user can login to the app from the login Screen. In order to successfully login to the app, the user need to input the email ID and the password. If the pair matches the pair in the Supabase database they user is successfully logged in and redirected to the Home screen. If the user inputs incorrect values they are prompted with an error. The register and login screens are shown in Figure 3.10

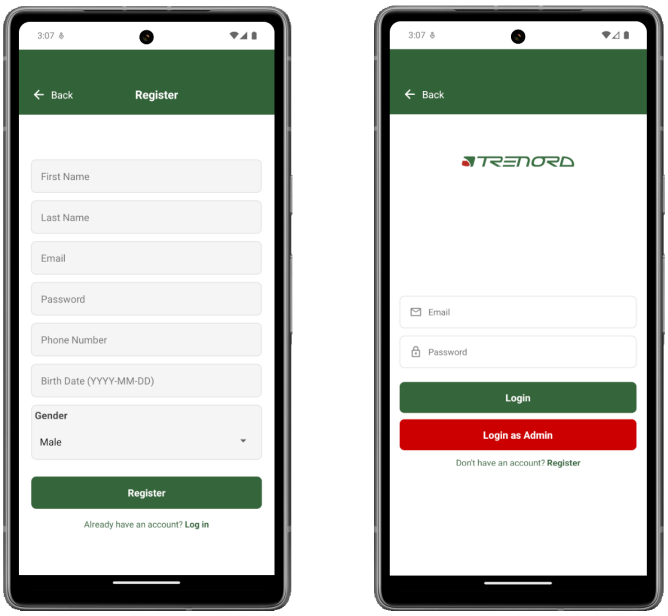

Figure 3.10: Register and Login screens, respectively.

The Login Screen also has the option for admin login. If the user has administrative permissions they will be able to login as admin. This allows the user the ability to modify the trips. that is they can modify the details such as the platform. The view of the screen during these stages has been shown in Figure 3.11.

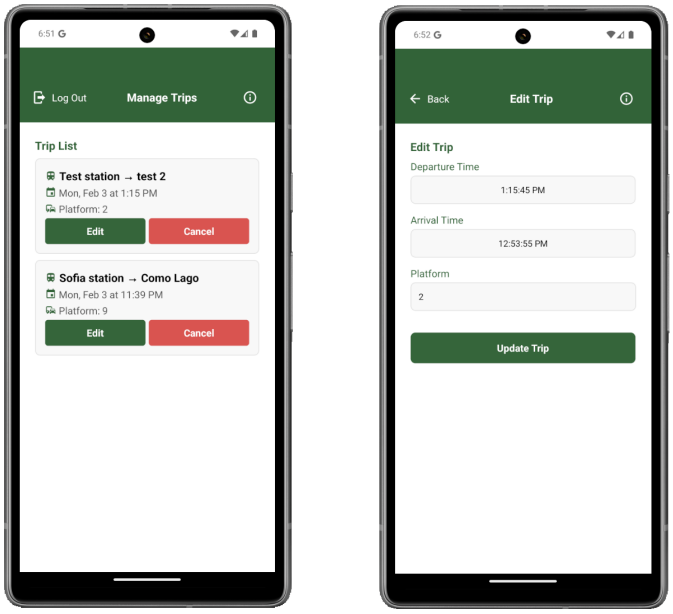

Figure 3.11: Register and Login screens, respectively.

UI8. Emegency Screen

In case of emergency the response of rescue services becomes crucial for avoiding fatalities. To aid rescue services about an emergency being faced by the user the app has a emergency button. On pressing the emergency button the user is able to share their location details. This helps the appropriate authorities to take necessary actions and find and help the user.

With some changes the feature can also be used to help the users share their location with the user's friends and family. The views of the emergency page is shown in Figure 3.12

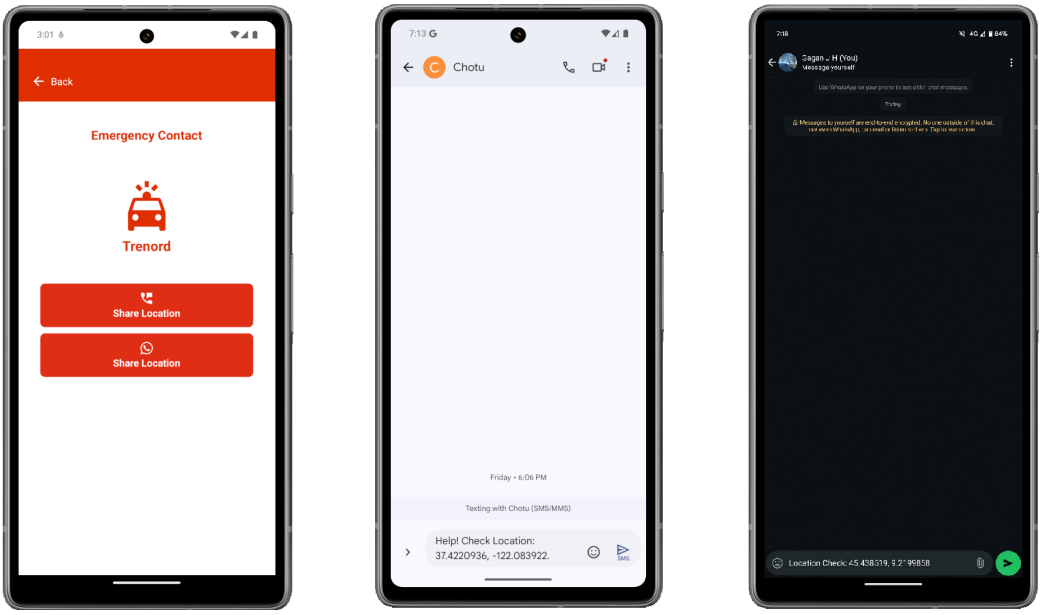

Figure 3.12: Emergency screens when the user attempts to share their location.

UI9. Notifications Page

The "Notifications" page informs users about updates related to their applications or internships. Each notification card includes a brief description of the update (e.g., application status change, internship status update) and the time elapsed since the update occurred. Notifications are listed in chronological order to ensure clarity and ease of access. The notifications screen can be found in Figure 3.13.

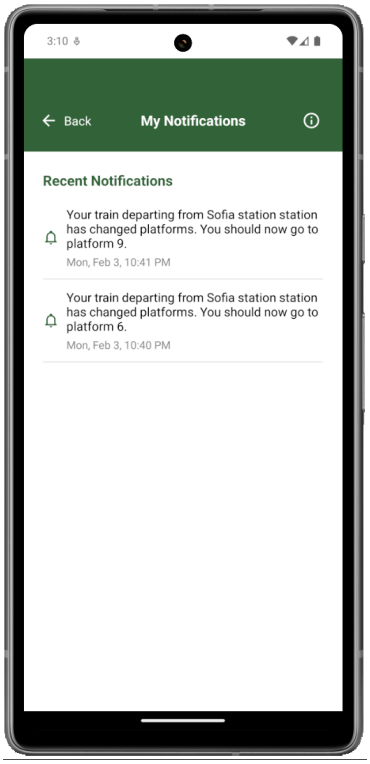

Figure 3.13: Notifications Page.

4 | Implementation, Integration and Test Plan

4.1. Implementation, Integration and Test Plan

4.1.1. Overview and Implementation Plan

The development of the Trenord improvement app is structured into iterative phases. This ensures an efficient overall development of the application. The implementation plan was designed to prioritize addition of new functionalities to the app, than to replicate the existing Trenord app. Subsequent iterations will enhance the overall stability of the application through extensive testing.

The implementation process for each phase will follow these steps:

1. **Requirement Analysis:** Review and refine functional and non-functional requirements for the features to be implemented.
2. **Design and Prototyping:** Create detailed designs, including UI mockups and sequence diagrams, to guide development.
3. **Development:** Implement the features using modular and reusable code practices.
4. **Testing:** Conduct testing to ensure correctness and reliability of the app.

4.1.2. Feature Identification

The table below summarizes the core features to be implemented, along with their respective priorities:

Feature	Priority	Description
Location based notifications	High	Sends real-time push notification based on user's location.
Notification System	High	Sends real-time updates to users regarding activities.
Emergency Contact	High	Notify emergency contacts about the user's location
User Registration	Medium	Allows users (STs and CPs) to create accounts.
Login and Authentication	Medium	Enables secure access to the platform.
Ticket Purchase and Management	Low	Purchase and management of ticket is already existing
Profile Management	Low	Enables users to update their profile.

Table 4.1: Core Features and Priorities

4.1.3. System Testing Strategy

Comprehensive testing ensures that the app operates reliably under various conditions. The following testing methodologies can be employed:

- **User Acceptance Testing:** The platform is to be tested by end-users (students) to gather feedback and validate that the newly added functionality and features are relevant and functions as intended.
- **Unit Testing:** Each component undergos rigorous testing to verify functionality against individual requirements.
- **Integration Testing:** After unit testing, integrated components should be tested to ensure seamless interaction and data consistency.
- **System Testing:** The fully integrated system should be tested for performance, reliability, and adherence to requirements under real-world conditions.

Note: User Acceptance Testing was carried out to verify the newly added features. Few unit tests were performed, but extensive testing was not achieved. We were unable to complete unit testing, integration testing, and System testing at the time of submission of the report.

4.1.4. Exception Handling Testing

Special attention will be given to scenarios where:

- **Incorrect Data:** Test how the system handles invalid inputs, such as incorrect login credentials, incomplete registration forms.

List of Figures

List of Tables