POLITECNICO
MILANO 1863

DESIGN AND IMPLEMENTATION OF MOBILE APPLICATIONS

Project Report

Junjie Mu, Xinmeng Wang,

******, ******

February 4, 2025

1 Introduction

1.1 Application overview

The **Trenord+** App is designed to enhance the user experience of railway commuters in the Lombardy region, featuring the following four core functional modules:

- **Train Tourism**: provides ticket adding, travel reminders, and location-based recommendations for nearby tourist attractions, restaurants, lodging, and shopping.
- **Travel Safety**: integrates safety features such as location sharing, emergency assistance (SOS), and emergency contact settings within the interface.
- **Voice Guidance**: utilizes text-to-speech technology to read out page transitions and important information, enhancing accessibility for users.
- **Personalized Appearance**: allows users to customize the interface theme color based on their preferences and usage habits.

We provide a more convenient, safer, and smarter railway travel experience through a streamlined interface, intelligent features, and a robust security system.

1.2 Features

This application focuses on four core modules: train travel, travel safety, support for vulnerable groups, and a personalized interface, offering the following key features:

1.2.1 Train Tourism

- **Train Reminders**: when a logged-in user opens the app and enters the homepage, if they have added a ticket, the most recent train ticket will be displayed. A red dot will appear on the reminder icon. When the user clicks and enters the reminder page, the departure time of the next train will be shown.
- **Ticket Management**: users can add purchased tickets by clicking the plus icon on the right side of the **Ticket** page. This page displays all upcoming tickets, while the My Tickets section in the **Profile** page allows users to view all added tickets, including used ones.
- **Nearby Recommendations**: on the **Home** page, users can search for a location, use their current location, or select a frequently used location. Under Train Trip, the app recommends nearby tourist attractions, restaurants, accommodations, and shopping centers based on the selected location. Users can tap on a

location card to open the phone's map application for easy navigation.

1.2.2 Travel Safety

- **Location Sharing**:in the **Security Center**, users can share their real-time location by tapping **Location**. They can obtain their current location, copy the location link, or share it via other communication apps.
- **Emergency Assistance**: in the **Security Center**, users can tap the **SOS** button in case of an emergency. If the user is not logged into the app or has not set up an emergency contact, the app will dial the 112 emergency hotline. If an emergency contact is set, the app will compose and send a message stating the emergency situation along with the user's current location.
- **Emergency Contact Setup**: in the **Security Center**, users can set up emergency contacts through the **Contact** option. This feature requires users to log in, and emergency contact information is stored in the database.

1.2.3 Voice Guidance

After enabling this feature on the **Profile** page, the app will automatically play a welcome message describing the current page's functions when switching pages. Different pages have specific voice playback features:

- On the **Home** page, the system announces the nearest upcoming trip.
- On the **Ticket** page, tapping a ticket will trigger audio playback of the ticket details.
- On the **Profile** page, the system prompts users to log in if they have not already done so.

1.2.4 Personalized Appearance

Users can customize the app's theme color and adjust brightness in the **Appearance** section of the Profile page. These settings are locally stored and will be applied automatically when the app is restarted.

1.3 Defination

1. **Trenord**: A railway company in the Lombardy region of Italy, providing intercity and regional train services.
2. **Flutter**: A cross-platform UI framework developed by Google, used in this task to build the Trenord+ mobile application.
3. **Dart**: A C-like programming language used by Flutter for UI logic design in front-end development, supporting hot reload and object-oriented programming.
4. **Firebase**: A cloud-based backend service provided by Google, supporting authentication, databases, storage, cloud functions, and more.
5. **Firestore**: A NoSQL cloud database provided by Firebase, supporting real-time data synchronization and offline storage.
6. **Google Maps API**: A map service API provided by Google, used for user location tracking and other functionalities.
7. **Geolocator**: A Flutter plugin used to obtain the device's geographic location. In this task, Geolocator is mainly used for retrieving GPS coordinates to enable location sharing and emergency assistance features.
8. **Text-to-Speech (TTS)**: A technology that converts text into speech. In Flutter, flutter_tts is used to implement TTS functionality for text narration.

2 User Interface Design

2.1 Overview

This application inherits the original UI style of the Trenord app while adding new features such as a Safety Center and personalized interface customization to enhance user experience.

2.2 User Interface

2.2.1 Home

Users can select a location on the Home page and receive recommendations related to the chosen location.

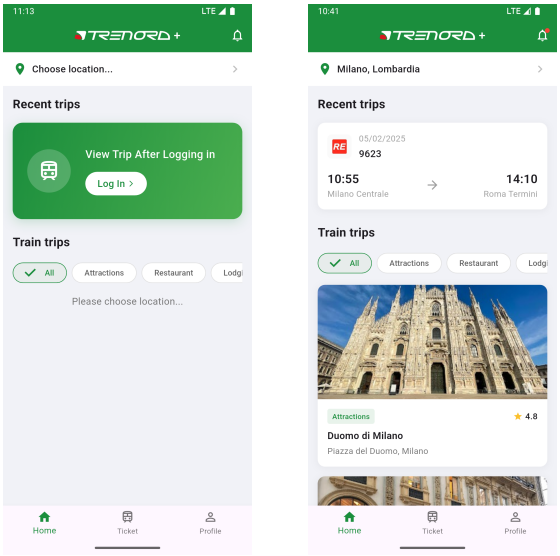

Figure 1: Home

2.2.2 Choose Location

At the top of the Home page, clicking "choose Location" navigates to the Location Selection page. Users can choose to use their current location or manually search and add another location.

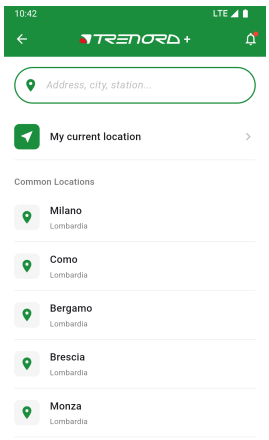

Figure 2: Choose location

2.2.3 Notification

Click the icon in the top right corner of the Profile page to access and view system notifications.

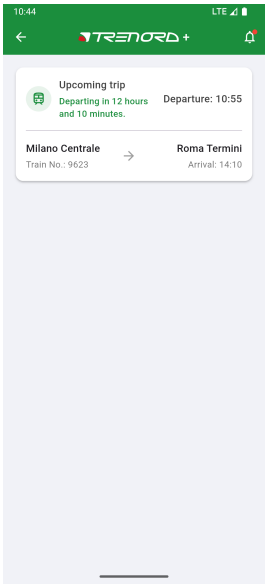

Figure 3: Notification

2.2.4 Ticket

Logged-in users can add tickets on the Ticket page and view previously added tickets.

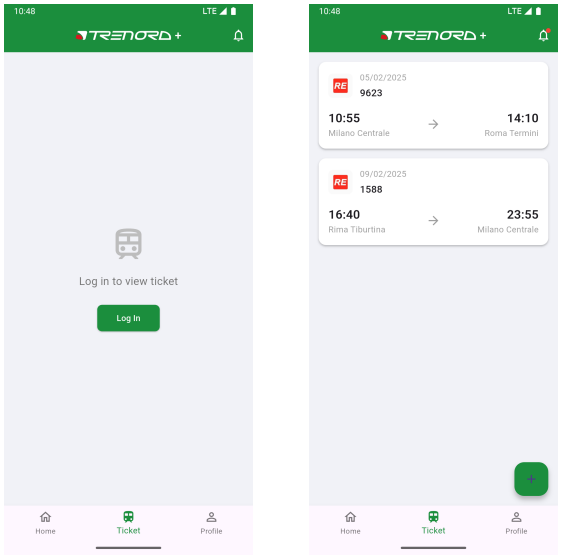

Figure 4: Ticket

2.2.5 Add ticket

After logging in, users can add tickets by clicking the green "+" icon in the bottom right corner of the Ticket page.

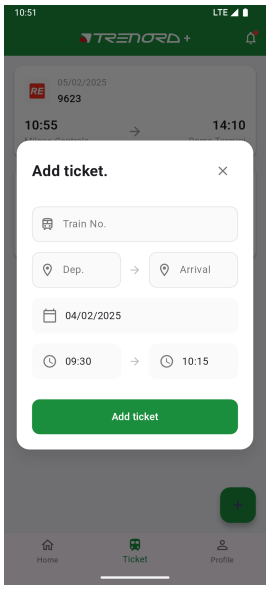

Figure 5: Add ticket

2.2.6 User Profile

The user registration, login, and logout page also serves as the entry point for accessing the Safety Center, using voice navigation, and customizing personalized services.

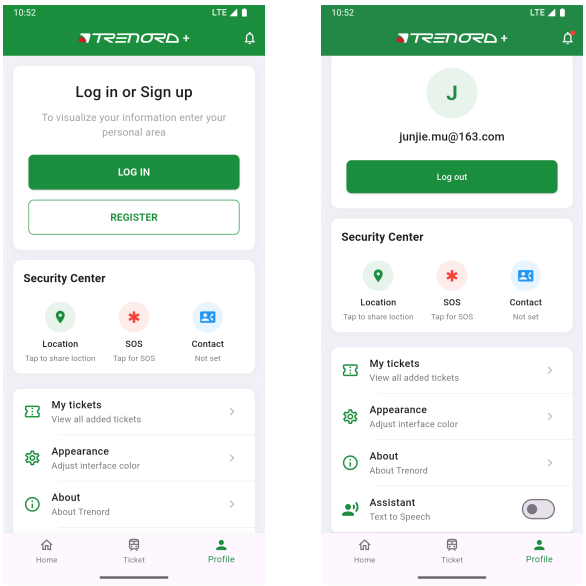

Figure 6: Profile

2.2.7 Login and Register

Existing users can log in by clicking "Login" on the Profile page. Users who have not registered can click "Register" to create an account.

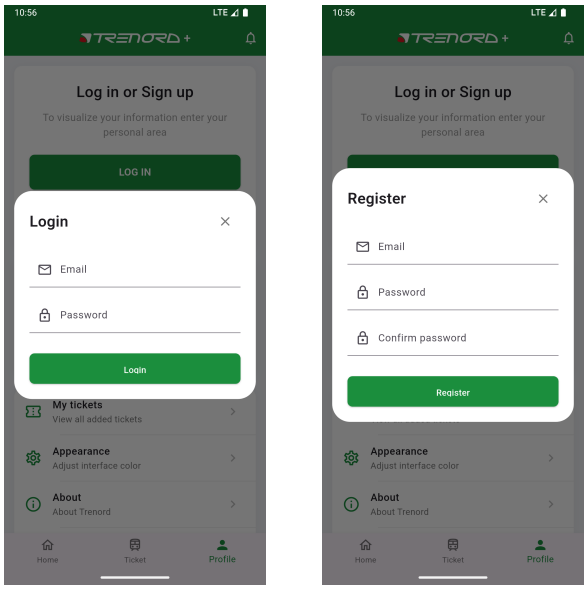

Figure 7: Login and Register

2.2.8 Appearance

Users can customize the app's color settings in the "Appearance" section on the Profile page.

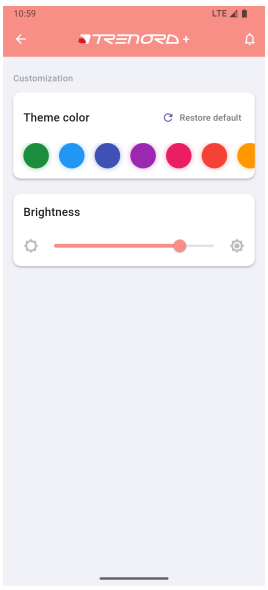

Figure 8: Appearance

2.2.9 Security Center

Logged-in users can access the **Security Center** to share their current location, request emergency assistance by tapping the **SOS** button, and set up emergency contacts.

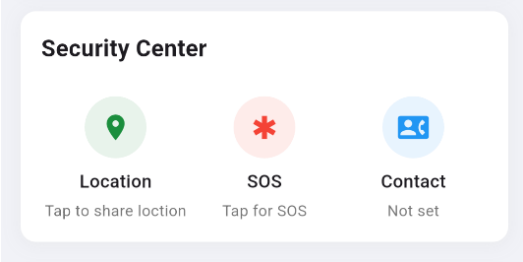

Figure 9: Security Center

2.2.10 User Function Area

Users can access My Tickets, customize the appearance, view information About the app, and enable the Voice Navigation Assistance button.

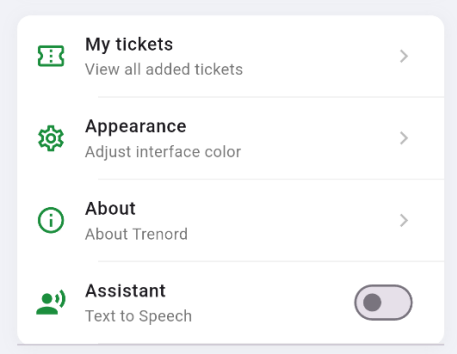

Figure 10: User Function Area

2.2.11 My tickets

Users can view all added tickets here, including those that have already been used.

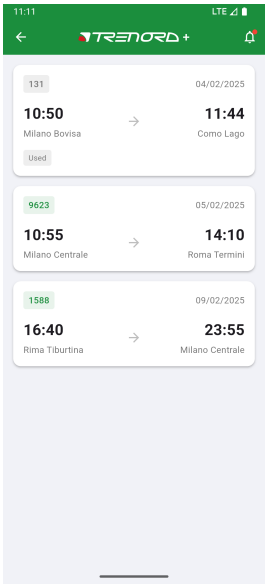

Figure 11: My tickets

3 Use Case

A Use Case is a way to describe how a system interacts with users or other systems, typically used in the software requirements analysis and design phase. By defining the interaction process between the user (Actor) and the system, it clarifies how the system should meet user needs. Use Cases are primarily used to describe functional requirements, helping developers, testers, and business stakeholders understand the expected behavior of the system.

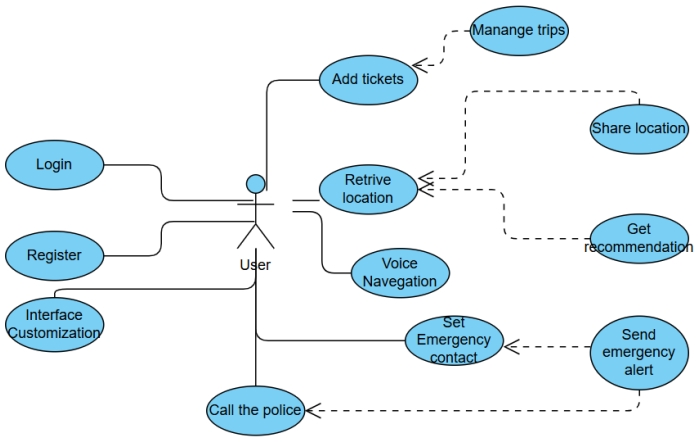

Figure 12: Use Case Diagram

4 Architectures&Functionalities

4.1 Flutter Overall Architecture

The Trenord+ App is structured into the following layers:

- **UI Layer**: Responsible for the user interface and interactions, primarily composed of Flutter Widgets.
- **Logic Layer(ViewModel)**: Utilizes GetX Controllers to handle business logic, manage data transformations, and communicate with the backend.
- **Model&Service**: Manages data storage and network interactions, supporting Firebase Firestore and REST API.

4.2 Firebase Interactions

Trenord+ integrates Firebase for backend services, supporting user registration, authentication, real-time database storage, and push notifications.

4.2.1 Firebase Authentication

- **Implementation**: User registration and login are implemented via email and phone number authentication. Data Flow:
- **Data Flow**: When the user opens the app, if the user is already logged in, the app automatically retrieves user information from Firebase.If not logged in, the user is redirected to the login page.Upon successful login, the app loads the user's ticket information.

4.2.2 Firestore Database

- **Implementation**: stores user ticket information and allows modifications based on actual conditions. Data Flow:
- **Data Flow**: after logging in, the app retrieves user data from Firestore. When a user adding a new ticker it is added to the tickets collection. The user can modify the trip details as needed based on actual conditions.

4.3 Function Implementation

4.3.1 Location Sharing Feature Implementation

The location-sharing feature is implemented using the Geolocator library to obtain the user's geographic location and the Share Plus library to share location information.
Implementation Process:

1. Permission HandlingCalls Geolocator.checkPermission() to check if the user has granted location permissions.If permission is denied, requests authorization using Geolocator.requestPermission().If permanently denied, prompts the user to manually enable permissions in system settings.
2. Retrieving GPS Location:Uses Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high) to obtain high-accuracy GPS coordinates.
3. Generates a clickable Google Maps URL: https:www.google.commapssearch?api= 1&query= latitude, longitude, allowing users to view the location directly in Google Maps.
4. Sharing Location:Calls Share.share() to allow the user to share the location link with others.

4.3.2 Emergency Assistance Feature Implementation

The implementation is divided into two methods:

- If the user has set an emergency contact, an SMS notification is sent to the emergency contact, including the user's current location.
- If the user has no emergency contact, the app directly dials 112.

Implementation Process:

1. **User Authentication Check**: the SOS feature is only available for logged-in users, as emergency contact information is stored in Firebase Firestore and must be retrieved using the user's ID.
2. **Location Retrieval**: The app obtains the latest GPS coordinates using Geolocator.getCurrentPosition().

3. **Fetching Emergency Contact**: retrieves the user's emergency contact from Firebase Firestore via FirebaseFirestore.instance.collection('users').doc(user.uid).get().

4. **Sending Emergency Notification**: if the user has an emergency contact, the app opens the SMS app using sms:phone?body=message, pre-filling the emergency message with the user's location for manual sending.

4.3.3 Text-to-Speech Implementation

Text-to-Speech (TTS) Feature is implemented using the flutter_tts plugin, a text-to-speech library in the Flutter framework that converts in-app text into speech playback.
Implementation process:

1. **Initialization**: in the TtsController controller, create a instance and initialize it in the onInit() method. Set language using flutterTts.setLanguage("en-US").Adjust speech rate with flutterTts.setSpeechRate(0.5), and configure volume, pitch, and other parameters.

2. **User Settings**: users can manually enable/disable TTS.On app startup, it checks the isTtsEnabled state to ensure speech playback occurs only when enabled.isTtsEnabled is globally managed using GetX and can be adjusted in the settings page.

3. **Triggering TTS**: at key interaction points, such as navigating to a page or clicking a button, the speak(text) method is called.This method verifies whether TTS is enabled and then calls flutterTts.speak(text) to read the provided text aloud.

4. **Playback Control&State Management**: the function setStartHandler() sets isSpeaking = true when playback starts.setCompletionHandler() sets isSpeaking = false after playback finishes.If an error occurs (e.g., the device does not support TTS), setErrorHandler() handles the error and notifies the user.

5 Test Plan

Testing is a crucial step in application development to ensure that our application functions as expected, remains stable across different devices, and provides a seamless user experience. We use manual testing to verify functionality, performance, and user interactions. We plan to use the following tests:

5.1 Static Test

Since the application is developed using Flutter and Dart, static type checking plays a crucial role in the development process. Dart's built-in type system helps detect type mismatches, missing properties, and incorrect function parameters during compilation. This process reduces runtime errors and enhances code reliability before execution.

5.2 Manual Test

Manual testing is intended to ensure that the core functionalities of the application work as expected under real user conditions. We plan to manually execute multiple test cases on various devices to verify usability, responsiveness, and stability.