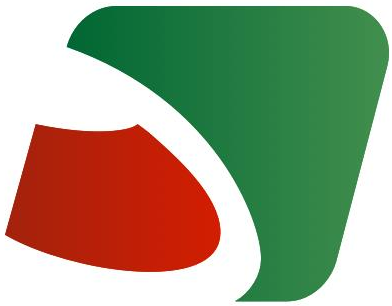
Trenord – Travel APP

Design Document



AUTHORS
Xiao Tan
Jiayi Su

INSTRUCTOR
Luciano Baresi

04/02/2024

Design and Implementation of Mobile Application (DIMA) course

Table of content

1. Introduction

1.1. Purpose

Trenord - Travel aims to develop a travel planning application designed specifically for railway travelers, offering features such as itinerary planning, attraction recommendations, and train schedule inquiries. Users can utilize Google Maps to visually explore destinations and nearby attractions, while integrating Trenord railway services to find suitable train schedules, enabling them to efficiently create personalized travel plans. The application allows users to manage their travel itinerary, browse detailed destination information, and access optimized railway travel solutions, enhancing the overall travel experience and making railway journeys more convenient and intelligent.

1.2. Requirements

This section lists the Functional Requirements and Non-Functional Requirements for the Trenord - Travel application to ensure that the application meets the user's needs and provides a good experience.

1.2.1. Functional Requirements

The application needs to have the following core functions:
1. User account management
o Users can register and login to the application.
o Users can log out and manage personal information (e.g. username, email, etc.).
2. Trip Planning
o Users can create a personalized travel plan by selecting the origin and destination.
o The application should recommend attractions along the way based on user input.
3. Map navigation and localization
o Integrate Google Maps to display the location of destinations and recommended attractions.
o Users can view the best route between different locations.
o Support real-time positioning to display the user's current location for more accurate trip planning.
4. Train Schedule Inquiry
o Combined with Trenord railroad data, users can query train schedules, timetables, and basic ticket information. o The application should provide the ability to search for trains by time and location.

o The application should provide the function of filtering train trips by time and location. 5.
5. Attraction Search and Recommendations
o Users can search for specific attractions and get detailed information (name, description, opening hours, etc.).
o Provide intelligent recommendation function to recommend relevant attractions based on user's itinerary, location and preference.
6. Trip Management
o Users can view, edit or delete created trip plans.
o The trip list should display information such as destination, mode of transportation, estimated time, etc.

1.2.2. Non-Functional Requirements

The application needs to meet the following requirements in terms of performance, availability and security:
1. Cross-platform support
o Developed using Flutter to ensure that the application works well on iOS and Android devices.
2. Responsive Layout
o Adopt responsive design to adapt to different screen sizes (cell phones, tablets).
o UI components can be adjusted according to device resolution and orientation (landscape/vertical).
3. Good User Experience
o Adopt intuitive UI design, so that users can quickly get started.
o Key operations such as itinerary creation, train search, attraction search, etc. should maintain a smooth interactive experience.
4. Performance Optimization
o The application needs to ensure fast response and avoid long loading delays.
o Data access should be asynchronous to improve user experience.
5. Data Storage and Management
o User trip data should be stored in a cloud-based database (Firestore Database) to support multi-device access.
o) to support multi-device access.
o Local caching is used to speed up the loading of frequently used data.
6. Security
o Adopt Firebase Authentication for user authentication to ensure account security.
o User data should be stored with appropriate encryption to avoid sensitive information leakage

1.3. Features Implemented

Trenord - Travel, a rail travel planning app, has implemented the following core features to enhance the user's travel experience.

1.3.1. User account management

- User Registration and Login: Supports new user registration, email login and Firebase Authentication for authentication.
- Personal Information Management: Users can view and update account information (e.g. username, email, avatar).
- Logging out: Users can log out of the application at any time to ensure account security.

1.3.2. Trip planning

- Create personalized travel plan: users can input departure and destination to make a detailed travel plan.
- Suggested Attractions: Based on user input, the app will suggest popular attractions along the way to help optimize the trip.
- Trip Management: Support users to view, edit and delete existing trip plans.

1.3.3. Map Navigation and Positioning

- Google Map Integration: Users can view destinations, recommended attractions and trip planning on the map.
- Real-time Positioning: The application supports GPS positioning to help users get the current location and plan suitable routes.

1.3.4. Train Schedule Inquiry

- Trenord Railway Data Integration: Users can query the frequency, timetable and related information of Trenord Railway.
- Conditional filtering: Support users to filter train frequency according to departure time and destination.
- Detailed Train Information: Display important information such as time, station and running status of each train.

1.3.5 Attractions Search and Recommendation

- Attractions Search: Users can input keywords to search for specific attractions and view detailed information (e.g. name, description, opening hours, etc.).
- Intelligent Recommendation: Recommend relevant attractions based on user's location, trip planning and preferences to optimize the travel experience.

1.3.6 Responsive Design and Multi-End Adaptation

- Adaptation to different screen sizes: Adopt responsive layout to ensure good display on different devices (cell phones, tablets).
- Horizontal/vertical screen adaptation: Automatically adjust the UI layout according to the orientation of the user's device to improve usability.

1.3.7 Data Storage and Management

- Firestore Database: All user data, itinerary information, favorite attractions, etc. are stored in Firestore, realizing real-time synchronization and cloud management.
- Local Cache Optimization: Use SharedPreferences/SQLite to cache part of the data to improve loading speed and reduce dependence on the network.

1.3.8 Security and Authentication

- Firebase Authentication: Ensure user data security, support email login.
- Database Access Control: Firestore uses access rules to ensure that users can only access their authorized data.

2. Application Architecture

2.1. Overview

Trenord - Travel is a cross-platform mobile application developed based on Flutter, supporting iOS and Android, aiming to provide rail travelers with trip planning, attraction recommendations, train schedule inquiries and other functions. The app uses Google Maps API for map navigation, and combines with Trenord railroad API to provide real-time train information. All user data and trip information are stored in Firestore Database to ensure real-time synchronization and cross-device access.

Architecture Design
The application adopts MVC (Model-View-Controller) architecture to ensure modularity, maintainability and scalability of the code:
- Model (data layer): manages all application data, including user information, trip planning, train schedules and attraction data. The data is mainly stored in Firestore Database and optimized with local cache (SharedPreferences/SQLite).
- View: Use Flutter widgets to build a responsive UI, adapting to different screen sizes to ensure a good user experience.
- Controller: Responsible for handling user input, interacting with API, and coordinating data transfer between Model and View.

Technology stack

Component	Technology
Frontend Framework	Flutter (Dart)
Backend Service	Firebase Firestore (Real-time Database)
Authentication	Firebase Authentication
Map Services	Google Maps API
Train Data API	Trenord Railway API
Data Storage	Firestore Database + Local Storage
State Management	GetX
UI Design	Material Design
Location Services	GPS + Location Plugin
HTTP Requests	http Plugin
Image Processing	image_picker Plugin
Environment Variables	flutter_dotenv Plugin
Push Notifications	Firebase Cloud Messaging
Route Planning	flutter_polyline_points Plugin

Data Flow and Interaction

1. User Login and Authentication

o Firebase Authentication is used for user authentication, supporting email login and Google login. o After successful login, the user data is pulled from the Firestore Database to maintain the session state.

o After successfully logging in, pull user data from Firestore Database and keep the session status.

2. Trip Planning and Storage

o User creates a trip plan, data is stored in Firestore Database and synchronized to local cache. o Supports offline access, ensuring that there is no need to access to the Firestore Database.

o Support for offline access ensures that saved trips can still be viewed when there is no internet access.

3. Map Navigation and Location

o Integrate with Google Maps API to display destinations, recommended attractions and best routes. o Combine with GPS positioning to dynamically update user's location.

o Combine with GPS positioning to dynamically update the user's location. 4.

4. Train Schedule Inquiry

o Get real-time train schedule and timetable information through Trenord Railway API. o Users can check the train schedule according to the departure time, destination and time of travel.
o Users can filter train information according to departure time, destination, etc. 5
5. Attraction Search and Recommendation
o Users can search for specific attractions and get detailed information (name, description, opening hours, etc.). o Combine the user's location and trip data.
o Combine user location and trip data to provide intelligent recommendations.
6. Responsive Layout
o Adopt Flutter responsive design to adapt to different screen sizes (cell phones, tablets). o The interface layout can be adapted to the device resolution.
o The interface layout can be adjusted according to the device resolution and orientation (horizontal/vertical screen).

2.2. Data Model

The structured storage of data displayed in the application pages follows the MVC pattern, where the Model layer is responsible for defining the data objects and interacting with the Firestore Database to ensure data organization and scalability. Based on the requirements analysis in Requirements and the page design, the following core data models were defined:
- UserModel and PositionModel: store detailed data of user registration information and current position.
- TripModel: defines the data structure of the trip plan, including origin, destination, train schedule and other information.
- TrainModel: stores detailed data of train schedule, such as train number, departure and arrival time.
- AttractionModel: defines attraction information, including name, location, opening hours, etc.
The data of the application comes from multiple sources and is obtained through the Firestore API. The data is parsed and converted into Dart class objects to fit the business logic of the application.

2.3. Data Management

2.3.1. Firestore

The application uses Firestore Database as the main data storage and NoSQL document structure to organize the data. Trip data is stored in the form of Collection and Document, which mainly contains user information, plan name, date and itinerary.

Firestore Database
```
| ——— trips (Collection)
|    ├——— {tripId} (Document)
|       ├——— userId: (String)
|       ├——— planName: (String)
|       ├——— date: (String)
|       ├——— routes: (Array)
|          ├——— 0: (Map)
|             ├——— waypoint: (String)
|             ├——— expanded: (Boolean)
|             ├——— transports: (Array)
|          ├——— 1: (Map)
|             ├——— waypoint: (String)
|             ├——— expanded: (Boolean)
|             ├——— transports: (Array)
|          ├——— ... (More waypoints)
```

Description of data fields

Field	Type	Description
userId	String	Associated user ID
planName	String	Name of the travel plan
date	String	Travel date
routes	Array	Collection of travel routes
waypoint	String	Travel waypoint location
expanded	Boolean	Indicates whether the waypoint details are expanded
transports	Array	List of transport methods for the waypoint

2.3.2. Local Storage

This application uses SQLite for local storage, which is mainly used to cache recently accessed trips and query records to reduce frequent requests to the Firestore Database, improve data loading speed, and provide basic accessibility in a network-less environment. Since most of the data relies on cloud storage, the use of local storage is less extensive and limited to improving user experience and optimizing performance

2.3.3. API Integration

The app integrates several APIs to provide key functionality such as map navigation, train schedule lookup and location services.
- Google Maps API: for map display, location search, route planning and navigation functions.
- Trenord Railway API: provides train schedule, timetable and station information query to help users get real-time railroad data.
- Geocoding API: support address and latitude/longitude conversion, used for location resolution and navigation.
- Location API: Get user's current GPS location to provide location-based trip suggestions.
These APIs interact with each other via HTTP requests to ensure that the application can obtain the necessary data in real time to optimize the user experience.

2.4. External Services

This application relies on several external services to provide functions such as user authentication, cloud storage and file management to ensure data security and real-time availability.
- Firebase Authentication: Used for user authentication, supports email login and Google login.
- Firestore Database: stores user data, trip plans and attraction information and supports real-time data synchronization.
- Firebase Storage: Used to store images or other attachments uploaded by users, such as avatar-related photos or text.
These external services are integrated through the Firebase SDK, enabling the application to efficiently and securely manage user data and provide a reliable cloud-based storage solution.

2.5Dependencies

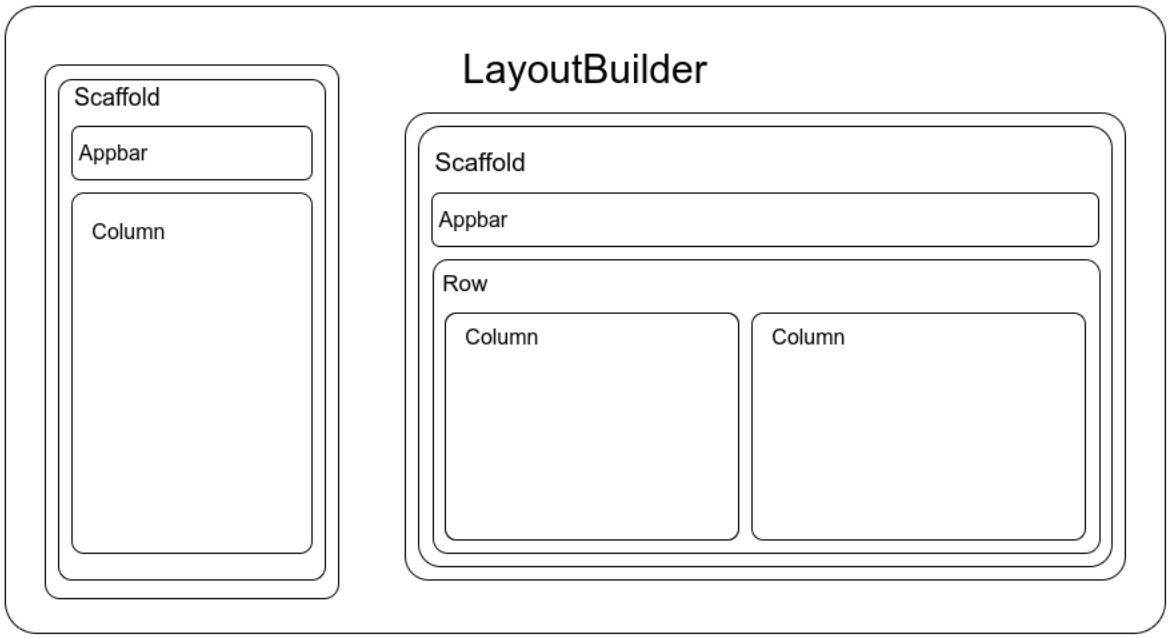
The most significant flutter packages included in the application are listed below.

Package	Description
cupertino_icons	iOS-style icons
fluttertoast	Display toast messages
get	State management and routing
flutter_svg	Render SVG images
firebase_core	Initialize Firebase services
firebase_auth	User authentication

google_sign_in	Google account authentication
firebase_ui_oauth_google	OAuth authentication for Google
google_maps_flutter	Google Maps integration
google_maps_flutter_web	Google Maps support for web
geocoding	Convert address to coordinates
flutter_dotenv	Manage environment variables
location	Get device location
flutter_polyline_points	Draw routes on Google Maps
http	Make HTTP requests
google_places_flutter	Google Places API integration
intl	Internationalization and date formatting
cloud_firestore	Cloud Firestore database
image_picker	Select images from gallery or camera
firebase_storage	Store files in Firebase Storage

2.6 Widget Architecture

This application is based on Flutter for UI development and uses componentized architecture for high reusability, modularity and efficient rendering. The file structure is as follows:



Component Hierarchy

1. Root Component (Root Widget)

o main.dart: the entry point of the application, initializes Firebase, routing, global configuration.

o firebase_options.dart: Firebase configuration file.
2. Screen Widgets
o Stored in the pages/ directory, contains the main pages of the application, such as:
o home.dart: the home page of the application, displaying maps and recommended places to visit.
o plan.dart: itinerary planning page that manages the user's travel plans.
o list.dart: trip list page, displaying user-created trips.
o login.dart / register.dart: user authentication page.
o train_booking.dart: the train booking page.
3. Reusable Components
o Stored in the components/ directory, they encapsulate reusable UI components, such as:
o attraction_card1.dart / attraction_card2.dart: attraction information card component.
o ticket_card.dart: train ticket information display component.
o yellow_button.dart: generic button component.
o my_textfield.dart: custom text input box.
4. Data Management
o data/ directory:
o listData.dart: stores local data related to the itinerary.
5. Services
o components/firestore_service.dart: encapsulate Firestore interaction logic, manage database operations.
o components/api_service.dart: encapsulate API calls, such as train data acquisition.
6. Routing
o routers/routers.dart: defines the navigation logic of the application and manages page jumps.
Architecture Features:
- Modular design: each page and function is encapsulated in an independent Widget to improve code reusability and maintainability.
- State management: Combine with GetX for page state management, optimize data transfer and interface refresh.
- Responsive Layout: MediaQuery, Flexible and other components are used to adapt to different screen sizes.
This architecture ensures good scalability, efficient data management and excellent user experience for cross-platform travel planning applications.

3. User Interface (UI)

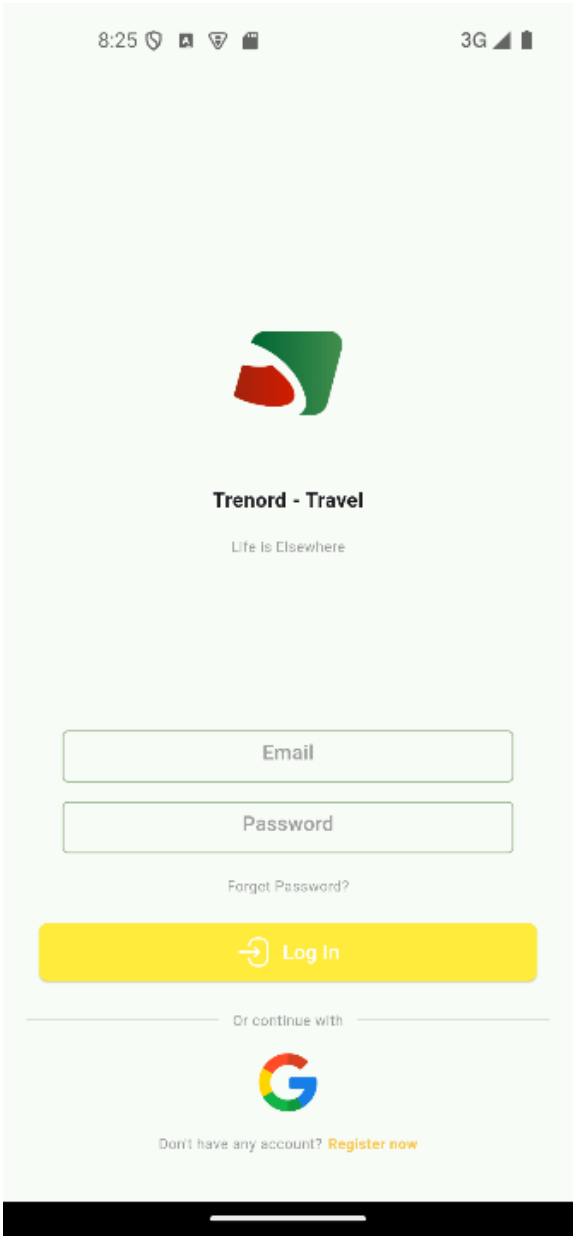
This section describes the user interface design of the application, including the overall UI design principles, the main page structure and component layouts. The application adopts Material Design specification to ensure consistency and good user experience. The main pages include home page, trip planning, trip list, train inquiry, personal center, etc. The functions and interactions of each page will be explained in detail in the subsequent sections. In addition, this section will also cover how responsive design is implemented and how to optimize the UI structure of the application through component reuse and navigation management.

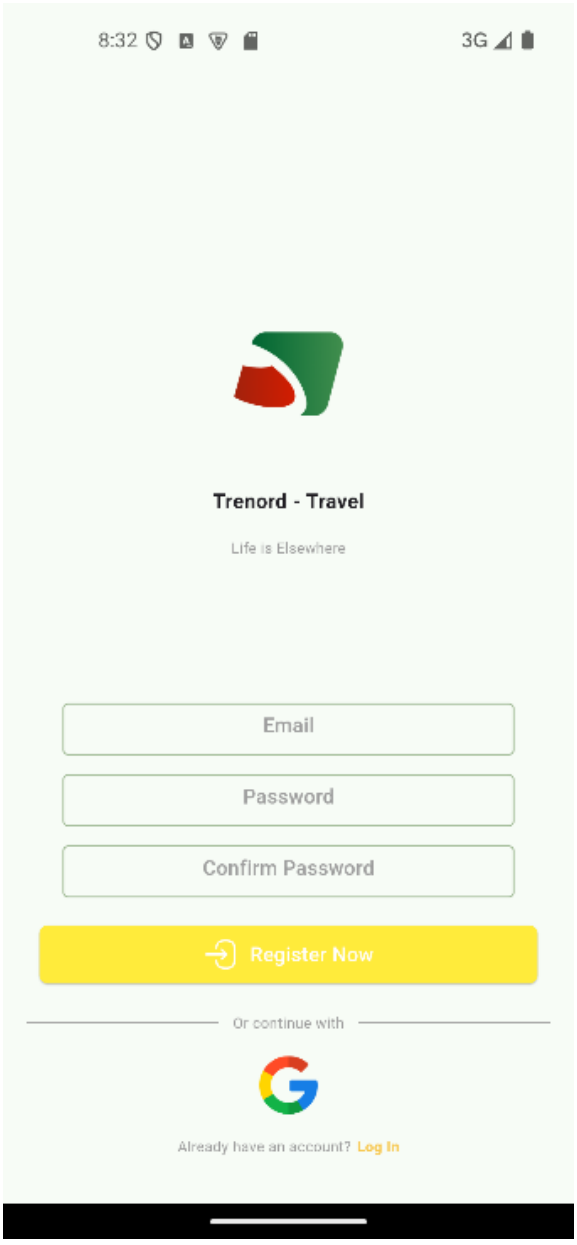
3.1. UI Design Choices

The UI design of this app follows Material Design specifications to ensure consistency, aesthetics and good user experience.
- Color style: A simple and intuitive color scheme is used, and different pages have a unified theme color to enhance the sense of visual hierarchy.
- Fonts and typography: Use easy-to-read fonts combined with proper spacing and alignment to enhance readability.
- Component style: All buttons, cards and input boxes follow Material Design specification to maintain consistency of interaction.
- Navigation design: Bottom Navigation and top AppBar are used to ensure users can quickly switch between pages.
- Responsive Layout: MediaQuery and Flexible components are used to adapt to different screen sizes to ensure good display on cell phones and tablets.
The overall UI design aims to provide a simple and intuitive interface to ensure that users can quickly get started and smoothly use the app's functions.

3.2. Main Screens

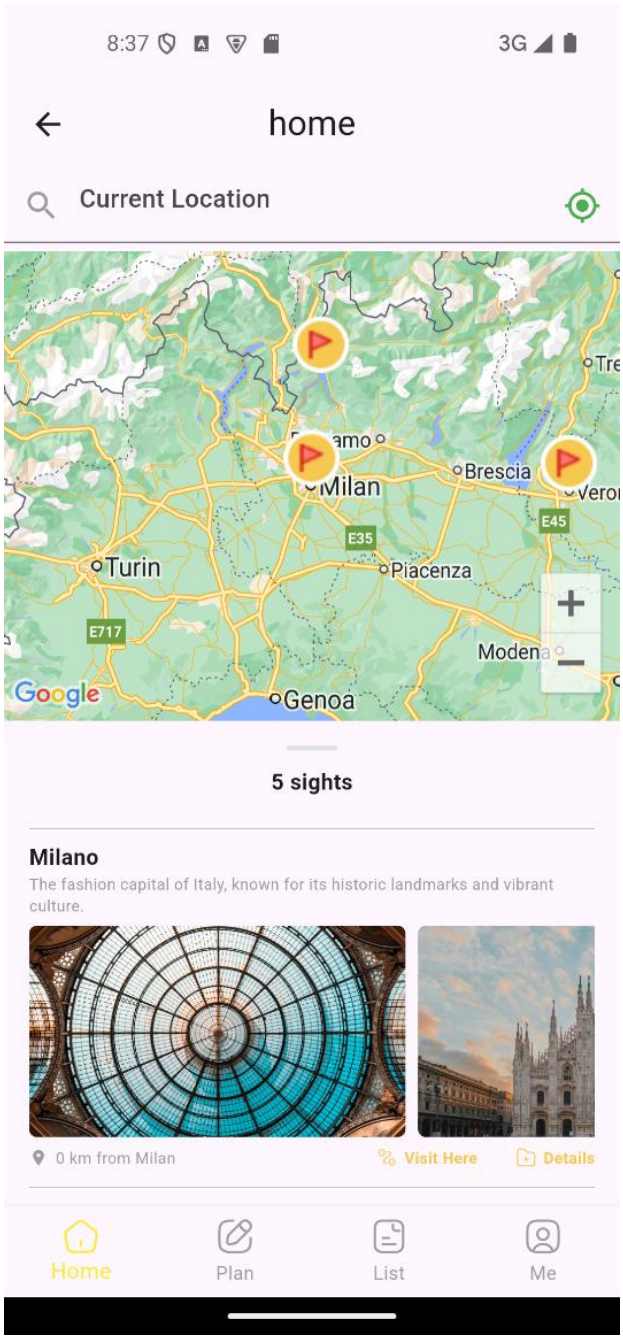
3.2.1. Login & Register





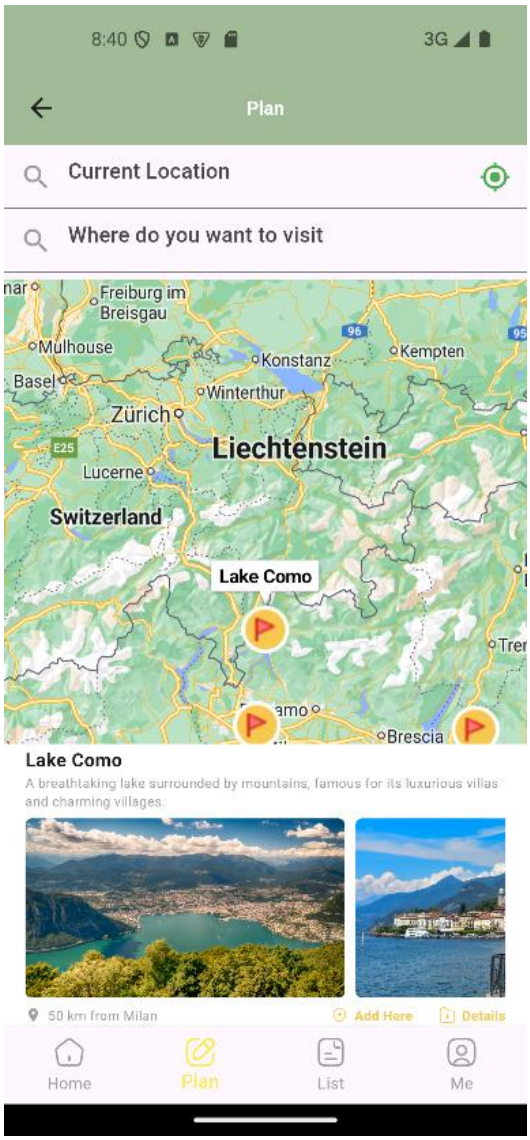
The login and registration page provides user authentication with a simple and intuitive UI design and supports email+password login and Google account login. The login page contains an email and password input box, where users can click “Log In” for authentication, and an option to forget password. The registration page contains an additional field to confirm the password, ensuring that the user enters the same information, and clicks “Register Now” to complete the account creation. Both provide a quick jump option to allow unregistered users to jump to the registration page or registered users to return to the login page, ensuring an efficient account management experience.

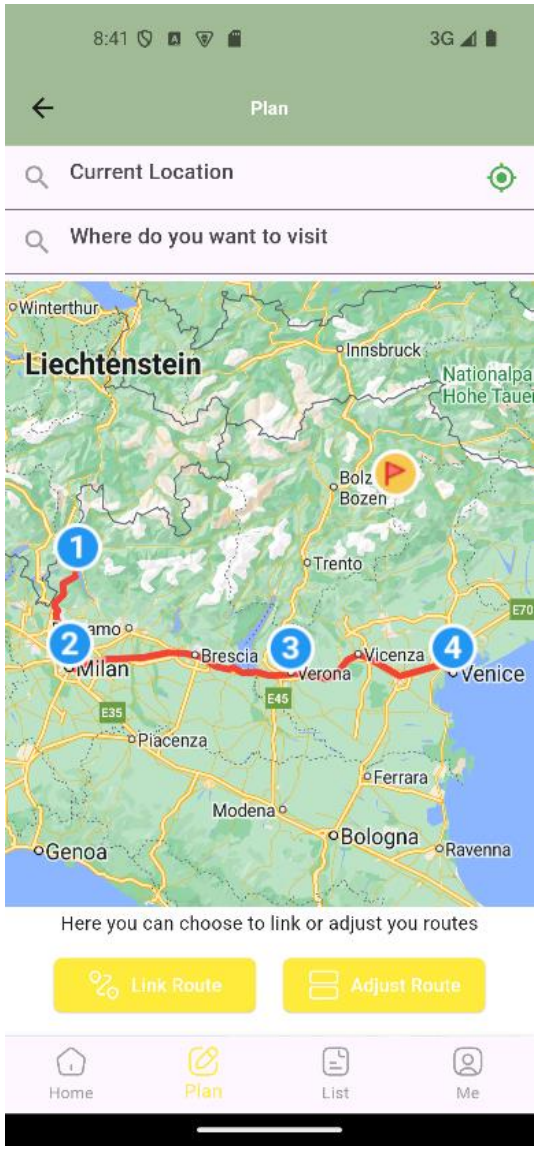
3.2.2. Home



The homepage provides a map view and a list of recommended attractions, allowing users to browse nearby attractions based on their current location and to find a specific place through the search function. Multiple attractions are labeled on the map and can be clicked on to view brief information, while the list area displays images, descriptions, and visit details for popular attractions. The details page displays high-resolution images, descriptions, opening hours, distances, and weather information for selected attractions, ensuring that users get a comprehensive travel reference. The Material Design interface, combined with intuitive card layouts and image displays, makes it easy to plan trips and learn more about the destination.

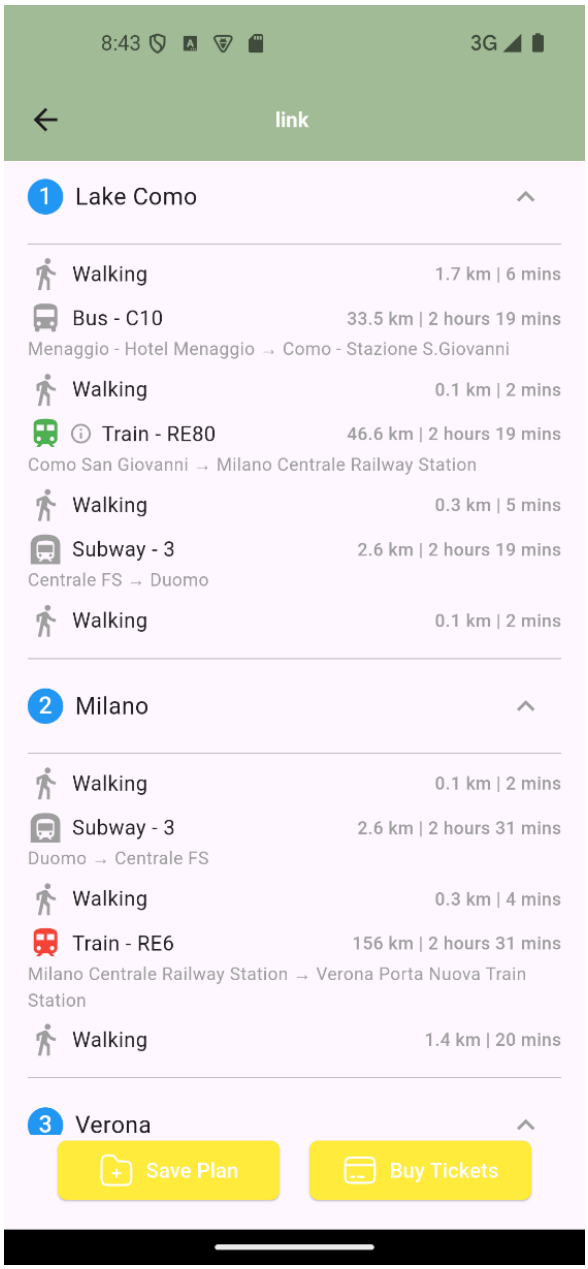
3.2.3. Plan

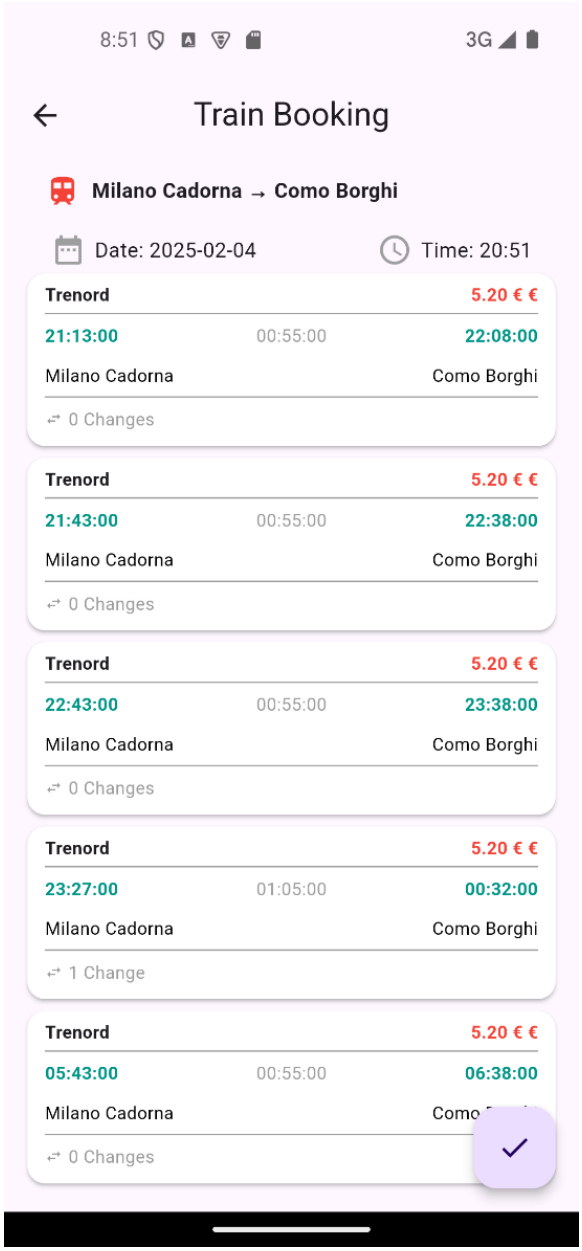




The Plan page allows the user to select a starting point and a destination, and to view recommended attractions and travel routes on a map. Users can enter the target location through the search box, and the map will display the selected attractions with detailed descriptions. Once the itinerary is confirmed, the system supports Link Route and Adjust Route, allowing the user to customize the itinerary. The intuitive map interface, combined with interactive marker points and route planning functions, helps users to efficiently manage their travel plans and optimize their travel routes.

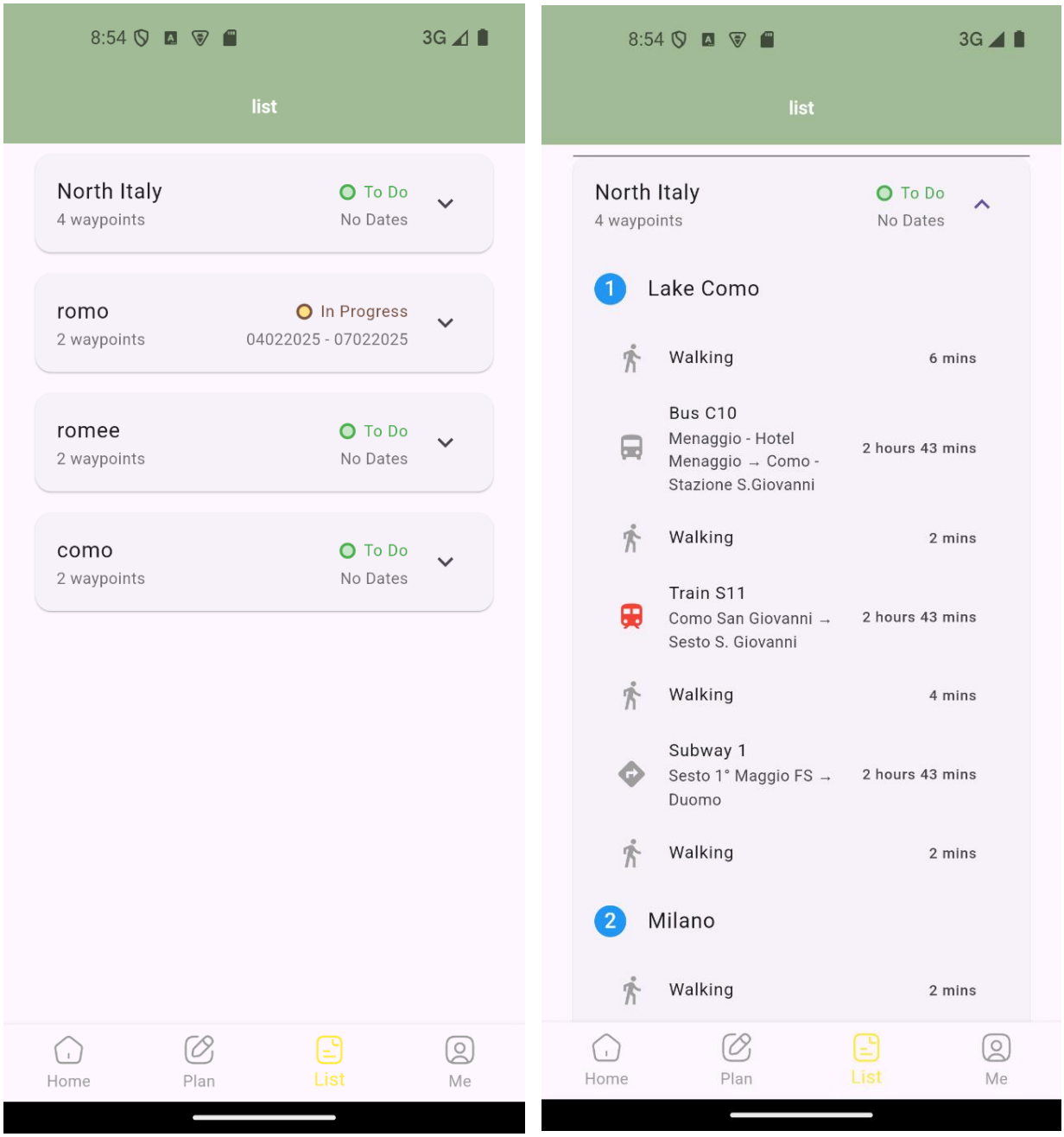
3.2.4. Route Linking and Train Booking





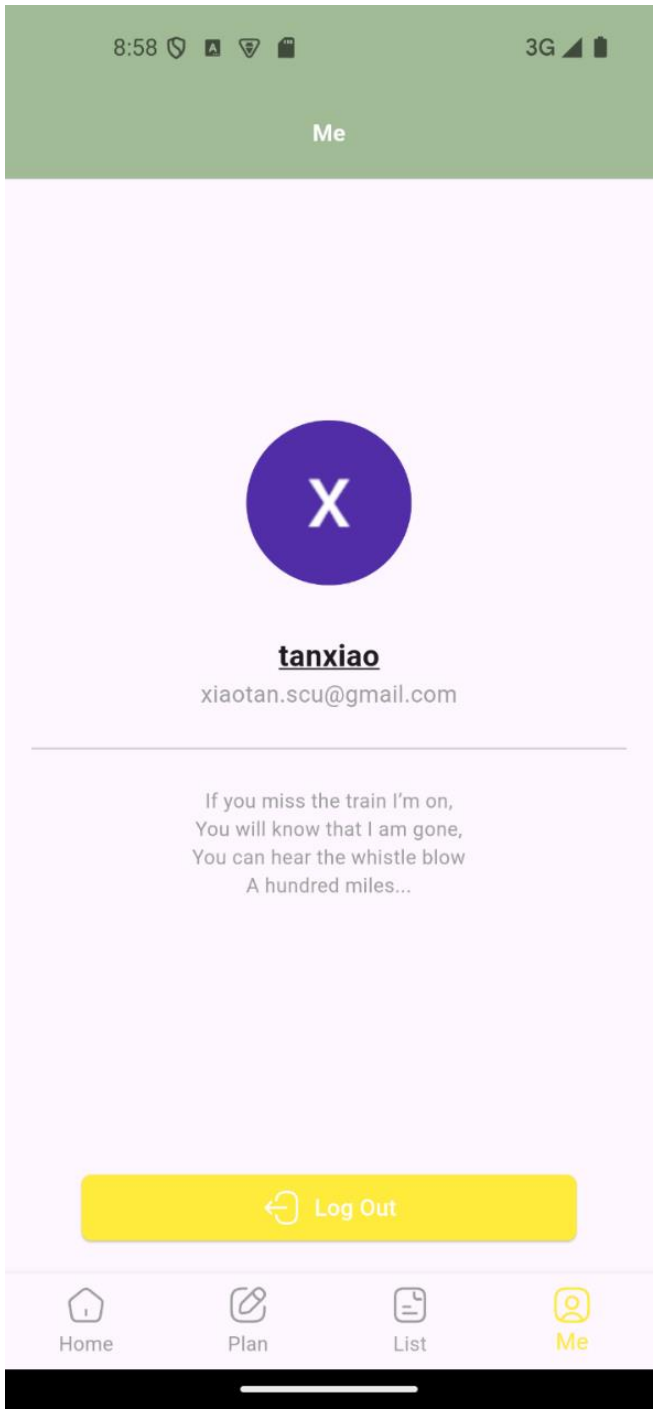
Route Linking and Train Booking pages provide users with detailed travel planning and train schedule information, while Train Booking page shows the departure and arrival times, journey duration and fares of different trains, allowing users to choose the right train and confirm the trip. Route Linking page integrates multiple modes of transportation such as walking, bus, train and subway, and provides interchange details, including passing stations, travel time and distance. Users can choose “Save Plan” to save the trip or click “Buy Tickets” to check the available travel time for efficient and convenient rail travel planning.

3.2.5. List



The List page is used to manage and view the travel plans created by the user, providing a clear overview of the itinerary. Users can view the name, number of stops, and current status (e.g. “To Do” or “In Progress”) of different trips. Clicking on an itinerary expands the detailed route, including transfers, walking distances and estimated times for each stop, helping users to efficiently plan and manage their travel arrangements. The interface design is simple and intuitive, ensuring that users can quickly access and adjust their travel plans.

3.2.6. Me

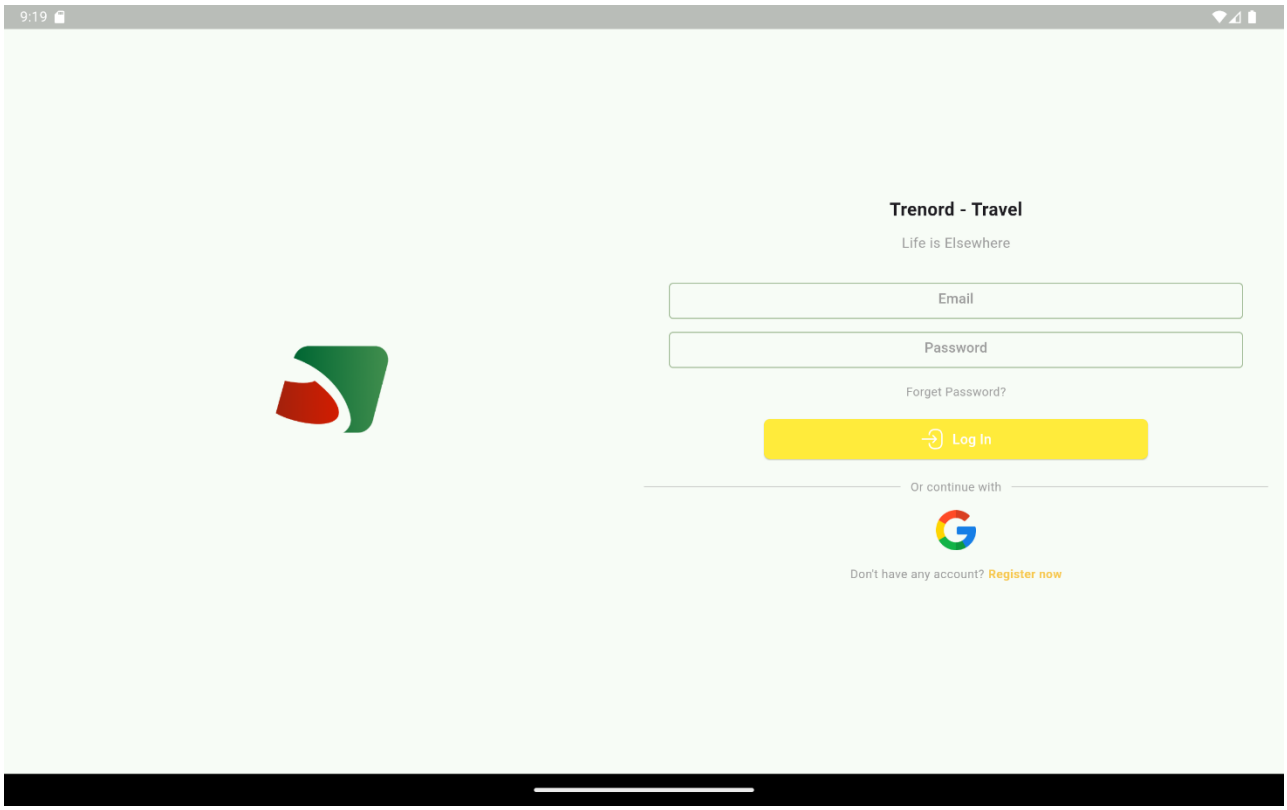


The Me page provides the user's personal information and account management features, including the user's avatar, name and registered email address, and displays the personal information in a clean layout. At the bottom of the page, a “Log Out” button is provided to allow users to securely log out of their accounts. The overall design follows the Material Design specification, keeping the interface clean and uncluttered so that users can quickly access and manage their account information.

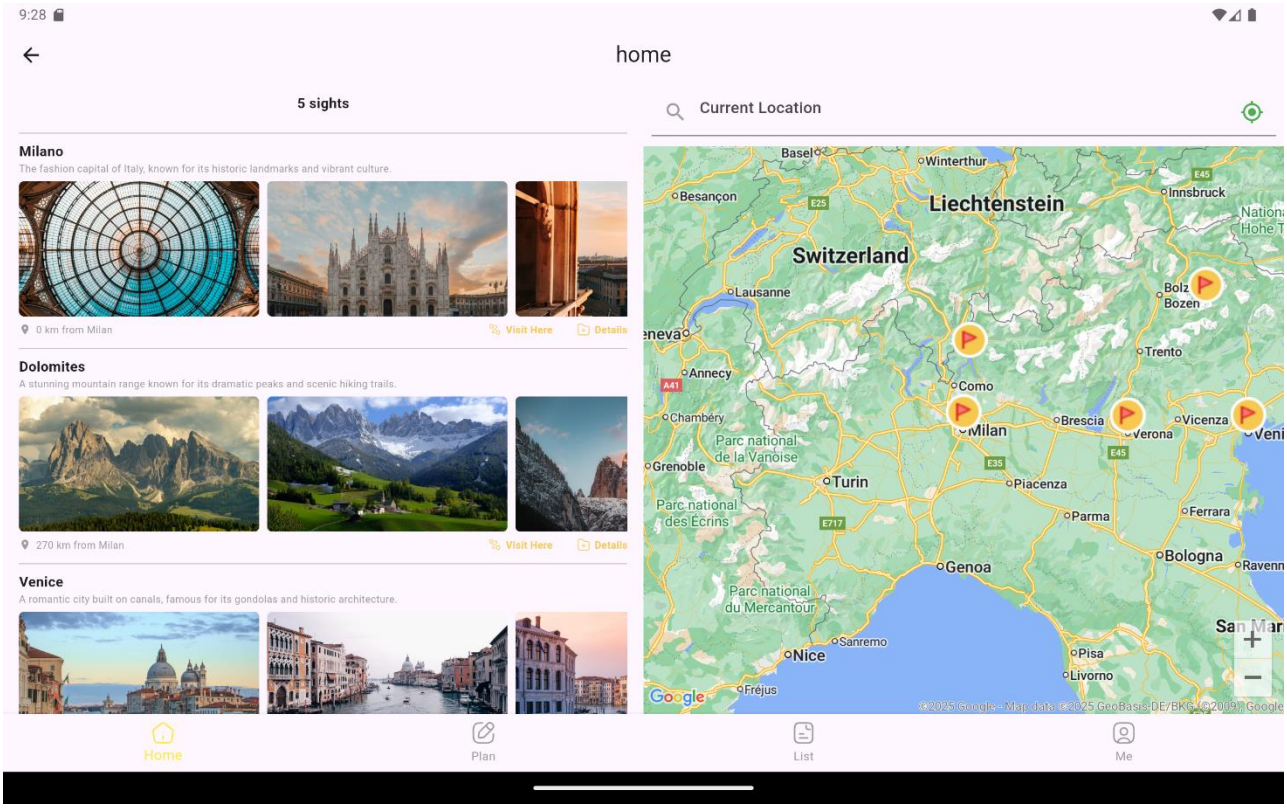
3.3. Tablet UI

The User Interface changes dynamically according to the screen size: this is done in order to improve the usability and the *look-and-feel* of the application. In this paragraph some of the most relevant page modifications w.r.t. the smartphone design is shown.

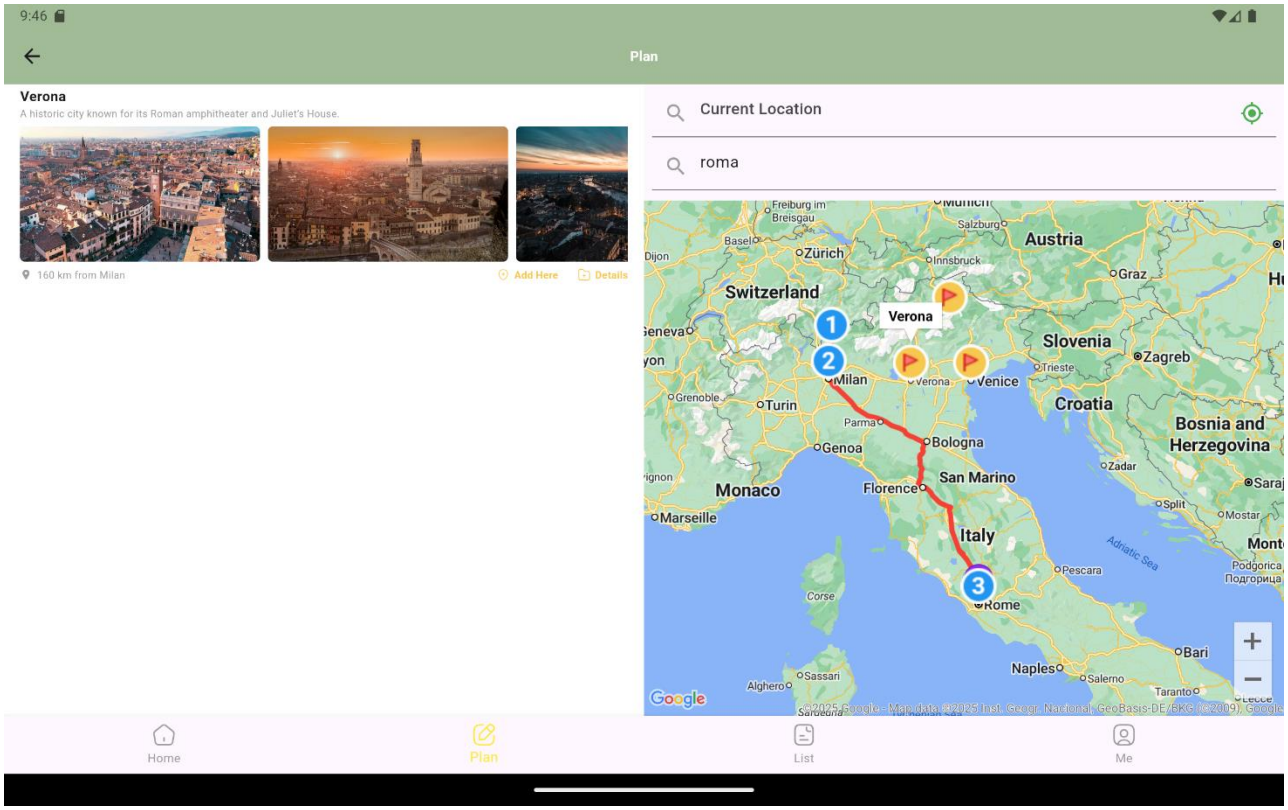
3.3.1. Login & Register



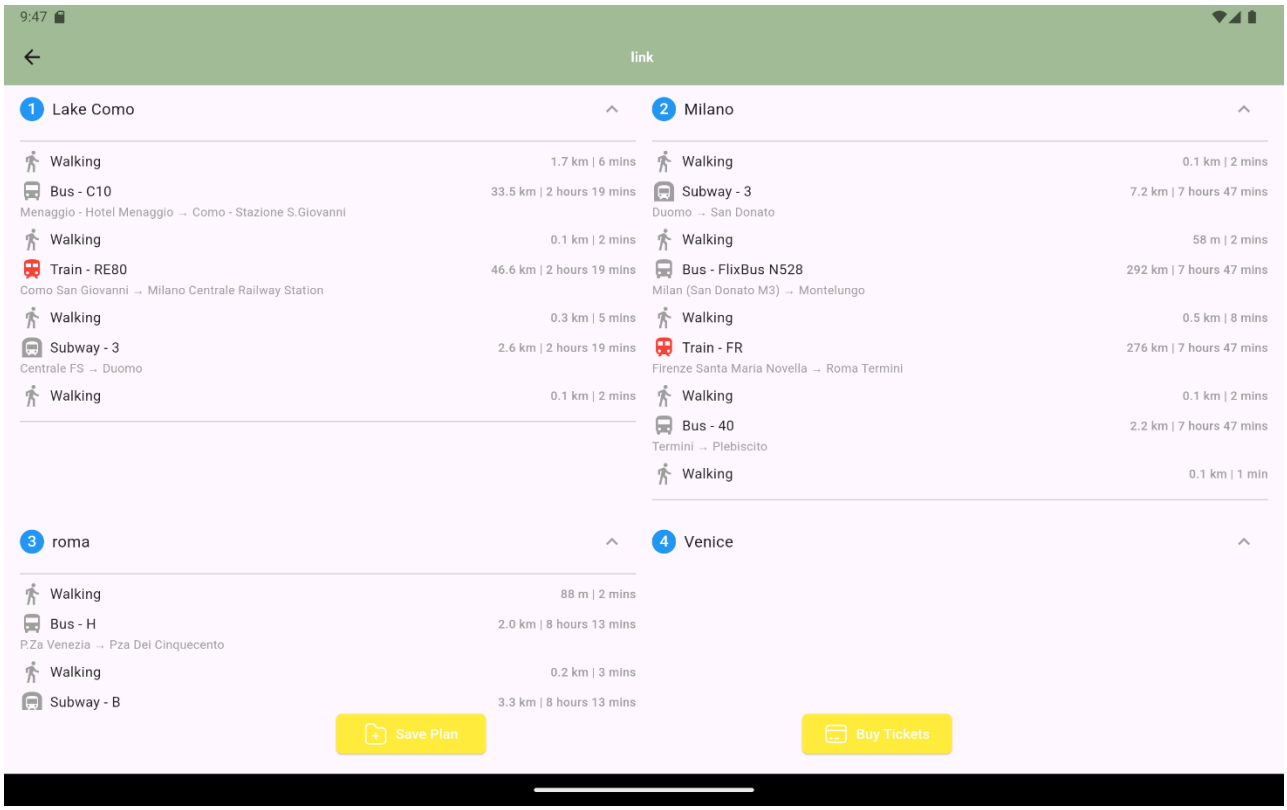
3.2.2. Home

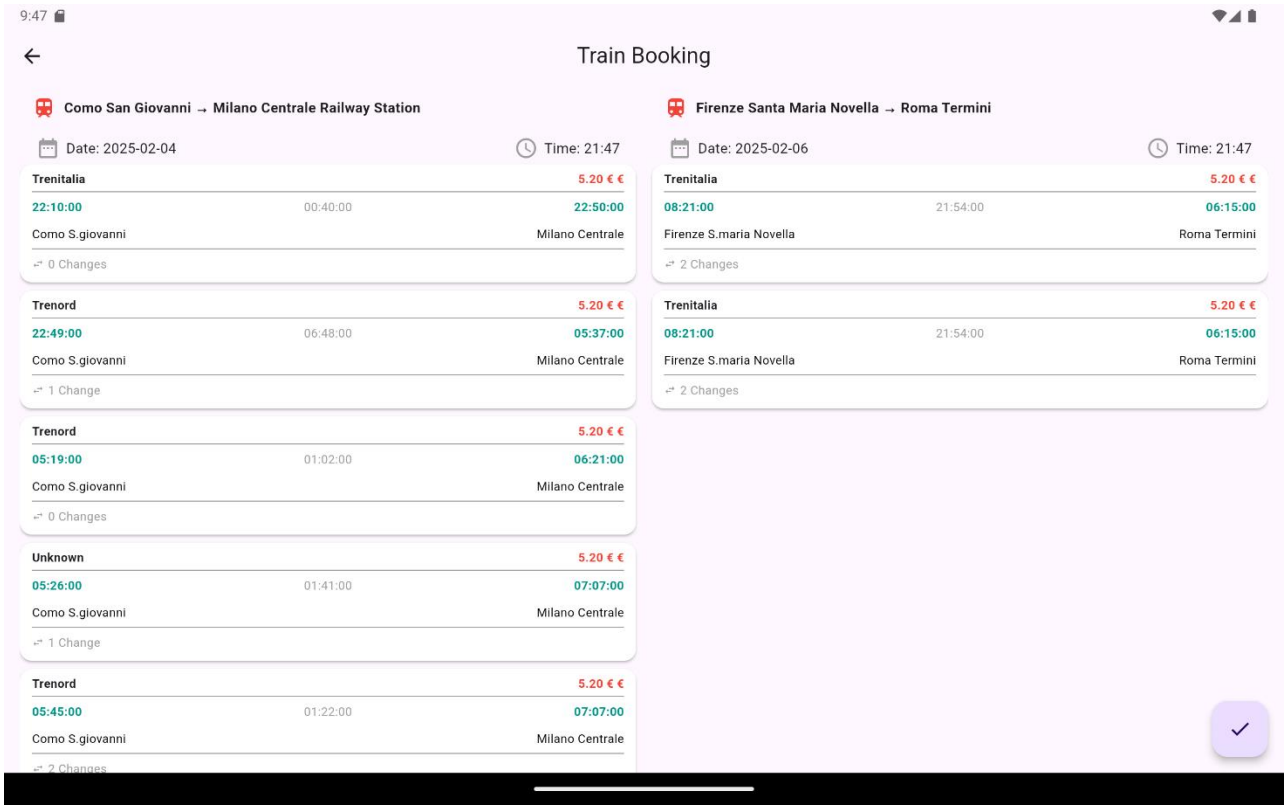


3.2.3. Plan

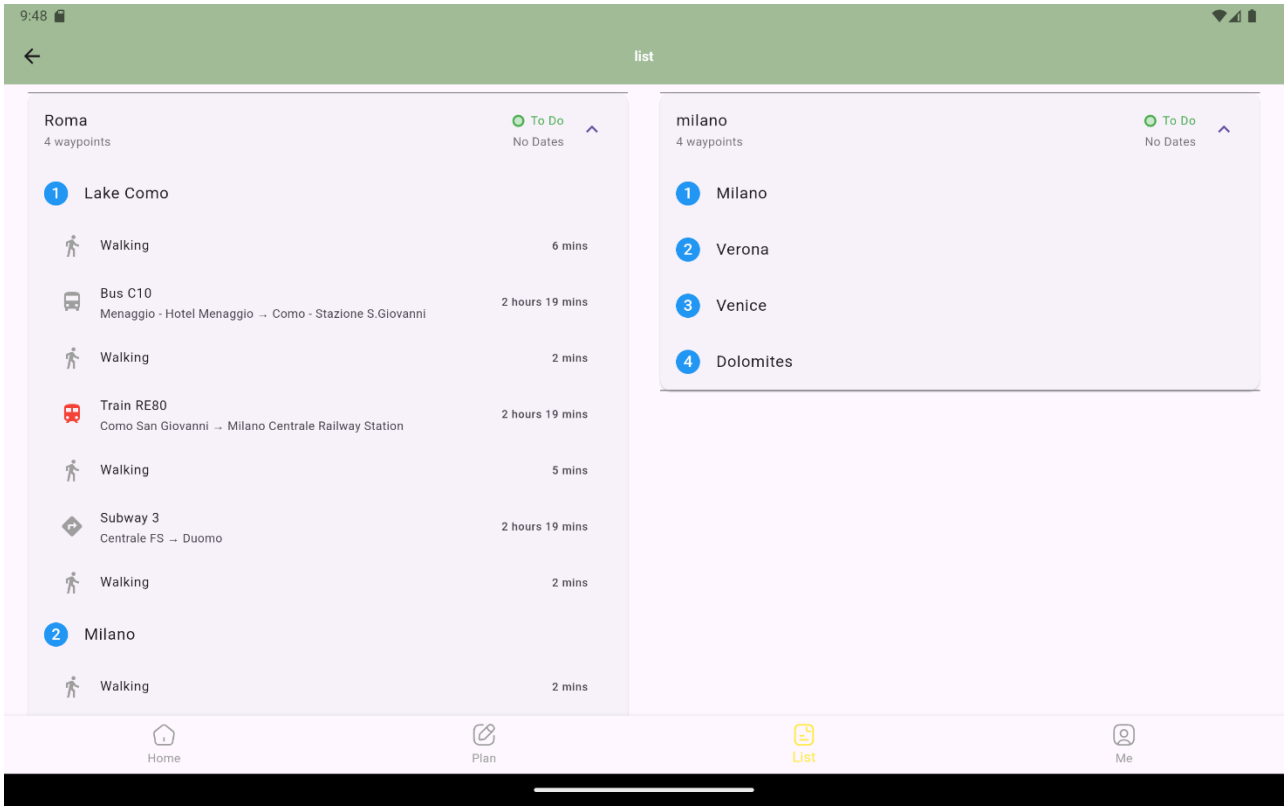


3.2.4. Route Linking and Train Booking

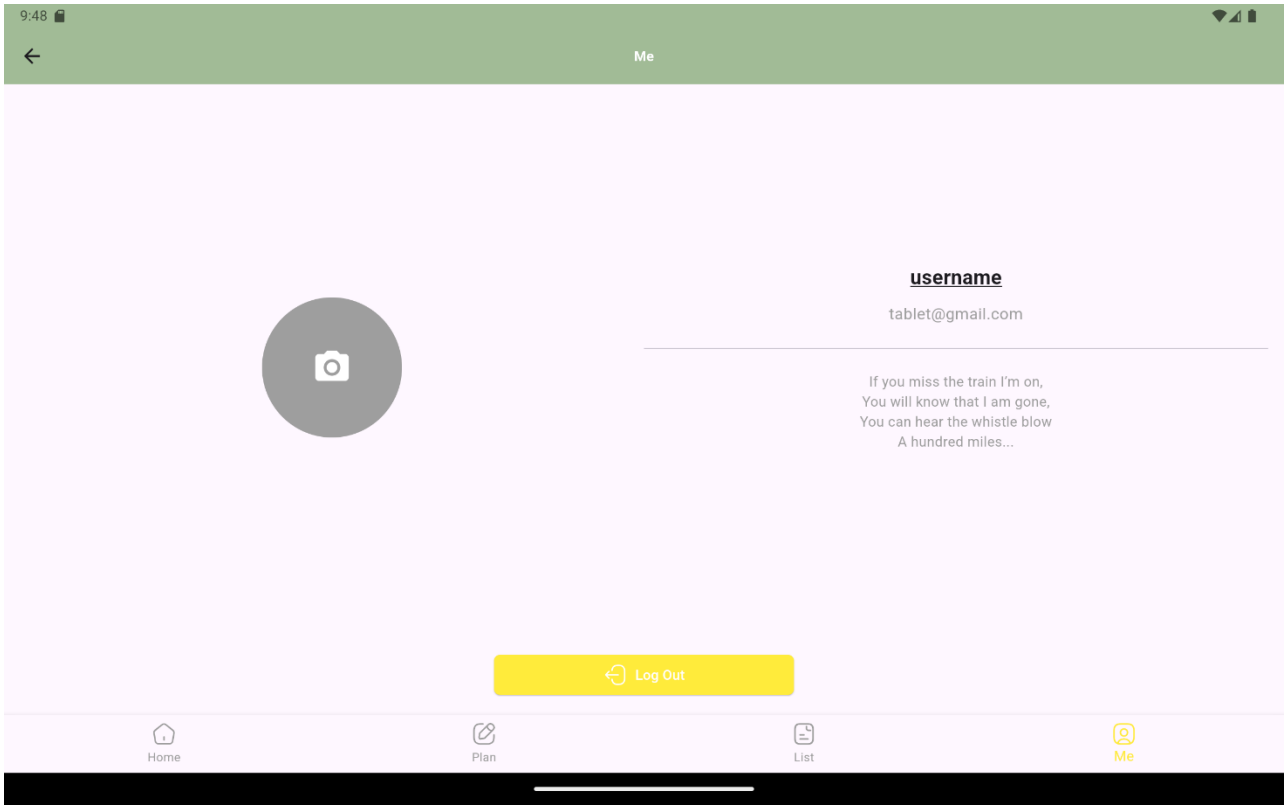




3.2.5. List



3.2.7. Me



4. Testing Campaign

4.1. Testing Environment

The testing environment of this application includes development environment, simulator testing and real device testing to ensure that the application runs stably on different platforms and devices.
- Development environment: Flutter + Dart is used for development, Firebase is integrated as the backend service, and all tests are conducted in Android Studio.
- Simulator testing: run on Android Emulator and iOS Simulator to test the compatibility of different screen sizes, resolutions and system versions.
- Real Device Testing: Using different brands and models of Android and iOS devices, we verified the performance, interaction experience and stability of the application in a real network environment.
- Testing tools: Flutter DevTools was used for debugging, Firebase Crashlytics was used to monitor crash logs, and Postman was used for API request testing.
This testing environment ensures high compatibility, stability and smooth experience on different platforms and devices.

4.2. Unit Test

Flutter's unit tests need to be run in a mock environment, and due to framework limitations, this application relies on multiple external services, and most of the logic is just calls to these services passing different parameters. Therefore, there are no reasonable unit tests to execute in this case.

4.3. Widget Test

The aim of this type of testing is to ensure that the proper widget is loaded and displayed correctly by simulating user interactions and validating UI components. Flutter Widget tests are executed in a **mock environment** due to framework constraints, where each screen is tested as a group of basic widgets to verify their behavior under different conditions.
The tests are performed on both **tablet and smartphone layouts** with minor modifications to accommodate design variations. Below is a list of the most relevant widget tests performed, categorized by screen:

Screen	Test Description
Login/Registration	Check registration with empty input
Login/Registration	Check registration with wrong input

Home Page	Check if the map and location markers load correctly
Home Page	Check if search functionality works properly
Plan Page	Check if user can add and remove destinations
Plan Page	Check if routes are displayed correctly
Train Booking	Check if train schedules are rendered properly
Train Booking	Check if price and time details are displayed correctly
List Page	Check if saved trips are displayed
List Page	Check if the trip details expand correctly
Me Page	Check if user profile information is rendered
Me Page	Check if logout functionality works properly

These tests ensure that the UI components render correctly, respond to user actions, and maintain a consistent experience across different devices.

4.4. Integration Test

Integration testing ensures that different modules of the application work together correctly, focusing on the interaction between UI components, backend services, and external APIs. The tests aim to verify data flow, authentication, and navigation logic to ensure a seamless user experience.

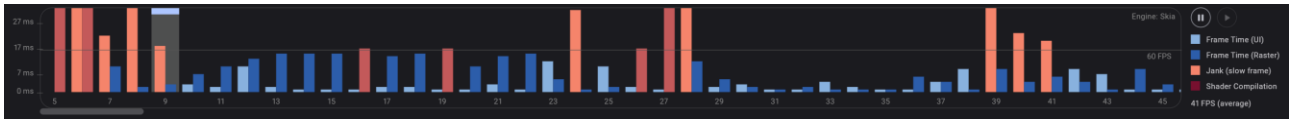
Testing Strategy:

- Authentication Flow: Ensures that login and registration processes work correctly with Firebase Authentication.
- Firestore Database Integration: Verifies that user data, trip plans, and saved destinations are correctly stored and retrieved.
- Google Maps API Interaction: Tests map rendering, location search, and marker placement.
- Trenord Train API Integration: Confirms that train schedule queries return valid results.

- Navigation and Routing: Ensures that page transitions and deep linking function correctly.
- User Data Persistence: Checks if local storage maintains user preferences and trip details.

Integration tests are executed in a simulated environment using Flutter integration_test and Firebase emulators, ensuring that all app components work together as expected.

4.5. Performance Testing



Performance testing using Flutter DevTools revealed that the application experiences occasional frame drops and performance inconsistencies, with an average FPS of 41, which is below the optimal 60 FPS for smooth animations. The presence of Jank (slow frames) and shader compilation delays indicates that certain UI components and animations are causing performance bottlenecks. High raster time suggests that some widgets require optimization to reduce rendering load. To enhance performance, improvements such as reducing unnecessary widget rebuilds, precompiling shaders, optimizing UI complexity, and improving image caching should be implemented. These optimizations will help achieve a smoother user experience with more stable frame rendering.

4.6. User Test

User testing was conducted to evaluate the overall usability, performance, and user experience of the application. The primary goal was to identify potential usability issues and gather feedback from real users to improve the app's functionality and interface.

Testing Process:

- Participant Selection: Users with different levels of technical expertise were selected to test the app.
- Test Scenarios: Users were asked to complete key tasks, such as registering an account, planning a trip, searching for train schedules, and navigating between screens.
- Feedback Collection: Participants provided feedback on ease of use, performance, and any encountered issues.

Key Findings:

- Positive Feedback: Users appreciated the intuitive UI, seamless navigation, and Google Maps integration.

- Challenges Identified: Some users found the train booking interface unclear, leading to UI adjustments.
- Performance Issues: Minor loading delays were reported in retrieving train schedules, requiring API optimizations.

Based on the user test results, improvements were implemented to enhance app usability and performance, ensuring a better experience for travelers.

5. Future Developments

The future development of the application will focus on expanding features, improving user experience, and enhancing system performance to better serve travelers. Several key areas of improvement have been identified to make the app more efficient, user-friendly, and comprehensive.

5.1. Real-Time Train Ticket Booking

Currently, the application only allows users to view train schedules. Future updates will integrate real-time train ticket booking, enabling users to purchase tickets directly within the app. This will require integration with Trenord's ticketing system and implementing secure payment gateways to facilitate transactions.

5.2. Advanced Trip Planning & AI-Powered Recommendations

To enhance travel planning, the app will introduce smart itinerary suggestions based on users' interests, previous trips, and popular destinations. AI-driven recommendations will offer optimized routes, suggest nearby attractions, and adjust plans based on real-time traffic or weather conditions.

5.3. Offline Mode & Data Caching

Many travelers may not always have stable internet access. A future update will implement offline functionality, allowing users to access their saved trips, previously viewed attractions, and train schedules without an active internet connection. This will be achieved through local storage caching and data synchronization when connectivity is restored.

5.4. Multi-Language & Accessibility Enhancements

Expanding the app to a global audience will require multi-language support beyond English. Future iterations will introduce language localization to support Italian, French, Spanish, and other commonly spoken languages among travelers. Additionally, accessibility features such as text-to-speech, font resizing, and color contrast adjustments will be introduced to ensure inclusivity.

5.5. Enhanced User Engagement & Social Features

To encourage greater user interaction, the app may introduce social features, such as the ability to share itineraries with friends, collaborate on trip planning, and post travel experiences. A rating and review system for destinations could also be integrated, helping users make informed decisions based on community feedback.

5.6. Performance & Security Optimizations

As the application scales, improving performance and security will be critical. Future updates will optimize database queries, reduce API response time, and implement background data processing to enhance app speed. Security enhancements such as two-factor authentication (2FA) and end-to-end encryption for user data will also be considered.

5.7. Integration with More Travel Services

To create a more seamless travel experience, potential integrations with hotel booking platforms, public transport services, and car rental agencies will be explored. This will allow users to plan their entire trip within a single app, eliminating the need to switch between multiple services.

By implementing these future developments, the application aims to become a comprehensive travel assistant, providing users with a seamless, intuitive, and feature-rich travel planning experience.