

TRIP VAULT

SUBMITTED BY

KARTHIKEYAN R (24MX111)

MOHAMED ALI AKRAM (24MX114)

23MX27 Mobile Application Development

REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF COMPUTER APPLICATION

ANNA UNIVERSITY



MAY 2025

DEPARTMENT OF COMPUTER APPLICATIONS

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE - 641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE - 641 004

TRIP VAULT

Bonafide record of work done by

KARTHIKEYAN R (24MX111)

MOHAMED ALI AKRAM (24MX114)

TRIP VAULT

REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF COMPUTER APPLICATION

ANNA UNIVERSITY

MAY 2025

FACULTY GUIDE

.....

Ms.K.GAYATHRI

ACKNOWLEDGEMENT

We immensely take this opportunity to express our sincere gratitude to **Dr.K.Prakasan**, Principal, PSG College of Technology, for providing us all the facilities within the campus for the completion of the project.

We profoundly thank Dr.A.Chitra, Professor and Head, Department of Computer Applications, PSG College of Technology, for her moral support and guidance.

We owe an extremely unbound gratitude and extend our thanks to our Programme Coordinator, **Dr.R.Manavalan**, Associate Professor, Department of Computer Applications, PSG College of Technology, whose motivation and support encouraged us in taking up and completing this project work.

We are overwhelmed in all humbleness and gratefulness in acknowledging our subject Coordinator and Project Guide **Ms.K.Gayathri**, Assistant Professor (Sr. Gr.), Department of Computer Applications, PSG College of Technology, for her priceless suggestions and unrelenting support in all our efforts to improve our project and for piloting the right way for the successful completion of our project.

We are grateful to our project review panel members **Ms.J.Gowri Thangam**, Assistant Professor (Sr. Gr.), **Ms.K.Gayathri**, Assistant Professor (Sr. Gr.). Their insights, expertise, and constructive feedback were incredibly valuable to us. They provided a fresh perspective and have inspired us to further refine and improve our project. We deeply express our gratitude for the time and effort they put in our project.

We also express our sincere thanks to all the staff members of the Department of Computer Applications for their encouragement. We also thank our parents and all the hands that helped us.

SYNOPSIS

Trip Vault is an innovative mobile application designed to elevate travel experiences by providing users with powerful tools to explore, bookmark, and document their journeys in a highly personalized manner. The app enables users to perform city-based searches to discover top-rated tourist destinations, complete with detailed information, visual content, and navigation support. By leveraging real-time geolocation features through the Nominatim API and integrating tourist spot data via the OpenTripMap API, Trip Vault ensures that travelers have access to accurate and up-to-date location-based content. Users can bookmark their favorite places into customizable categories for easy organization and future reference. In addition to place discovery, Trip Vault also acts as a digital travel journal, allowing users to store memories in multiple formats, including photographs, personal notes, audio clips, and star-based ratings. The application features seamless map integration using OSMDroid, enabling interactive and offline-friendly map experiences. Behind the scenes, Firebase Authentication is used to manage user login and registration securely, while Firestore and Firebase Storage handle efficient and scalable data management for user-generated content. Built entirely using Kotlin and Jetpack Compose, Trip Vault follows the MVVM (Model-View-ViewModel) architecture, ensuring a clean separation of concerns and smooth user interactions. This combination of modern technologies and thoughtful features makes Trip Vault an ideal companion for both casual tourists and seasoned travelers who wish to keep their adventures well-organized and memorable.

TABLE OF CONTENTS

No	TITLE	PAGE NO
1	INTRODUCTION	1
	1.1 PROJECT OVERVIEW	1
	1.2 TECHNOLOGY OVERVIEW	1
	1.3 ARCHITECTURE	4
2	SYSTEM ANALYSIS	5
	2.1 EXISTING SYSTEM	5
	2.2 PROPOSED SYSTEM	5
	2.3 FUNCTIONAL REQUIREMENT	6
	2.4 NON-FUNCTIONAL REQUIREMENT	6
3	SYSTEM DESIGN	8
	3.1 SYSTEM FLOW DIAGRAM	8
	3.2 DATABASE SCHEMA	11
	3.3 USE CASE OF THE APPLICATION	13
4	SYSTEM IMPLEMENTATION	17
	4.1 AUTHENTICATION MODULE	17
	4.2 LOCATION DISCOVER MODULE	17
	4.3 BOOKMARKING MODULE	19
	4.4 TRAVEL MEMORY MODULE	20
5	SYSTEM TESTING	22
	5.1 TESTING TECHNIQUES	22
	5.2 TEST CASE REPORT	23
6	CONCLUSION AND FUTURE ENHANCEMENTS	26
7	BIBLIOGRAPHY	27

CHAPTER 1

INTRODUCTION

1.1 Project Overview

Trip Vault is an Android-based mobile application designed to support travelers in discovering tourist destinations, bookmarking places of interest, and documenting their experiences. Users can search for attractions in any city, explore categorized listings such as landmarks, parks, or restaurants, and access detailed information for each location.

The app also facilitates the creation of digital travel diaries, where users can capture and save notes, photos, voice memos, and ratings for each place they visit. The goal is to enhance the travel experience by combining exploration, organization, and memory preservation in one unified application.

1.2 Technology Overview

- **Frontend:** Kotlin with Jetpack Compose for declarative UI design
- **UI Components:** Material Design for consistent and responsive UI elements
- **Backend Services:**
 - Firebase Authentication for secure login via email or Google
 - Firestore for cloud database management
 - Firebase Storage for handling images and media

- **APIs Integrated:**
 - Nominatim API for geolocation and city search
 - OpenTripMap API for retrieving place details
- **Maps Integration:** OSMDroid for displaying interactive maps
- **Development Tools:** Android Studio for coding and testing; Postman for API validation
- **Jetpack Compose (UI Framework):** Jetpack Compose is a modern declarative UI toolkit used in Trip Vault for building dynamic and responsive Android interfaces. It simplifies UI development by enabling UI definitions in Kotlin code and supports features like recomposition, theming, animations, and real-time previews. The result is a smoother and more efficient development process for creating intuitive user experiences.
- **Firestore Authentication:** Trip Vault uses Firebase Authentication to enable secure login functionality for users via email and Google sign-in. It simplifies the process of identity verification and manages sessions securely, ensuring only authorized users access personal travel data and bookmarks.
- **Firestore (Cloud Database):** Firestore serves as the real-time NoSQL cloud database for storing user-related data. This includes user profiles, bookmarks, categorized locations, and multimedia memories. Its document-based schema allows flexible data modeling that evolves with the app's functionality.
- **Firestore Storage:** All images, audio files, and multimedia memories are stored securely in Firebase Storage. It supports seamless upload/download capabilities, version control, and robust

file access rules to ensure secure and scalable media management.

- **Nominatim API (OpenStreetMap):** Nominatim API is used to convert user-input city names into geographic coordinates (latitude and longitude). This geocoding capability enables the app to focus searches on specific locations and improves map navigation.
- **OpenTripMap API:** OpenTripMap provides real-time access to detailed information about popular tourist destinations. It returns place details such as category, name, coordinates, and open hours, which are displayed in the app's discovery and details screens.
- **OSMDroid (Mapping Library):** Trip Vault integrates OSMDroid to render interactive maps and display markers for searched and bookmarked locations. This open-source mapping library enables zooming, panning, and layer customization, offering a robust and offline-capable mapping solution.
- **Kotlin Coroutines:** Asynchronous tasks, such as fetching API data or uploading images, are managed using Kotlin Coroutines. This ensures smooth user interaction without freezing the UI thread and supports concurrent operations in a structured and readable manner.
- **Android Studio & Postman (Development Tools):** Android Studio serves as the main development IDE, offering debugging, testing, and emulation capabilities. Postman is used to test external APIs (Nominatim and OpenTripMap) before integration, ensuring reliable communication and data accuracy.

1.3 Architecture

Trip Vault follows the MVVM (Model-View-ViewModel) architecture pattern. It separates concerns clearly into:

- **Model:** Represents the data layer interacting with Firebase and external APIs.
- **ViewModel:** Handles logic, API calls, and prepares data for the UI.
- **View (UI):** Composable functions that observe data and reflect changes dynamically.

This architecture improves testability, maintainability, and ensures reactive data binding between the UI and backend logic using LiveData and State.

CHAPTER 2

SYSTEM ANALYSIS

2.1 Existing System

Existing solutions for travel planning often require switching between multiple apps for navigation, bookmarking, and memory storage. Popular mapping or booking platforms provide basic place details but lack personalized features like multimedia journaling or custom bookmarking. This leads to a fragmented and less engaging travel experience.

Disadvantages of the Existing System

Existing travel planning solutions often create a fragmented user experience, as users must switch between multiple applications for navigation, bookmarking, and memory storage. Most platforms provide only basic place information and lack features for personalized suggestions or multimedia journaling. Additionally, they do not offer options to organize bookmarks or store rich content like photos, notes, or audio. This lack of integration and personalization makes travel planning less efficient, less engaging, and more time-consuming for users.

2.2 Proposed System

The proposed Trip Vault system addresses these gaps by offering a single application for place discovery, real-time navigation, bookmarking, and personal memory storage. By leveraging reliable APIs and Firebase services, the system reduces dependency on multiple tools, thus delivering a seamless travel assistant.

Advantages of the Proposed System

The proposed Trip Vault system offers several advantages over existing solutions by integrating all essential travel features into a single application. It enables users to discover places, navigate in real-time, bookmark locations, and store personal memories like photos and notes—all in one platform.

2.3 Functional Requirements

- **User Authentication:**

TripVault supports secure login and registration using email or Google accounts, enabling personalized access to user data.

- **City Search & Place Retrieval:**

Users can search any city to instantly view its top tourist spots, making trip planning fast and easy

- **Category Filters:**

Travelers can filter results by categories like parks, restaurants, or museums to find places that match their interests

- **Place Details View:**

Each location shows essential details such as name, address, category, and open hours for easy decision-making

- **Bookmark Management:**

Users can save, edit, and organize favorite locations into categories for efficient trip planning

- **Multimedia Memories:**

TripVault allows users to enrich bookmarks with photos, notes, and ratings, creating personalized, shareable travel memories.

- **Map Integration:**

An interactive map shows both the user's location and place markers, helping with real-time navigation and exploration

2.4 Non-Functional Requirements

- **Security**

TripVault ensures all user sessions and data are securely managed using Firebase Authentication and Firestore security rules.

- **Performance**

The app delivers real-time data access and smooth UI transitions, providing a responsive and seamless user experience.

- **Scalability**

TripVault is designed to scale efficiently, supporting growing numbers of users and their expanding travel data without performance loss.

- **Availability**

With cloud-based storage, users can access their travel logs and bookmarks anytime, from any device with internet access.

- **Usability**

Built using Jetpack Compose, the app offers a clean, user-friendly, and responsive interface for effortless navigation.

- **Compatibility**

TripVault supports Android 8 and above, featuring adaptive layouts to ensure consistent performance across various devices.

CHAPTER 3

SYSTEM DESIGN

This Chapter discusses system design. System design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements.

The design activities are of main importance in this phase, because in these activities decisions ultimately affect the success of the software implementation and its ease of maintenance is made. This decision has the final bearing on the reliability and maintainability of a system. Design is the only way to accurately transfer the requirements into finished software or system.

3.1 Flow Diagram

A system flow diagram, also known as a flowchart, is a graphical representation of the logical steps or flow of a process or algorithm in software engineering. It is a visual tool used to understand, analyze, and communicate the sequence of activities or decisions involved in a software system.

The system flow diagram for Trip Vault illustrates the main processes and workflows within the application:

User Authentication Flow:

1. User opens the application
2. System checks if user is logged in a. If not logged in, display login/registration screen b. User enters credentials or registers new account c. System validates credentials with Firebase Authentication d. On successful authentication, proceed to home screen e. On failure, display appropriate error message
3. If already logged in, proceed directly to home screen

Location Search Flow:

1. User enters city or location name in search bar
2. System sends request to Nominatim API to convert location name to coordinates
3. System queries OpenTripMap API with coordinates to retrieve tourist attractions

4. Results are displayed to user categorized by type (restaurants, parks, etc.)
5. User can filter results by selecting specific categories
6. User can select a location to view details

Bookmarking Flow:

1. User views location details
2. User taps bookmark icon
3. System prompts user to select category or create new category
4. System saves bookmark to Firebase Firestore
5. Confirmation message appears
6. Bookmark appears in user's saved lists

Travel Memory Capture Flow:

1. User selects a location (either from search results or bookmarks)
2. User accesses the memory capture interface
3. User can: a. Take photos using device camera b. Record audio notes c. Write text notes d. Assign ratings
4. User saves the memory
5. System uploads media to Firebase Storage
6. System saves reference and metadata to Firebase Firestore
7. Memory is associated with the location and accessible from location details

Navigation Flow:

1. User selects a location
2. User taps navigation button
3. System displays location on interactive map
4. User can view directions and navigate to the destination

The system flow diagram helps visualize how different components of Trip Vault interact, ensuring a logical and efficient user experience throughout the application.

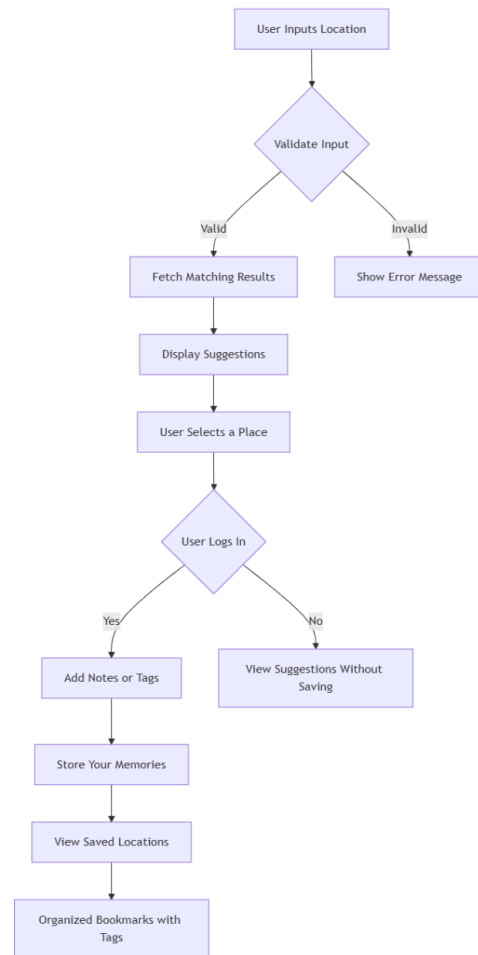


Figure 3.1 System Flow Diagram of TripVault

3.2 Use Case Diagram :

A use case diagram is a visual representation in software engineering that depicts the interactions between users (actors) and a system to illustrate the functionality and behavior of the system from a user's perspective. It is a powerful tool for capturing and documenting the requirements of a system and understanding the various use cases or scenarios in which the system will be utilized.

The use case diagram for Trip Vault illustrates the key functionalities available to users and the system's interaction with external services. The primary actor is the Traveler (User) who interacts with the system through various use cases.

Primary Use Cases for Trip Vault:

1. **Register/Login** - User can create an account or log in to existing account
2. **Search Locations** - User can search for tourist attractions by city name
3. **Filter by Category** - User can refine search results by categories
4. **View Location Details** - User can see detailed information about places
5. **Bookmark Location** - User can save favorite places
6. **Categorize Bookmarks** - User can organize bookmarks into custom categories
7. **Capture Travel Memories** - User can add photos, notes, and audio to locations
8. **Rate Visited Places** - User can assign ratings to visited locations
9. **View Maps** - User can see locations on interactive maps

External Actors:

1. **Firebase Authentication** - Handles user authentication

2. **Firebase Firestore** - Manages data storage and retrieval
3. **Firebase Storage** - Handles media file storage
4. **Nominatim API** - Provides geolocation services
5. **OpenTripMap API** - Supplies tourist location data

The relationships between these actors and use cases show the comprehensive functionality of the Trip Vault application, highlighting how the system integrates discovery, navigation, and memory capture in a unified platform. which is shown in Figure 3.2 as Use Case Diagram

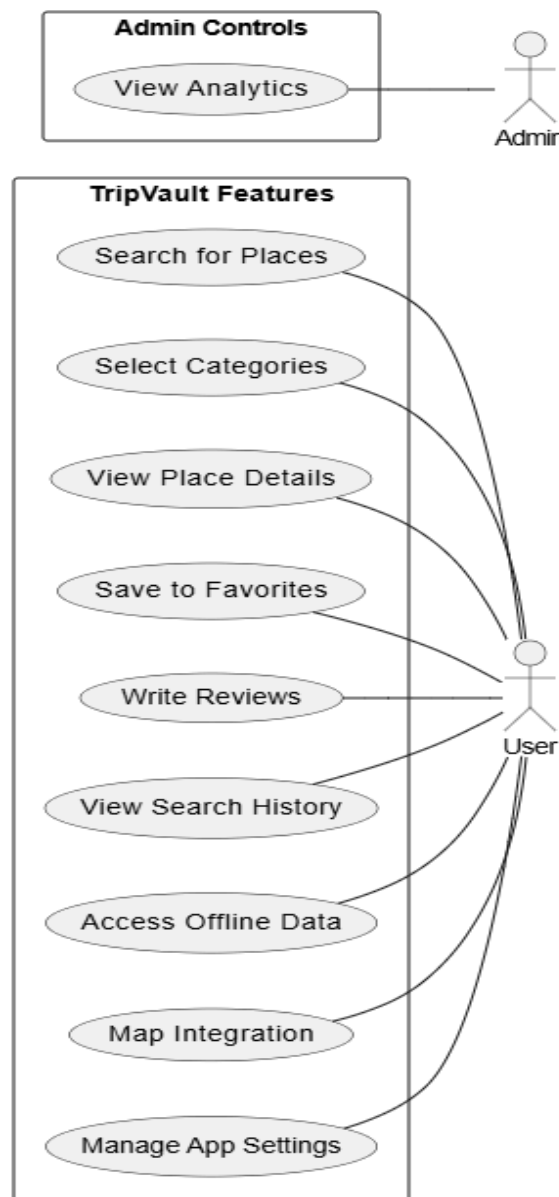


Figure 3.2 Use Case Diagram of TripVault

3.3 Database Schema

Trip Vault uses Firebase Firestore, a NoSQL document database, to store and manage application data. The database schema is designed to efficiently support the application's functionalities while maintaining flexibility for future enhancements.

Users Collection:

```
users {  
  userId: String, // Firebase Authentication UID  
  displayName: String,  
  email: String,  
  photoURL: String, // Profile picture URL  
  createdAt: Timestamp,  
  lastLogin: Timestamp  
}
```

Bookmarks Collection:

```
bookmarks {  
  bookmarkId: String,  
  userId: String, // Reference to user who created bookmark  
  placeId: String, // Unique identifier from OpenTripMap  
  placeName: String,  
  address: String,  
  category: String,  
  coordinates: {  
    latitude: Double,  
    longitude: Double  
  },  
  createdAt: Timestamp,  
  notes: String // Optional user notes  
}
```

Categories Collection:

```
categories {
  categoryId: String,
  userId: String, // Reference to user who created category
  name: String,
  description: String,
  iconName: String,
  color: String,
  createdAt: Timestamp
}
```

Memories Collection:

```
memories {
  memoryId: String,
  userId: String, // Reference to user who created memory
  placeId: String, // Reference to place
  title: String,
  notes: String,
  rating: Number, // 1-5 scale
  visitDate: Timestamp,
  createdAt: Timestamp,
  updatedAt: Timestamp,
  mediaItems: [
    {
      type: String, // "photo" or "audio"
      url: String, // Storage reference URL
      thumbnailUrl: String, // For photos
      duration: Number, // For audio recordings
      createdAt: Timestamp
    }
  ]
}
```

RecentSearches Collection:

```
recentSearches {
  searchId: String,
  userId: String,
  query: String,
  coordinates: {
    latitude: Double,
    longitude: Double
  },
  timestamp: Timestamp
}
```

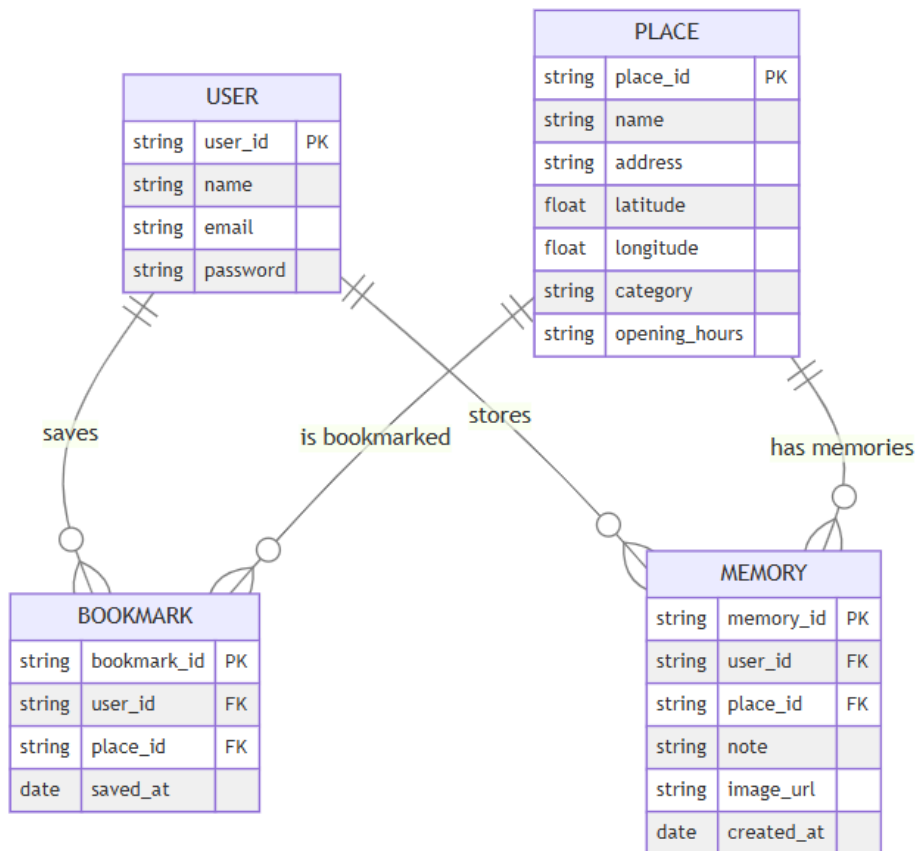


Figure 3.3 Database Schema Diagram of TripVault

This database schema supports the core functionalities of Trip Vault while maintaining good performance characteristics. The structure allows for efficient querying of user-specific data while maintaining relationships between different entities. Firebase Firestore's hierarchical document model is particularly well-suited for this application, as it allows for flexible schema evolution as the application grows and new features are added.

CHAPTER 4

SYSTEM IMPLEMENTATION

This chapter discusses the system implementation. System implementation is a set of procedures performed to complete the design contained in the systems design document.

4.1 Authentication Module

The Authentication Module in Trip Vault provides secure user registration and login functionality. This module ensures that only authorized users can access personalized features such as bookmarking locations and storing travel memories. The implementation uses Firebase Authentication to handle user credentials securely and efficiently.

The module offers two authentication methods:

- Email and password authentication
- Google sign-in integration
 - Initial loading screen of TripVault is shown in Figure 4.1
 - User Registration screen of TripVault is shown in Figure 4.2

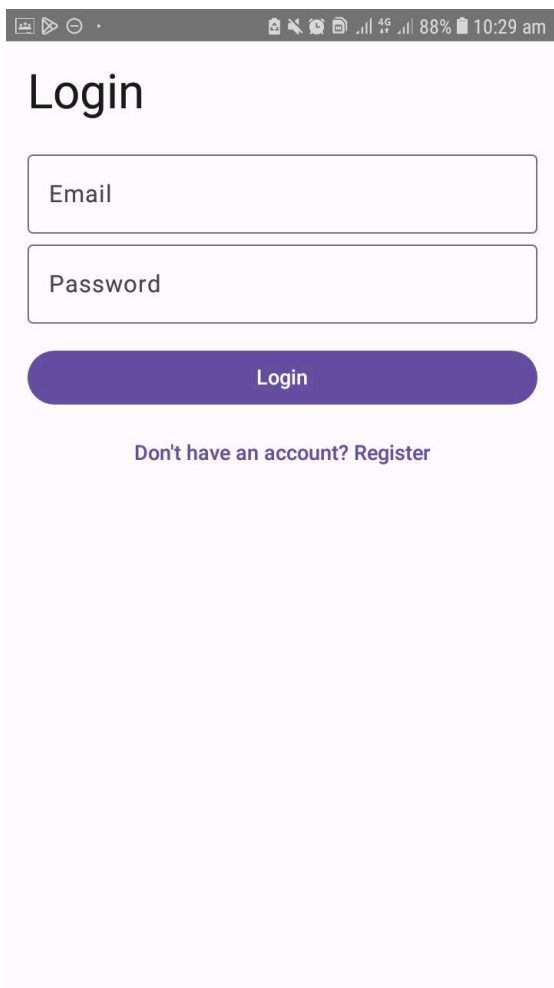
When a user opens the application for the first time, they are presented with options to either sign up for a new account or log in with existing credentials. The authentication process is designed to be seamless and user-friendly, with clear error messages and validation

4.2 Location Discovery Module

The Location Discovery Module is the core functionality of Trip Vault, allowing users to search for and discover tourist attractions based on location and preferences. This module integrates with external APIs to provide accurate and up-to-date information about points of interest. The module uses the Nominatim API to convert user-provided city names into geographic coordinates (latitude and longitude). Once the coordinates are obtained, the OpenTripMap API is queried to retrieve a list of nearby tourist attractions.

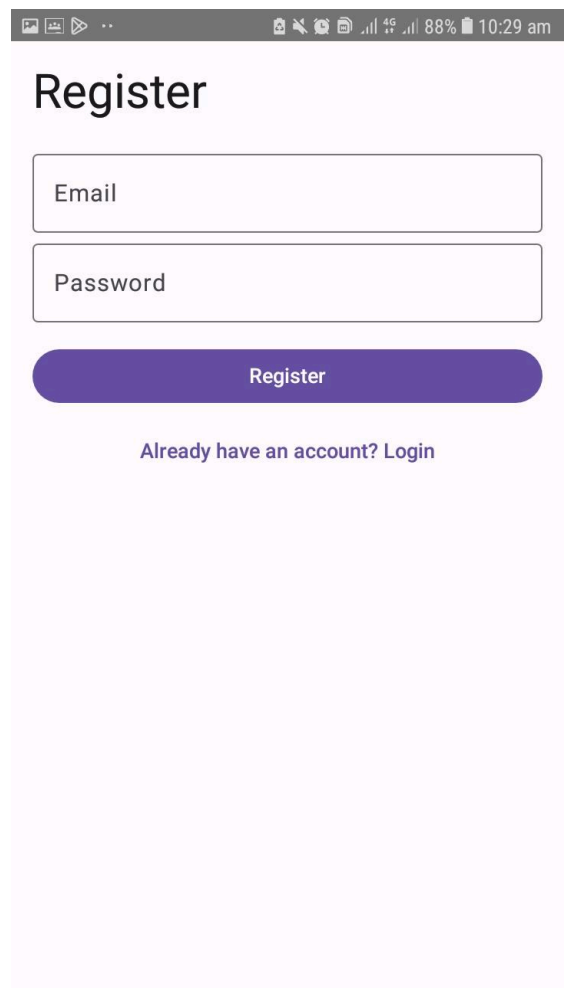
Users can further refine their search results by applying category filters such as:

- Historical sites
- Natural attractions
- Museums and galleries
- Restaurants and cafes
- Parks and recreation areas
- Religious sites



The image shows a mobile application interface for logging in. At the top, there is a status bar with various icons and the time 10:29 am. Below the status bar, the word "Login" is displayed in a large, bold, black font. Underneath, there are two input fields: "Email" and "Password", both with light purple borders. Below these fields is a large, rounded purple button with the word "Login" in white text. At the bottom, there is a link that says "Don't have an account? Register" in a smaller, purple font.

Figure 4.1 shows the TripVault splash screen.



The image shows a mobile application interface for registering. At the top, there is a status bar with various icons and the time 10:29 am. Below the status bar, the word "Register" is displayed in a large, bold, black font. Underneath, there are two input fields: "Email" and "Password", both with light purple borders. Below these fields is a large, rounded purple button with the word "Register" in white text. At the bottom, there is a link that says "Already have an account? Login" in a smaller, purple font.

Figure 4.2 shows the user registration page.

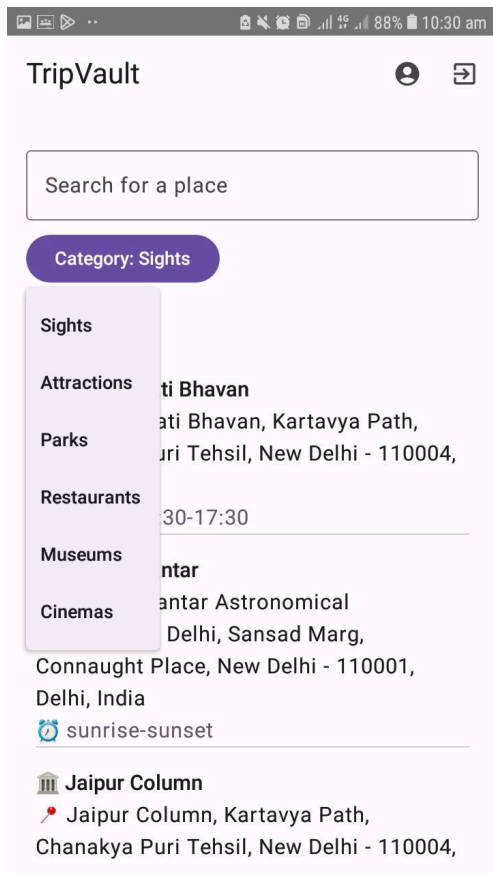


Figure 4.3 shows location search by category.

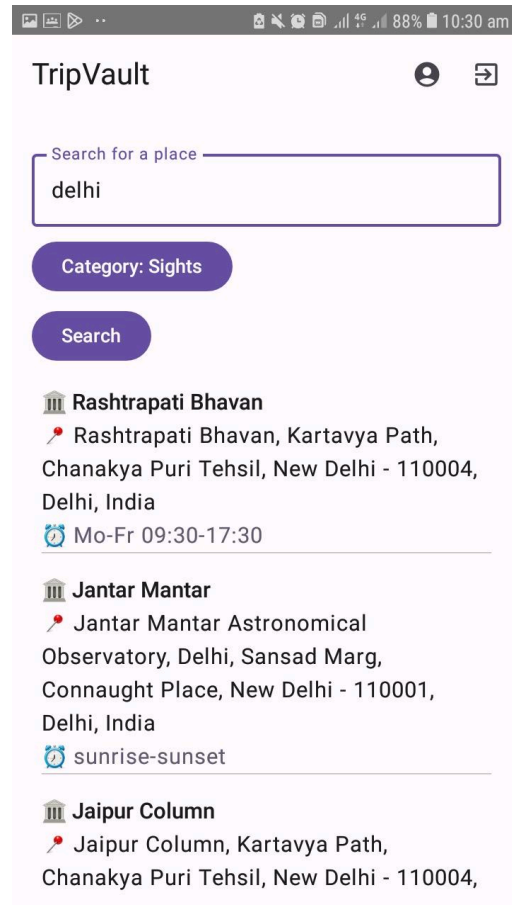


Figure 4.4 shows dynamic search results.

The search results are displayed in a visually appealing list format, showing key information such as the name, category, and a representative image for each location.

User can search for locations based on specific categories for a more personalized experience, as shown in Figure 4.3.

Matching results are dynamically fetched and displayed once a user enters a location or category, as shown in Figure 4.4.

4.3 Bookmarking Module

The Bookmarking Module enables users to save and organize their favorite locations within the app for future reference. This feature enhances the overall travel experience by helping users easily plan itineraries, revisit saved spots, and keep track of places they are interested in exploring during their trips.

The implementation includes:

- Adding locations to bookmarks with a single tap
- Organizing bookmarks into custom categories
- Viewing all bookmarked locations in a user's profile
- Removing locations from bookmarks when no longer needed

The bookmarking functionality is fully integrated with Firebase Firestore, ensuring that user data is synchronized across devices and persistently stored in the cloud.

Users can review selected locations, bookmark favorites, and upload memories such as photos or notes for future reference, as shown in Figure 4.5.

The selected location is accurately displayed on the map for easy viewing, as shown in Figure 4.6.

4.4 Travel Memory Module

The Travel Memory Module allows users to document their travel experiences by creating rich, multimedia memories associated with specific locations. This feature transforms Trip Vault from a simple discovery tool into a personal travel journal.

Users can create memories that include:

- Photos captured at the location
- Written notes and descriptions
- Audio recordings of their experiences
- Ratings and personal reviews

These memories are stored securely using Firebase Storage for media files and Firebase Firestore for the associated metadata. The implementation ensures efficient uploading and retrieval of media content, even in varying network conditions.

All saved data, including bookmarked locations and uploaded memories, are stored under the user's profile along with their personal details, as shown in Figure 4.7

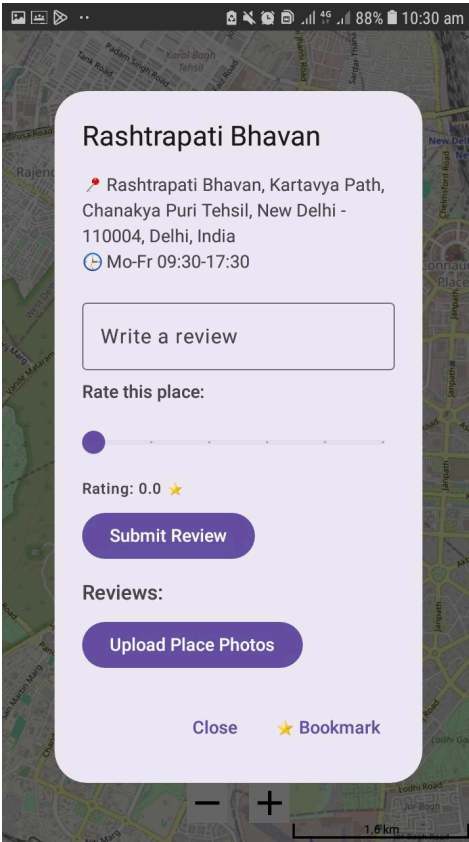


Figure 4.5 shows bookmarking and memory upload.

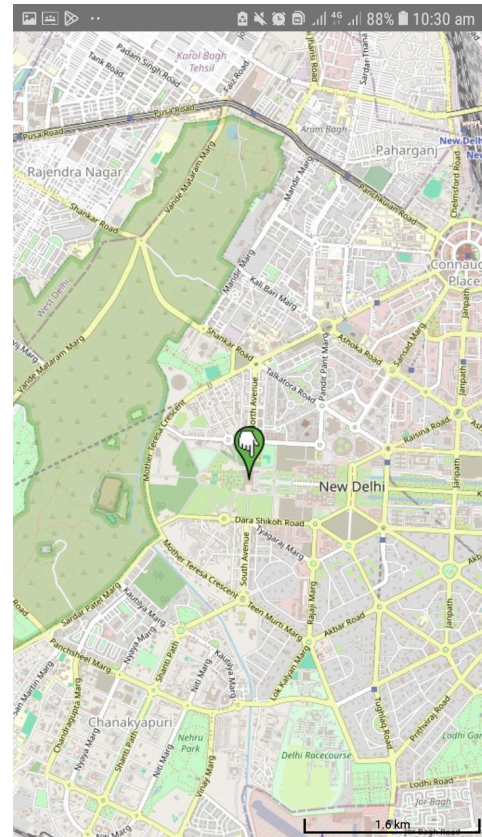


Figure 4.6 shows map with selected location.

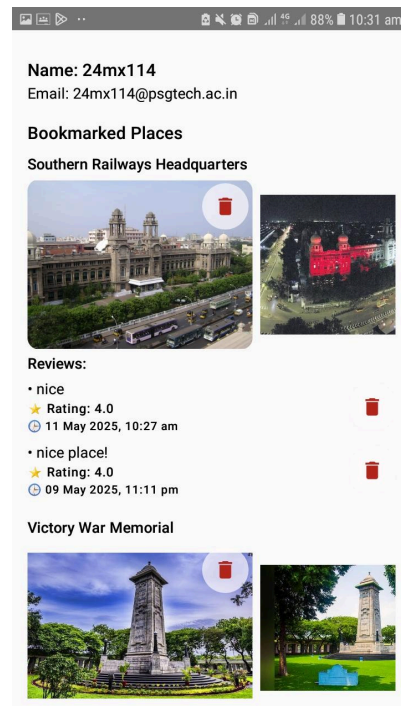


Figure 4.7 shows the user profile with saved data

CHAPTER 5

SYSTEM TESTING

This chapter delves into the critical phase of system testing, elucidating diverse testing methodologies employed throughout the application's development lifecycle. Rigorous testing was conducted at various stages to detect and rectify bugs promptly, ensuring the delivery of a flawless end product. The primary goal was to validate that the expected outcomes align seamlessly with the defined inputs.

5.1 Testing Techniques

The testing process intricately examines the software's logical internals, affirming the completeness of code execution and scrutinizing functional aspects. It guarantees that specified inputs yield results consistent with the anticipated outcomes. Testing is integral to the development cycle, its extent contingent upon the application's size and complexity. This chapter elucidates the diverse testing strategies embraced in this project.

Unit Testing

Unit testing is meticulous, focusing on individual modules within the source code. Each module undergoes rigorous scrutiny to ensure correctness, validity, and compliance with objectives. Errors, if detected, are swiftly addressed, enhancing program clarity. In Trip Vault, the system is compartmentalized into modules, each of which undergoes exhaustive unit testing, streamlining error detection and rectification. Each unit namely Authentication module, Location Discovery module, Bookmarking module, and Travel Memory module were tested independently.

Integration Testing

Integration testing scrutinizes the synergy among different modules, services utilized by the application. TripVault includes validating interactions with external APIs like Nominatim and OpenTripMap, as well as ensuring seamless collaboration between the app and Firebase services. These tests, albeit more resource-intensive, are crucial as they necessitate the simultaneous operation of multiple application components. Integration testing was performed after successfully integrating Firebase Authentication, Firestore database, external APIs, and the OSMDroid mapping library with the application.

Functional Testing

Functional tests zero in on the application's business requirements, concentrating solely on the output of specific actions. Unlike integration tests, functional tests do not delve into intermediate system states but validate the direct results of actions. In Trip Vault, functional testing verified that user searches returned appropriate results, bookmarking functionality correctly saved locations, and travel memories were properly created and stored.

User Interface Testing

GUI Testing is a software testing type that checks the Graphical User Interface of the Software. For Trip Vault, the following aspects were tested:

- Verification of all GUI elements for size, position, width, length, and acceptance of characters or numbers
- Checking the intended functionality of the application using the GUI
- Ensuring error messages are displayed correctly
- Verifying that fonts used in the application are readable
- Checking the alignment of text is proper
- Ensuring colors of fonts and warning messages are aesthetically pleasing
- Verifying that images are properly aligned

5.2 Test Case Report

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and code generation. Once the source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. For Trip Vault, the following test cases were defined and executed:

S. No	Test ID	Test Case	Description	Expected Outcome
1	TC001	User Registration	Verify that users can register with email and password.	New users should be successfully registered and stored in Firebase Authentication.
2	TC002	User Login	Ensure that registered users can log in with correct credentials.	Users with valid credentials are granted access to the application.
3	TC003	Google Sign-in	Verify that Google sign-in is working correctly.	Users can authenticate using their Google accounts.
4	TC004	City to Coordinates	Verify that city name input returns valid coordinates.	The Nominatim API correctly converts city names to geographic coordinates.
5	TC005	Fetch Tourist Attractions	Ensure that tourist attractions are fetched based on coordinates.	The OpenTripMap API returns a list of relevant tourist spots for the specified location.
6	TC006	Bookmark Locations	Verify that users can bookmark locations.	Selected locations are successfully saved to the user's bookmarks in Firestore.
7	TC007	Organize Bookmarks	Ensure that bookmarked locations can be organized into categories.	Users can create categories and assign bookmarks to them.

8	TC008	Add Travel Memories with Photos	Verify that users can create travel memories with photos.	Photos are uploaded to Firebase Storage and linked to the location.
9	TC009	Display Travel Memories	Ensures that travel memories are displayed correctly in user's profile	All created memories are fetched from Firestore and displayed with their associated media

CHAPTER 6

CONCLUSION

Trip Vault successfully addresses the needs of modern travelers by providing a comprehensive platform for discovering, organizing, and documenting travel experiences. By leveraging the power of Kotlin, Jetpack Compose, and Firebase, the application delivers a seamless and intuitive user experience while ensuring robust functionality and data security. The implementation of features such as location discovery, bookmarking, and travel memory creation aligns perfectly with the project's initial objectives. The application enables users to efficiently search for tourist attractions, filter results by categories, and access accurate information about points of interest. Additionally, the bookmark functionality allows for organized travel planning, while the memory creation feature transforms the app into a personal travel journal. Through rigorous testing and quality assurance, Trip Vault has demonstrated reliability and performance across various scenarios and device configurations. The adoption of the MVVM architecture ensures maintainability and scalability, setting a solid foundation for future enhancements.

FUTURE ENHANCEMENTS

Trip Vault can be significantly enhanced through several upcoming features. These include advanced location discovery using filters, historical data, user reviews, and augmented reality for immersive exploration. Social features like sharing memories, community tips, and collaborative planning will boost user engagement. Offline capabilities such as saved data access, sync-on-connect, and offline maps will improve usability. Navigation can be upgraded with turn-by-turn directions, smart itineraries, and public transport options. Personalization using machine learning for suggestions, custom plans, and smart notifications will elevate the experience, positioning Trip Vault as a smart, user-friendly travel companion.

BIBLIOGRAPHY

- [1] <https://developer.android.com/jetpack/compose>
- [2] <https://firebase.google.com/docs/auth>
- [3] <https://firebase.google.com/docs/firestore>
- [4] <https://firebase.google.com/docs/storage>
- [5] <https://nominatim.org/release-docs/latest/api/Overview/>
- [6] <https://opentripmap.io/docs>
- [7] <https://osmdroid.github.io/osmdroid/>
- [8] <https://developer.android.com/kotlin/flow>
- [9] <https://developer.android.com/topic/architecture>
- [10] <https://developer.android.com/jetpack/compose/navigation>
- [11] <https://developer.android.com/training/data-storage>
- [12] <https://developer.android.com/guide/navigation>
- [13] <https://developer.android.com/topic/libraries/architecture/viewmodel>
- [14] <https://developer.android.com/topic/libraries/architecture/livedata>
- [15] <https://developer.android.com/kotlin/coroutines>
- [16] <https://developer.android.com/reference/android/location/Location>
- [17] <https://firebase.google.com/docs/android/setup>
- [18] <https://developer.android.com/jetpack/compose/state>
- [19] <https://developer.android.com/jetpack/compose/tooling>
- [20] <https://developer.android.com/studio/debug>