iJRASET
International Journal For Research in Applied Science and Engineering Technology



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 14 **Issue:** XI **Month of publication:** November 2026

**DOI:** https://doi.org/10.22214/ijraset.2026.85619

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue XI Nov 2026- Available at www.ijraset.com*

# EXPLORA: A Cross-Platform Mobile Application for Matching Compatible Travel Companions Using Proximity and Preference-Based Filtering

Gunjan Agrawal
*B.Tech. in Computer Science and Engineering (AI & ML), GL Bajaj Group of Institutions, Mathura, India Supervisor: Er. Pramod Kumar*

***Abstract: Planned trips frequently fall through when a companion cancels at the last minute, and solo travellers often struggle to find reliable people to travel with. This report presents Explora, a cross-platform mobile application that connects travellers who share destinations, travel dates, budgets and travel styles, and who are geographically close to each other.***
***Explora is built with Flutter (Dart) for the user interface and Google Firebase (Authentication, Cloud Firestore, Cloud Storage, Cloud Functions and Cloud Messaging) for the backend. A rule-based matching module compares the preference profiles of users and returns a compatibility percentage; an interactive map built with Flutter Map shows nearby travellers within an adjustable search radius. The application also provides real-time one-to-one and group chat, collaborative itinerary planning, shared-expense tracking, and a post-trip rating and reporting system intended to build community trust.***
***The report documents the requirements, architecture, data model, implementation and testing of the system, and evaluates it through functional testing and user acceptance testing with a small group of beta users. It also discusses the limitations of the current rule-based approach and outlines a path towards learned, collaborative-filtering recommendation.***
***Keywords: travel companion matching; mobile application; Flutter; Firebase; location-based services; recommender systems; collaborative trip planning.***

**ACKNOWLEDGEMENT**
I express my sincere gratitude to my project supervisor, Er. Pramod Kumar, for his guidance and encouragement during the development of Explora.
I also thank the Department of Computer Science and Engineering, GL Bajaj Group of Institutions, Mathura, for the academic environment and resources provided, and the beta testers whose feedback improved the application.
Explora was carried out as a B.Tech. group project. I thank my project teammates, Yash Pratap Mathuria, Rohit Raj Khanna and Palak Gupta, for their participation in the project.

## I. INTRODUCTION

### *A. Background and Motivation*

Travel is increasingly planned and booked online, yet finding a trustworthy companion remains an informal, word-of-mouth process. When a friend withdraws from a trip, the remaining traveller must either cancel, travel alone or absorb the full cost of shared accommodation and transport. Explora addresses this gap by making companion discovery a first-class feature of a mobile application. The main motivations are:

*1)* Better travel experiences – shared journeys are more enjoyable and memorable.
*2)* Safety – travelling with a vetted companion adds support in unfamiliar destinations.
*3)* Lower cost – shared lodging and transport reduce the cost per traveller.
*4)* Community and cultural exchange – matching travellers from different backgrounds encourages new friendships and shared learning.
*5)* Flexibility – the platform supports both weekend getaways and long-term journeys.

©IJRASET: All Rights are Reserved 

### B. Problem Statement

Existing channels for finding travel partners (social-media groups, forums and generic chat apps) lack structured matching, make it hard to verify who one is talking to, and do not combine discovery with trip planning and cost sharing. The problem addressed in this project is to design and build a mobile application that (i) matches travellers on structured criteria and on proximity, (ii) supports secure communication, and (iii) helps matched travellers plan and fund a trip together.

### C. Objectives

1) Design a matching module that connects travellers by destination, dates, budget, travel style and location.
2) Provide safety features: authenticated accounts, profile information, ratings, reviews and a reporting mechanism.
3) Provide real-time one-to-one and group messaging.
4) Support collaborative itinerary planning with an interactive map.
5) Provide expense-tracking and cost-splitting tools to reduce travel costs.
6) Evaluate the application through functional testing and user acceptance testing.

### D. Related Work

Tourism recommender systems have been studied extensively; Borràs et al. [1] survey intelligent tourism recommenders that suggest destinations, attractions and routes, and Lu et al. [2] review the wider landscape of recommender-system applications. Most of this work, however, recommends *places* to a user. Explora instead recommends *people* to a user, and ties that recommendation to a concrete trip. Existing travel-companion platforms share a common set of features, which Explora also adopts and combines:

| Feature | Typical implementation in companion-finding platforms |
|---|---|
| Matching | Filters on destination, travel dates, activities and budget. |
| Profiles | Bio, photos, interests and trip plans so that users can judge compatibility. |
| Social features | Messaging, discussion boards, and event listings. |
| Location | GPS-based discovery of nearby travellers and points of interest. |
| Cost sharing | Shared accommodation, transport and activity costs. |

Explora's distinguishing choice is to integrate all five in a single cross-platform application, with proximity-aware matching, real-time chat, collaborative itineraries and built-in expense splitting.

### E. Contributions

1) A requirements specification and modular architecture for a travel-companion platform.
2) A rule-based, explainable compatibility-scoring module that combines trip and proximity criteria.
3) A working Flutter/Firebase implementation with real-time chat, map-based discovery, itinerary planning and expense splitting.
4) A test and user-acceptance evaluation, together with an analysis of limitations and a roadmap for learned recommendation.

### F. Organisation of the Report

Chapter 2 specifies the requirements. Chapter 3 presents the system design, including architecture, data flow, class, sequence and database diagrams. Chapter 4 describes the implementation and user interface. Chapter 5 reports testing and evaluation. Chapter 6 concludes and discusses limitations and future work. References and appendices follow.

## II. REQUIREMENTS ANALYSIS

This chapter defines what Explora must do (functional requirements), how well it must do it (non-functional requirements) and how it should look and behave (user-interface requirements).

### A. *Functional Requirements*

| Area | Requirement |
|---|---|
| Registration and login | Sign-up with e-mail and password (and, as planned extensions, mobile-number OTP and social accounts) through Firebase Authentication; password reset and account recovery; login-attempt limiting. |
| User profile | Name, age, gender, nationality, bio, languages, travel interests, preferred destinations, budget and availability; optional profile picture; profile can be edited at any time. |
| Matching | Filter-based matching on destination, travel style (e.g. adventure, leisure, cultural), budget and dates; a compatibility percentage; matches update when a profile changes. |
| Search and filters | Search by location, travel dates, activities, language and age preference; combinable, customisable filters. |
| Messaging | One-to-one and group text chat, image sharing, timestamps and read receipts; voice/video as a future extension. |
| Trip planning | Create trips and invite companions; day-wise itineraries, notes and checklists; co-editing by all participants; export (PDF/ICS) and calendar integration as planned features. |
| Safety and verification | E-mail verification; optional ID and social-profile verification; post-trip ratings and reviews; flag/report feature for inappropriate behaviour. |
| Expense sharing | Record costs for lodging, transport and food; split equally or manually; per-user balance summary. |
| Events and activities | Suggestions based on trip location, interests and local festivals (planned extension). |
| Payments (optional) | Encrypted payment gateway (e.g. Razorpay or Stripe) for shared bookings, with payment history and refund tracking (planned extension). |

### B. *Non-Functional Requirements*

| Attribute | Requirement |
|---|---|
| Performance | Key screens (profile, search results) load in under 3 seconds; backend responds in 1–2 seconds for normal requests; supports 500+ concurrent users. |
| Scalability | Supports a growing user base using Firebase's managed, horizontally scalable services and real-time synchronisation. |
| Security and privacy | Sensitive data (credentials, identity documents, payment data) is encrypted in transit and at rest; Firestore security rules restrict access; data handling follows the principles of consent, minimisation and user control. |
| Usability | Intuitive navigation and a clean, responsive layout on phones and tablets. |
| Availability | Target availability of 99.9 %, with regular backups and a recovery plan. |
| Compatibility | Android and iOS through a single Flutter code base. |

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

### *C. User-Interface Requirements*

*1)* Responsive design across screen sizes.
*2)* Profile page with clear sections for personal information, trip details and photos.
*3)* Search interface with a search bar and filters for destination, dates and preferences.
*4)* Messaging with simple real-time chat.
*5)* Match results that show compatibility scores and key details of each candidate.

## III. SYSTEM DESIGN

Explora is a mobile-based travel-companion platform built on a modular architecture, so that matching, planning, messaging and mapping can be developed, tested and scaled independently.

### *A. System Architecture*

The system follows a client–backend architecture (Figure 3.1). The Flutter client sends user input (profile data, preferences, trip data and messages) to Firebase services, which store the data and return filtered results. The mapping module retrieves and renders route and location data using Flutter Map.

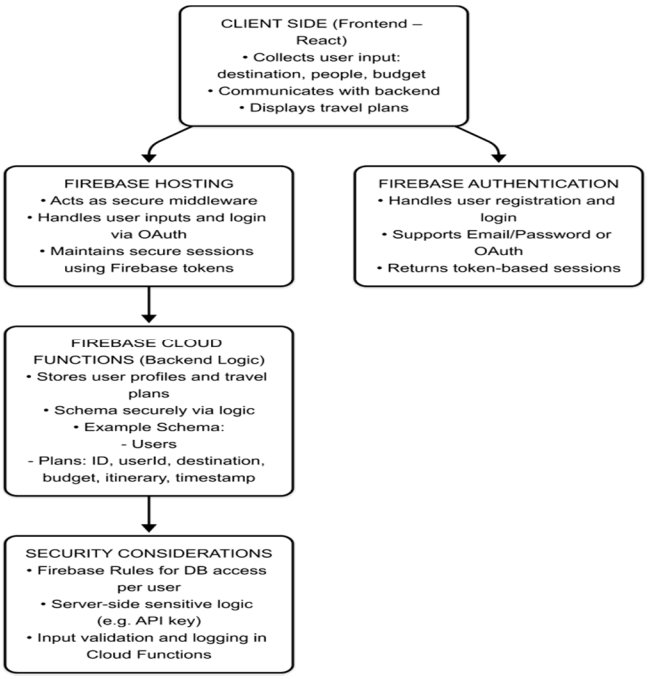



Figure 3.1 System architecture of Explora

### *B. Frontend Design*

The frontend is developed in Flutter (Dart), which allows one code base to target both Android and iOS with a responsive layout. Its main features are summarised below.

| Feature | Description |
|---|---|
| User login | Secure sign-up and login for new and existing users. |
| Profile management | Edit personal details, preferences and trip plans. |
| Match interface | Shows suggested travel companions based on entered preferences. |
| Feature | Description |
| Chat | Send and receive messages and notifications about matches and trips. |
| Trip planner | Create and view personal or shared itineraries. |
| Map integration | Uses Flutter Map to display routes, nearby users and trip locations. |

### C. Backend Design

The backend is built on Firebase, which provides a real-time database, authentication and cloud storage without server management. It handles data transactions, access control and communication between users.

| Component | Purpose |
|---|---|
| Firebase Authentication | Registration, login and e-mail verification. |
| Cloud Firestore | Stores user profiles, trips, expenses and chat messages. |
| Cloud Functions | Server-side logic such as notifications and automatic updates. |
| Cloud Storage | Stores profile images and shared files. |
| Cloud Messaging (FCM) | Push notifications for matches, trip changes and messages. |

### D. Map and Location Services

Explora integrates Flutter Map to let users view potential companions near their location, plot and modify routes, and see destinations and points of interest on an interactive map.

| Functionality | Description |
|---|---|
| Real-time location view | Users may share and view companion locations (opt-in). |
| Nearby travellers | Travellers within an adjustable search radius are shown with their distance. |
| Trip route display | Draws suggested routes for planned trips. |
| Points of interest | Shows nearby attractions and services as map markers. |

### E. Modules and Functional Breakdown

| Module | Key features |
|---|---|
| User management | Sign-up by e-mail, profile with travel details, editable at any time. |
| Search and matching | Filter-based search by destination, budget and dates; shows a compatibility percentage. |
| Itinerary and budget | Collaborative day-wise itineraries, shared notes, expense tracking and cost splitting. |
| Chat and communication | Secure real-time messaging between matched users. |
| Map and navigation | Location display, saved routes and marked destinations. |
| Database management | Storage and retrieval of profiles, trips and messages. |

### F. Data Flow Diagram

Figure 3.2 shows the level-1 data flow. Users submit preferences, which the application processes and stores in the database; users then receive match notifications, access trip information and use the chat and map features.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

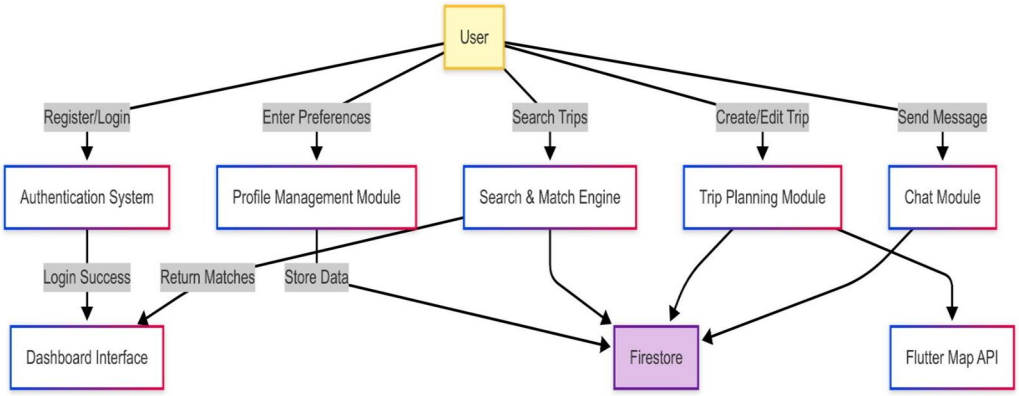



Figure 3.2 Level-1 data flow diagram

*G. Class Design*

Figure 3.3 outlines the main classes and relationships: a User (name, contact information, preferences), a Preferences object (destination, activities, budget, group size), a Trip with an itinerary and budget, a Match that links users with similar preferences, a Chat for messaging, and map-location data.

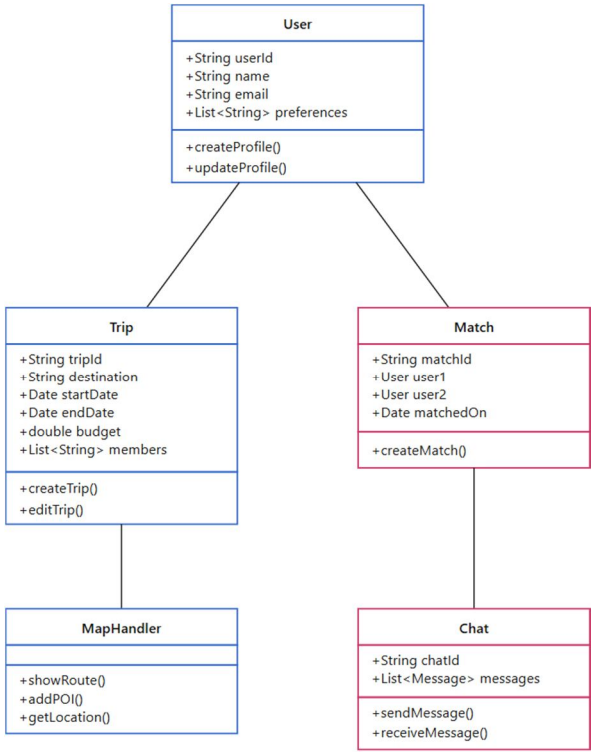



Figure 3.3 Class (UML) diagram

*H. Sequence Diagram*

Figure 3.4 shows the interaction between the user, frontend, backend and database for trip creation:

*1)* The user opens the app and logs in.
*2)* The frontend sends the credentials to Firebase Authentication.
*3)* Firebase validates them and returns a success response.
*4)* The user selects “Create Trip”; the inputs are passed to the backend.
*5)* The backend stores the trip data in Firestore.
*6)* The app shows a success message and displays the new trip route on the map.

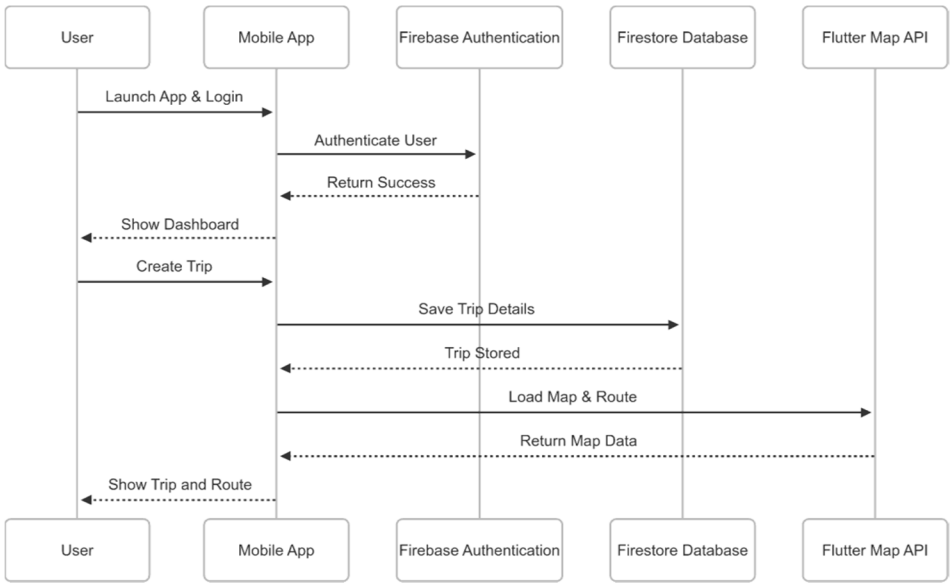



Figure 3.4 Sequence diagram for login and trip creation

*I. Data Schema*

Figure 3.5 shows the logical data schema linking users, preferences, chat messages and map locations to the database.

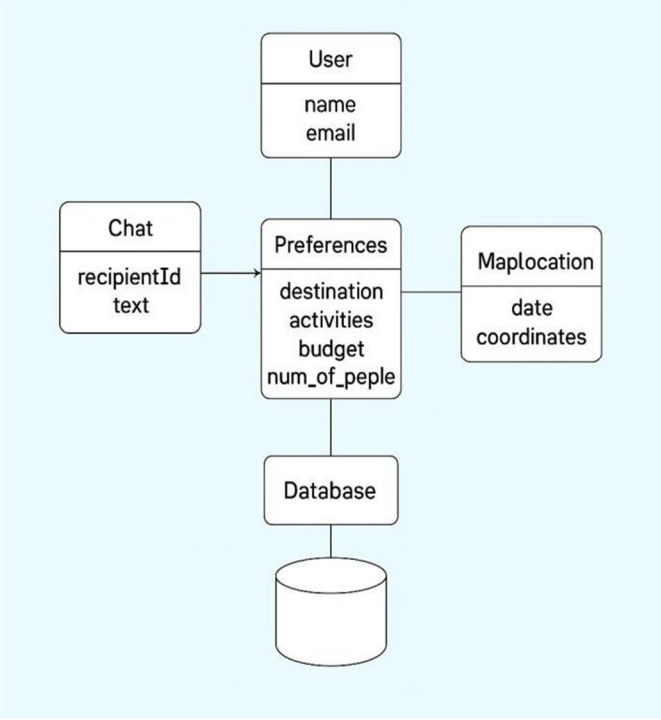



Figure 3.5 Data schema

## IV. IMPLEMENTATION AND USER INTERFACE

Explora was implemented with Flutter (Dart) for the frontend and Firebase for the backend. The implementation focused on user-centred features, backend integration and scalable data handling, so that the application supports real-time interaction and cross-device access.

### A. Technology Stack

| Layer | Technology | Purpose |
|---|---|---|
| Frontend | Flutter (Dart) | Cross-platform mobile user interface |
| Backend | Firebase (Firestore, Authentication) | Real-time database and user authentication |
| Mapping | Flutter Map | Route visualisation and nearby-traveller display |
| Messaging | Firebase Cloud Messaging (FCM) | Push notifications and in-app chat |
| Storage | Firebase Cloud Storage | Profile images and itinerary files |

### B. Companion Matching

The matching module is deliberately rule-based and explainable: it does not use a trained model. It compares the preference values entered by two users against the selected filters and produces a compatibility percentage that is shown next to each suggested companion.

| Component | Description |
|---|---|
| Matching criteria | Destination, travel dates, budget, gender preference and travel type (leisure, adventure, cultural). |
| Compatibility score | The number of criteria on which two profiles agree is converted to a percentage; the score is recomputed when either profile changes. |
| Proximity | Nearby travellers are listed with their distance and can be limited to an adjustable search radius (for example, 50 km). |
| Discovery | Users can browse all matches or be shown the most compatible profiles first. |

A rule-based approach was chosen because the platform starts without interaction data: it is transparent to users, easy to test and requires no training data. Chapter 6 discusses how learned recommendation could be added once enough data exists.

### C. Feedback and Trust Module

To build community trust, users can rate and review companions after a trip. Ratings use a 1–5 scale with optional comments; feedback is shown on user profiles to help future companions; and users can report inappropriate behaviour for review.

### D. Expense-Sharing Module

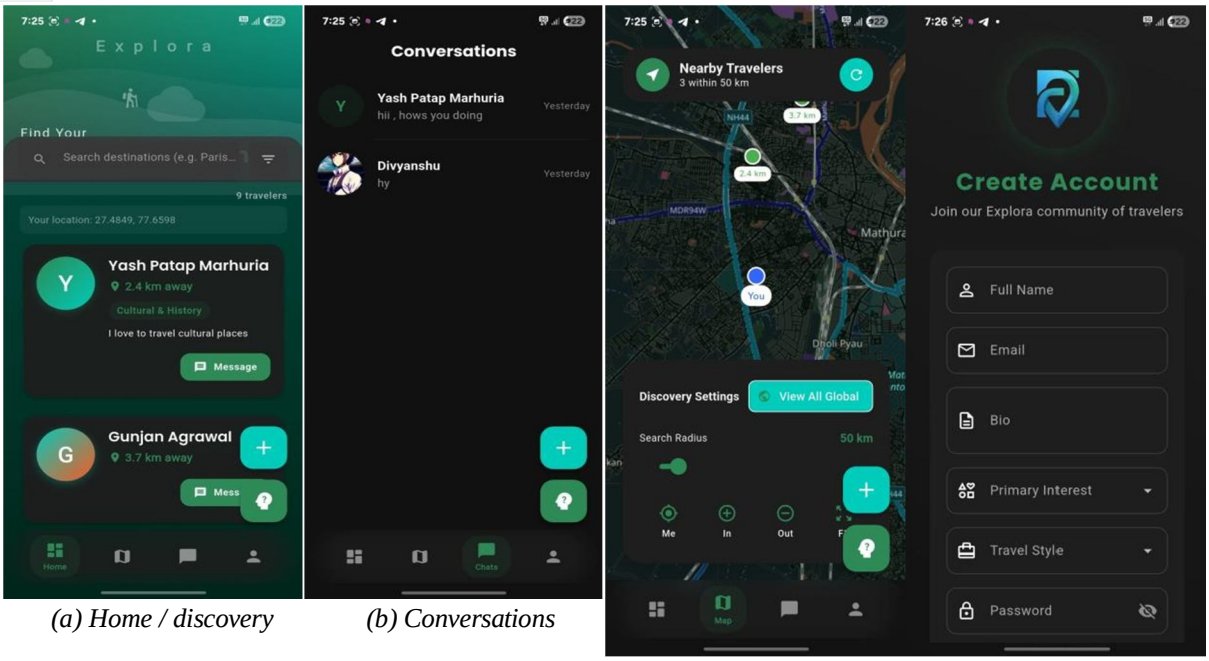
Group members record costs such as lodging, transport and food. Costs can be divided equally or assigned manually to specific users, and a summary view shows each person’s contribution and balance. The summary can be exported as a PDF or viewed in the app.

### E. User-Interface Walkthrough

The interface is built from Flutter widgets with a dark, responsive theme and a bottom navigation bar (Home, Map, Chats, Profile). Figure 4.1 shows four key screens.

- Home/ discovery lists traveller profiles with distance, travel interest and a one-tap message button, and provides destination search and filters.
- Conversations lists private and group chats with the latest message.
- Map shows nearby travellers within the chosen radius, with controls for search radius, zoom and a global view.
- Create account collects name, e-mail, bio, primary interest, travel style and password.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |




*(a) Home / discovery* *(b) Conversations*

*(c) Nearby-traveller map* *(d) Create account*

Figure 4.1 Application screens: (a) home, (b) chat, (c) map, (d) account creation

The trip-management screens let users create a trip with destination, dates and notes, build a day-wise itinerary with checklists, see the route and attraction pins on the map, and invite others to view or edit the plan.

### *F. Data and Privacy*

Explora stores two kinds of user-generated data: **text** (travel preferences, destinations, bios, reviews and itineraries) and **images** (profile pictures and optional travel photos). Text fields are standardised and validated (blank, duplicate and inappropriate entries are rejected), and travel categories are tagged with predefined keywords to support efficient filtering.

Privacy is treated as a design requirement. All data is stored in Firebase with access-control rules; users are informed about data use at sign-up and control what others can see; personally identifiable information is kept out of public reviews where possible; and location sharing is opt-in. These choices are intended to be consistent with the principles of the EU General Data Protection Regulation [5]; a formal compliance assessment was outside the scope of this project.

## V. TESTING AND EVALUATION

### *A. Objectives and Strategy*

Testing aimed to validate the correctness and completeness of the system, ensure secure and smooth interaction across devices, and detect and fix defects before release. A combination of manual and automated techniques was applied in the following phases:

- Unit testing – individual components such as login, profile update and search filters were verified in isolation.
- Integration testing – interactions between profile, matching and chat, and with Firebase and the map module, were validated.
- System testing – end-to-end testing on Android devices and emulators for functionality, responsiveness and screen-size compatibility.
- Regression testing – previously validated components were re-tested after bug fixes and new features.
- User acceptance testing (UAT) – conducted with beta users (Section 5.4).

Firebase Test Lab and the Android Studio Profiler were used for cross-device testing and for analysing performance and memory use.

### *B. Test Cases*

Table 5.1 Sample functional test cases

| ID | Description | Expected result | Status |
|---|---|---|---|
| TC01 | Login with valid credentials | Dashboard is displayed | Passed |
| TC02 | Profile creation with missing fields | Error message is shown | Passed |
| TC03 | Matchmaking based on a common destination | Compatible matches are shown | Passed |
| TC04 | Real-time chat with a matched user | Messages are sent and received instantly | Passed |

### *C. Results*

All executed functional test cases listed above passed. Minor issues found during testing, including UI responsiveness on small screens and glitches in profile update, were fixed, and the final build behaved consistently across the tested devices.

### *D. User Acceptance Testing*

UAT was carried out with a small group of beta users, including solo travellers, students and frequent backpackers. Participants completed common flows (signing up, setting travel preferences, using the match feature and starting a conversation), and feedback was collected through surveys and interviews.

- Feedback: users found the interface intuitive, the suggested matches relevant, and the chat smooth and responsive; they valued the safety features.
- Improvements made: the filter options were refined for more accurate search, and cropping was added for profile photos.

## VI. CONCLUSION, LIMITATIONS AND FUTURE WORK

### *A. Conclusion*

Explora was conceived to solve a real problem: travellers lose trips, money and safety when companions cancel or cannot be found. The project delivered a cross-platform mobile application that:

- connects travellers with compatible companions using explainable, preference- and proximity-based matching;
- supports safe interaction through authentication, ratings, reviews and reporting;
- lowers the cost of travel through built-in expense sharing; and
- supports collaborative planning through shared itineraries, real-time chat and an interactive map.

Testing and user feedback indicate that the application meets its functional goals and is usable in practice.

### *B. Limitations*

- Matching is rule-based; it cannot learn from user behaviour and treats all criteria equally.
- Evaluation was limited to functional testing and a small UAT group; no large-scale load or field study was carried out.
- Identity verification is basic; stronger verification would be needed before a public launch.
- Payments, event suggestions, calendar export and voice/video calling are planned but not yet implemented.

### *C. Future Work*

- Learned recommendation: collaborative-filtering or embedding-based matching trained on match acceptance, chat and review data [3], [4], with weights learned per user instead of fixed rules.
- Trust and safety: government-ID verification, reputation scores, and automated moderation of reviews and chat.
- Scalability studies: formal load testing against the 500-user target.
- Features: payments, event and activity suggestions, calendar integration, offline maps, multilingual support and an iOS release.
- Evaluation: a larger user study with quantitative usability measures such as the System Usability Scale.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

## REFERENCES

[1] Borràs, J., Moreno, A., & Valls, A. (2014). Intelligent tourism recommender systems: A survey. Expert Systems with Applications, 41(16), 7370–7389.
[2] Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. Decision Support Systems, 74, 12–32.
[3] Ricci, F., Rokach, L., & Shapira, B. (Eds.). (2022). Recommender Systems Handbook (3rd ed.). Springer.
[4] Resnick, P., & Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40(3), 56–58.
[5] European Parliament and Council. (2016). Regulation (EU) 2016/679 (General Data Protection Regulation). Official Journal of the European Union, L 119, 1–88.
[6] Google. Firebase documentation. https://firebase.google.com/docs
[7] Google. Flutter documentation. https://docs.flutter.dev
[8] Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.
[9] Nielsen, J. (1994). Usability Engineering. Morgan Kaufmann.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue XI Nov 2026- Available at www.ijraset.com*

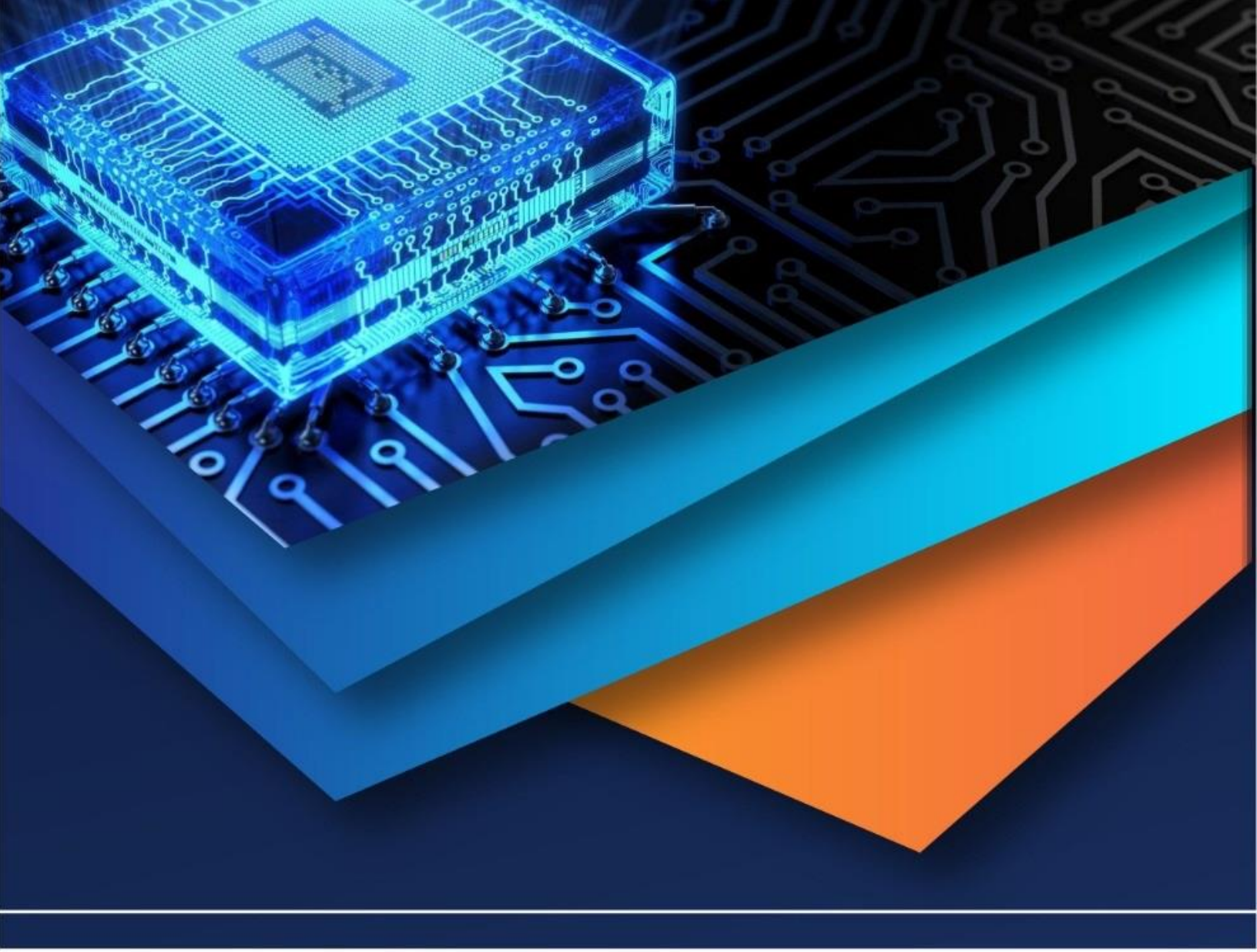
# Appendix A Firebase Setup Summary

1. Create a Firebase project in the Firebase console.
2. Register the Android app with its package name.
3. Download google-services.json and place it in the project.
4. Enable Authentication, Cloud Firestore and Cloud Storage.
5. Add the Firebase SDK dependencies to the build configuration.

# Appendix B Glossary

| Term | Definition |
|---|---|
| Matchmaking | Connecting users on the basis of shared travel interests and trip details. |
| Firebase | Google's cloud platform providing authentication, database, storage and messaging. |
| Flutter | Google's open-source framework for building cross-platform apps from one Dart code base. |
| UAT | User acceptance testing: confirms that the software meets user needs. |
| Itinerary | A planned route or journey with dates, places and activities. |
| GPS | Global Positioning System, used for location services. |

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)