**Diagram, logo

Description automatically generated**

MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY JAMSHORO

Mobile Application Development (MAD)

**Complex Engineering Problem (CEP)**

|  |  |
| --- | --- |
| Roll No: | 22SW106  &  22SW130 |
| Section: | 3 |

# RoamFeed— Project Documentation

**Project Title: RoamFeed — Local Travel Experience Sharing App**

# ABSTRACT

RoamFeed is an offline-first mobile application developed using Flutter that allows travelers to share and explore authentic experiences without relying on commercial tourism sources. Unlike mainstream platforms that highlight commercialized destinations, RoamFeed encourages users to share lesser-known travel experiences, cultural interactions, scenic viewpoints, and real stories in a community-driven environment.

The application uses Hive, a lightweight NoSQL local database, ensuring seamless offline usage, making it ideal for remote travel areas. Users can post images, descriptions, and tags and interact with others via likes and comments. The UI is responsive and optimized for multiple screen sizes. This project demonstrates strong implementation of offline data management, state handling via Provider, and user experience design in Flutter.

# REAL WORLD PROBLEM IDENTIFICATION

Modern travel planning is heavily influenced by social media and commercial tourism websites. While these platforms showcase popular destinations, they often fail to highlight cultural authenticity, local stories, and hidden travel spots. This creates a gap for travelers seeking genuine and immersive travel experiences.

Additionally, remote travel destinations such as mountain areas, deserts, and rural regions frequently lack internet connectivity. Existing application platforms become inaccessible in such environments due to their reliance on cloud-based storage and online data retrieval.

Therefore, travelers require a platform that allows the discovery and sharing of travel experiences offline, ensuring accessibility even in low-connectivity areas.

# PROPOSED SOLUTION

RoamFeed provides a community-driven environment where travelers can share real experiences through posts containing images, descriptions, tags, and timestamps. It operates fully offline and stores user-generated data locally using the Hive database. This ensures fast retrieval and smooth user experience independent of network availability.

The application’s UI is designed using Flutter’s Material Design principles, ensuring intuitive navigation, responsive layouts, and platform consistency across devices. State management is handled by Provider, which ensures efficient updates across UI components.

# Core Features

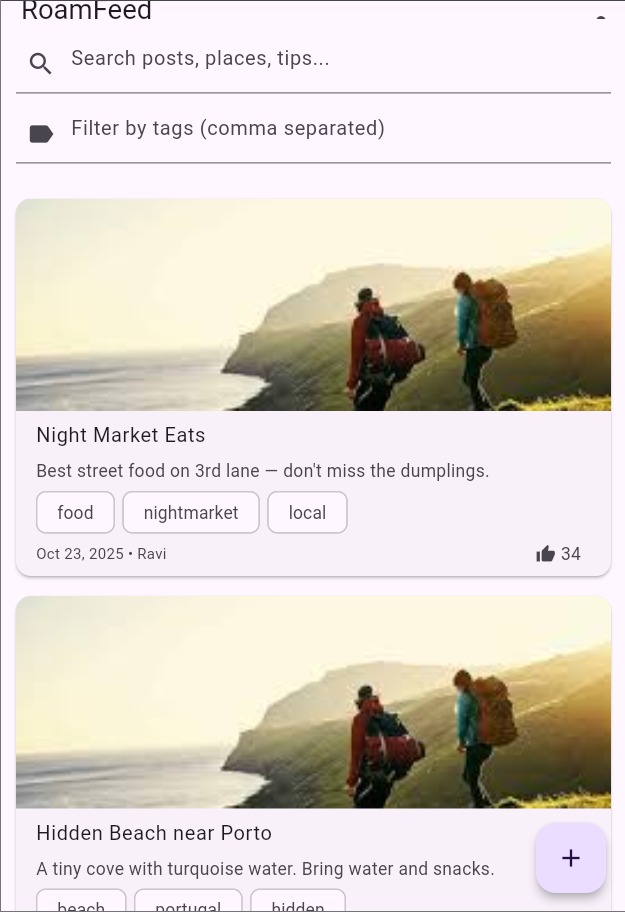
| **Feature** | **Description** |
| --- | --- |
| **Create and Share Travel Posts** | Users can create posts describing unique travel experiences, add location notes, and share personal insights. |
| **Attach Images to Posts** | Photos are stored locally as Base64 strings to ensure full offline access. |
| **Tag-Based Categorization** | Each post can include one or more tags (e.g., mountains, heritage, food) to improve organization and search. |
| **Offline Storage Using Hive** | All posts, comments, and likes are stored in local Hive boxes, enabling usage without internet. |
| **Like System for Interaction** | Users can like posts to appreciate or bookmark experiences. |
| **Comment Section for Discussions** | Visitors can ask questions, share suggestions, or interact with the traveler directly under each post. |
| **Search and Filter Posts** | Users can search by keywords or filter by tags to quickly find specific experiences. |
| **Responsive and Clean UI** | The interface adjusts fluidly for different screen sizes using Flutter's layout widgets. |

## App Flow / Working Order

| **Step** | **Screen** | **Description** |
| --- | --- | --- |
| **1** | **Home Screen** | The user opens the app and is presented with a feed of all travel posts, each showing image preview, tags, author, and like count. |
| **2** | **Add Post Screen** | The user taps the **"+"** button to create a new post, uploads an optional image, writes title & description, and adds tags. |
| **3** | **Post Saved in Hive** | The new post is stored locally in the Hive postsBox, ensuring offline access. |
| **4** | **Post Detail Screen** | When a user selects a post, full details including image, text, tags, likes and comments are displayed. |
| **5** | **Like Interaction** | The user can press the **Like** button to increase engagement. The stored post’s like count updates in Hive. |
| **6** | **Add Comment Screen / Field** | The user can add a comment to engage with other travelers' experiences. Comments are stored locally in the post object. |
| **7** | **Search & Filter Interface** | The user may search for posts by keywords or apply tag filters to view specific categories of travel experiences. |
| **8** | **Offline Usage** | All data remains available without internet, as RoamFeed operates entirely using Hive local storage. |

# 5. RESPONSIVE USER INTERFACE (SCREENSHOTS)

Figure 3.1 – Home Screen



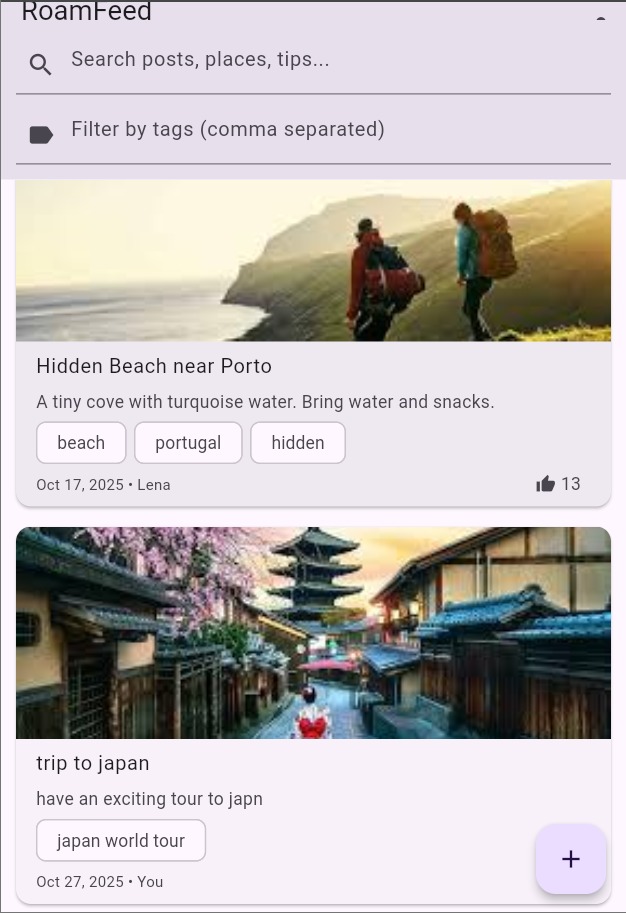


Figure 3.2 – Add Post Screen

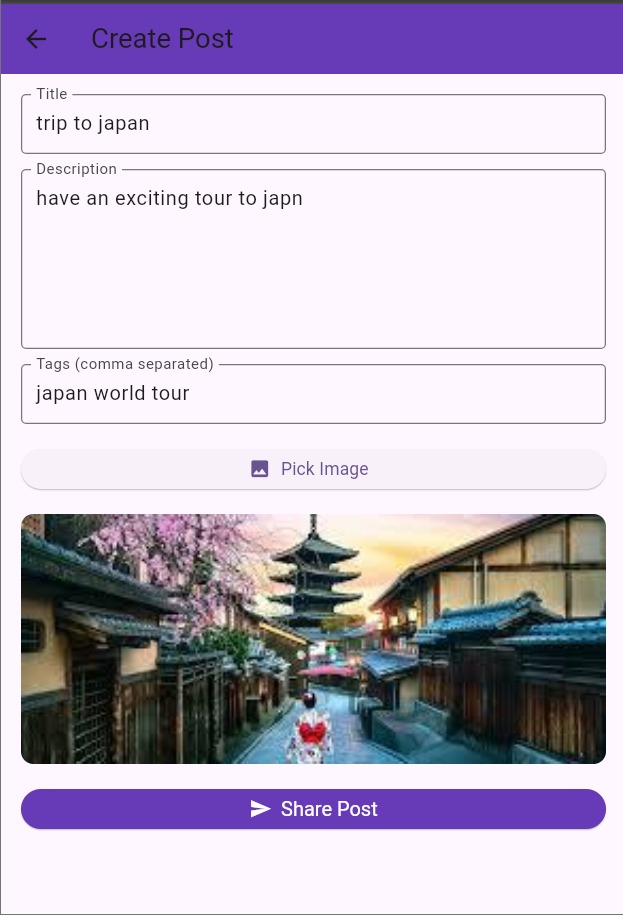


Figure 3.3 – Post Detail Screen

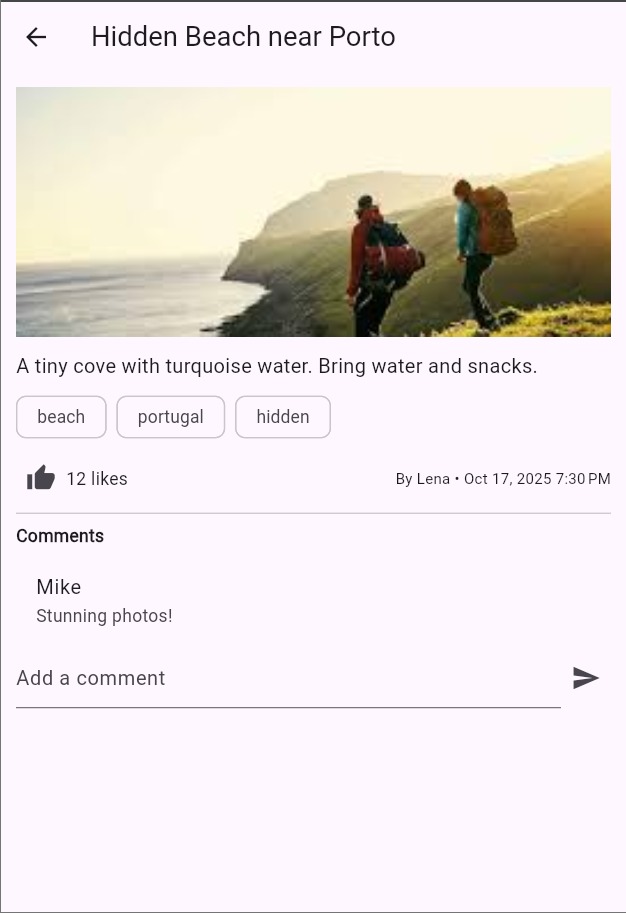


Figure 3.4 – Search & Filter Interface

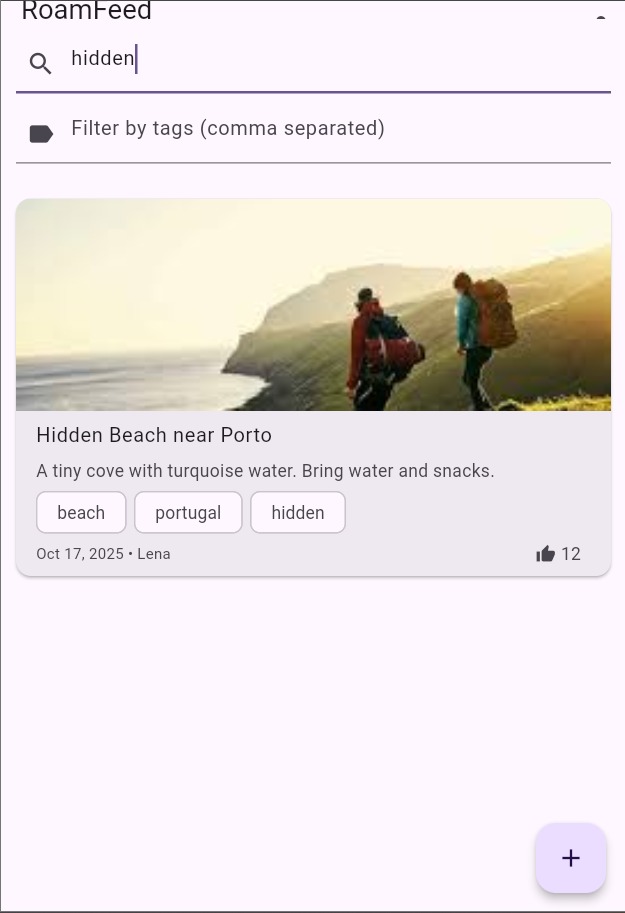
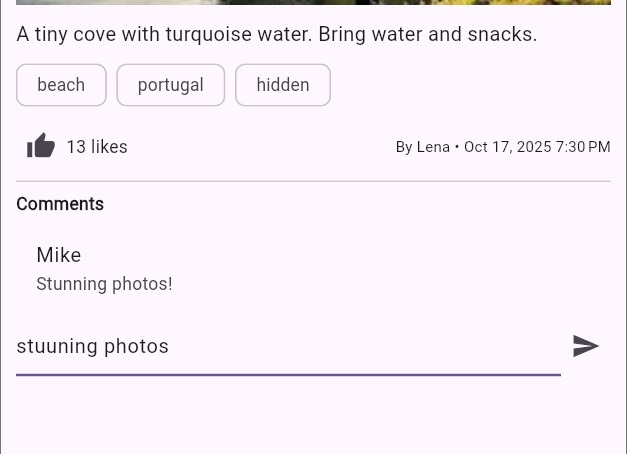


Figure 3.5 – Comments Section



# 6. DATA STORAGE (HIVE)

Hive is selected due to its lightweight nature, speed, offline capability, and Flutter integration. It allows storing structured objects locally without needing external servers.

Post Model Fields: id, title, description, imageBase64, tags, createdAt, comments, likes, author

Comment Model Fields: id, author, text, timestamp

Hive Boxes:  
• postsBox  
• metaBox

* **Hive Boxes Used in RoamFeed:**

| **Box Name** | **Model Stored** | **Purpose** |
| --- | --- | --- |
| **postsBox** | **Post** | Stores all travel posts including title, description, image, tags, likes, and comments. |
| **commentsBox** | **Comment** | Stores all comments separately if not embedded directly within Post objects. |
| **metaBox** | **Meta/Flags** | Stores metadata such as whether initial sample posts have been added or first-time setup is complete. |

* **Model Structure (Hive Objects):**

| **Model Name** | **Fields** | **Explanation** |
| --- | --- | --- |
| **Post** | id, title, description, imageBase64, tags, createdAt, likes, comments, author | Represents a full travel post shared by a user. Contains optional image stored as Base64 for offline access. |
| **Comment** | id, author, text, timestamp | Represents a user’s discussion reply within a post. Used to encourage community interaction. |

* **Why Hive Was Chosen:**

| **Reason** | **Justification** |
| --- | --- |
| **Offline Access** | Works **without internet**, perfect for remote travel areas. |
| **Fast and Lightweight** | Performs faster than SQLite for small to medium data storage. |
| **No Serialization Boilerplate** | Models are stored directly using Hive Adapters. |
| **Simple Integration in Flutter** | Easy setup and minimal code required for CRUD operations. |
| **Efficient Storage of Media** | Supports Base64 string storage for offline image access. |

### ****Hive Adapters Registered in main.dart:****

* Hive.registerAdapter(PostAdapter());
* Hive.registerAdapter(CommentAdapter());

# 7. PACKAGES / PLUGINS USED:

|  |  |  |
| --- | --- | --- |
| Package | Purpose | Justification |
| hive & hive\_flutter | Local persistent storage | Offline-first fast data access |
| provider | State management | Efficient UI updates and reactivity |
| uuid | Unique ID generation | Ensures data uniqueness |
| image\_picker | Attach images | Allows users to upload photos |
| intl | Date formatting | Improves readability of timestamps |

# 8. ISSUES AND BUGS ENCOUNTERED & RESOLVED

|  |  |  |
| --- | --- | --- |
| Error Screenshot | Cause | Solution |
| Figure 6.1 –  IMG-20251027-WA0019 | Provider state not updating | Added notifyListeners() appropriately |
| IMG-20251027-WA0018  IMG-20251027-WA0017 | Hive adapters missing | Registered adapters + ran build\_runner |
| Figure 6.3 –  IMG-20251027-WA0015  IMG-20251027-WA0014  IMG-20251027-WA0013  IMG-20251027-WA0012 | Image.file not supported on Web | Replaced with base64 decoding |
| Figure 6.4 –  IMG-20251027-WA0016 | Asset path misconfiguration | Corrected pubspec.yaml and ran flutter pub get |

# CONCLUSION:

RoamFeed provides a practical demonstration of an offline-first mobile application that supports authentic travel storytelling. It successfully addresses the problem of dependency on commercialized content and unreliable connectivity. The integration of Hive, Provider, and Flutter ensures a smooth and scalable application that delivers meaningful user experience to travelers.