PROJECT REPORT

Submitted in partial fulfilment of the requirements for the

MOBILE PROGRAMMING COURSE AT THE LEBANESE UNIVERSITY
FACULTY OF ENGINEERING – BRANCH II

By:

CHOUITY Ghady
KHOURY Chloé

RoumianSpace Mobile Application

Under Supervision of:

Dr AOUDE. Mohammad

Presented on February 2026

1. Project Overview

RoumianSpace is a mobile application designed for space enthusiasts, offering insights into space missions, rockets, crew members, and technological achievements. The app features Cosmo, an AI-powered chatbot, which can answer space-related questions in real-time. Built with user experience in mind, RoumianSpace offers smooth navigation, personalized profiles, saved favorites, audio experiences, and real-time updates. The application is cross-platform, running on mobile, web, and desktop.

2. Technologies & Architecture

Technology	Description
Flutter SDK	Cross-platform framework (Android, iOS, Web, Desktop)
Dart	Programming language (SDK >=3.3.2 <4.0.0)
Firebase	Authentication, Firestore database, Cloud Storage
SpaceX API	REST API (api.spacexdata.com/v4) for real-time data
Python Flask	Backend API server for AI chatbot
Wolfram Alpha	Computational knowledge engine for chatbot queries
Hugging Face	AI model hosting (DeepSeek-V3) for conversational AI

3. Application Features

3.1 Authentication & Profiles

- User registration, login, and password recovery
- Profile customization with photos
- Secure session management

3.2 Home Dashboard

- Rockets and launch pads with details and images
- Interactive carousels and shimmer loading effects
- Cached images for better performance

3.3 Ships & Fleet

- Information on SpaceX ships
- Mission history and status indicators
- Search, filter, and image zoom features

3.4 Crew Information

- Astronaut profiles with missions and biographies
- Agency affiliations and launch history

3.5 AI Chatbot (Cosmo)

- Answers space-related questions using AI
- Combines Wolfram Alpha and Hugging Face models
- Friendly, concise responses with space-themed touches

3.6 Saved Items

- Offline access with SQLite and Sembast databases
- Bookmark rockets, launches, and ships
- Persistent storage across sessions

3.7 Audio Experience

- Space-themed background music and sound effects
- Playback controls for immersive experience

3.8 Onboarding

- Tutorial for first-time users
- Feature highlights and animated splash screens

4. Technical Implementation

4.1 State Management

The application implements BLoC pattern for predictable state management. Each feature module contains dedicated Cubits for handling business logic, API calls, and state transitions. This ensures separation of concerns and makes the codebase highly testable and maintainable.

4.2 Chatbot Architecture

The chatbot feature implements a microservices architecture with a Python Flask backend serving as an API gateway. The backend integrates with two AI services: Wolfram Alpha for computational and factual queries, and Hugging Face's DeepSeek-V3 model for conversational AI. The Flutter frontend communicates with the backend via HTTP requests, enabling real-time question-answering capabilities while maintaining separation of concerns between the mobile UI and AI processing.

4.3 Dependency Injection

GetIt service locator is used for dependency injection, allowing for loosely coupled architecture. All repositories, API services, and Cubits are registered at app startup, enabling easy testing and modular development.

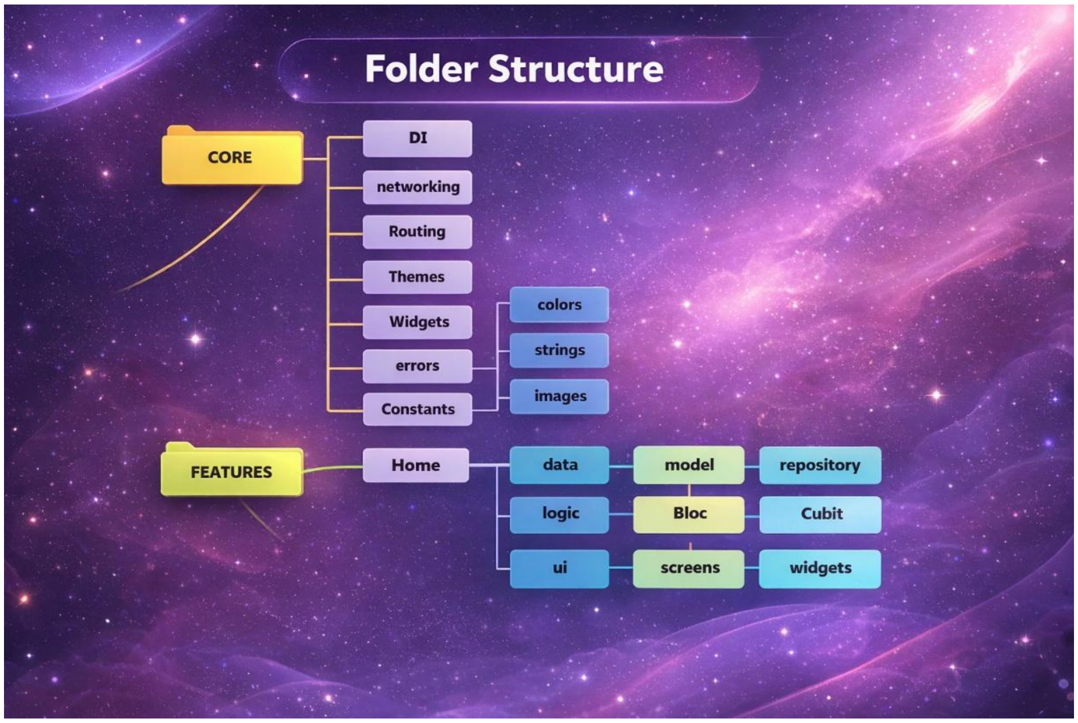
4.4 Error Handling

Comprehensive error handling using the Dartz package for functional programming patterns. The application implements Either types for handling success and failure states, providing detailed error messages and fallback mechanisms for network failures and API errors.

4.5 Data Persistence

Multi-layered data persistence strategy: SharedPreferences for user settings, SQLite for structured mobile data storage, Sembast for cross-platform NoSQL database support (especially web), Firebase Firestore for cloud-synced user profiles, and Flutter Secure Storage for sensitive authentication tokens. This approach ensures optimal performance across all supported platforms.

5. Project Architecture



6. Conclusion

RoumianSpace demonstrates modaern mobile development practices with Flutter, combining clean architecture, real-time data, and AI features. The Cosmo chatbot provides an interactive way to explore space, while features like saved items, audio experiences, and cross-platform support enhance usability.

The project highlights best practices such as modular design, separation of frontend and backend concerns, and scalable architecture—making RoumianSpace a rich, user-friendly, and educational application.