Lebanese University

Faculty of Engineering / Branch II

Barista AI

Technical Report

Course:	Mobile Programming with Flutter
Instructor:	Dr. Mohamad Aoude
Students:	Clovis Abou Kheir, Imad El Murr
Date:	February 1, 2026

Contents

1 Abstract

Barista AI is a Flutter-based mobile application that enables users to discover and explore cocktail recipes through a modern, animated interface. The app integrates Firebase Authentication and Cloud Firestore to provide secure per-user favorites, real-time updates, and scalable data storage. This report documents the system architecture, data model, state management approach, user experience design, and the main technical trade-offs of the current prototype. It also outlines future enhancements such as true AI-driven recipe generation and personalization.

2 Introduction

Cocktail discovery apps commonly provide static lists and unremarkable UI experiences. The Barista AI project explores a more immersive approach by combining a curated recipe dataset with dynamic visuals, animations, and an AI-inspired creation flow. The project was implemented as a university assignment for the "Mobile Programming with Flutter" course and aims to demonstrate:

- A complete Flutter application structure with modular layers.
- Real-time data synchronization using Firebase Cloud Firestore.
- Authentication flows and access control in navigation.
- A polished UI with reusable components and animated transitions.

3 System Overview

The application follows a clean layered architecture:

- **Presentation Layer**: Screens, widgets, and visual components written in Flutter.
- **State Layer**: Controllers using Provider + ChangeNotifier to expose reactive state to the UI.
- **Data Layer**: Repositories that abstract Firebase Auth and Firestore operations.
- **Model Layer**: Immutable data objects such as `Cocktail` and `Ingredient`.

A high-level flow of control is:

1. The user authenticates (or is redirected to sign-in).
2. The Discover and Search screens subscribe to Firestore streams.

3. State controllers publish updates to the UI.

4. The user can favorite items, which are stored under their user document.

4 Architecture and Routing

Navigation is handled by **GoRouter** with guards that enforce authentication. The router starts at a splash screen and transitions into the app shell after the authentication state is initialized. The shell provides bottom-tab navigation between Discover, Search, Favorites, Creator Studio, and Profile screens. This design ensures that the UX remains consistent while individual screens manage their own state.

4.1 State Management

Provider is used to inject repositories and controllers into the widget tree. The `AuthController` observes Firebase authentication changes, while the `FavoritesController` listens to user-specific Firestore subcollections. This approach simplifies dependency injection, keeps UI widgets stateless where possible, and ensures that the UI remains reactive to backend updates.

5 Data Model and Firestore Design

The data model centers around cocktails and per-user favorites.

5.1 Cocktail Documents

Each cocktail document contains metadata (name, subtitle, tags), rendering data (color gradients for UI cards and layers), and recipe content (ingredients and steps). These fields are intentionally structured to support both list views and detail views without additional client-side transformations.

5.2 Favorites Subcollection

Favorites are stored under `users/{uid}/favorites` as documents keyed by cocktail ID. This design provides:

- Per-user isolation of favorite records.
- Simple toggle logic (create vs. delete document).
- Real-time synchronization between devices.

6 User Experience and UI Design

Barista AI emphasizes a vibrant, premium aesthetic. Key UI features include:

- Gradient cards and animated hero transitions.
- A cocktail detail screen with a recipe tab and an animated "pour" visualization.
- A creator studio with interactive controls (chips, sliders, switches) for mock generation.

Animations and visual layers are implemented with custom painters and Flutter rendering primitives to simulate liquid layering inside a cocktail glass. This provides a visually engaging element without requiring external animation assets.

7 Creator Studio and AI Concept

The Creator Studio is a mock interface intended to simulate a future AI generation workflow. Users select flavor profiles, sweetness levels, and drink attributes, and the UI provides a visual preview of the resulting cocktail. The "Generate" action currently produces a mock name and preview only. This is intentionally scoped as a prototype feature to demonstrate UX flow prior to connecting to a real generative model.

8 Security and Authentication

Firebase Authentication enforces email/password sign-in before accessing application content. Navigation guards prevent unauthenticated access to protected routes. Firestore access relies on user authentication; security rules should be configured accordingly in production. For development, the database can be run in test mode with limited security.

9 Testing and Debug Utilities

The project includes a Firestore seeder utility that can populate cocktail documents with mock data in debug builds. This improves development velocity and ensures consistent test data. Flutterś built-in testing framework is available for widget and unit testing, although this prototype primarily focuses on integration behavior and UI flow.

10 Limitations

- The AI generation is currently a mocked experience without an external model.
- There is no offline caching or persistence for network interruption.

- Cocktail data relies on Firestore indexing and requires correct setup.

11 Future Work

Potential improvements include:

- Integrating real AI generation (OpenAI, Vertex AI, or custom model APIs).
- User personalization and recommendation logic.
- Admin tooling for cocktail management.
- Offline-first caching with local persistence.

12 Conclusion

Barista AI demonstrates a complete, modern Flutter application that combines authentication, real-time data access, and high-quality UI design. The project architecture provides clear separation of concerns, while Firebase services simplify backend infrastructure. As a university project, it achieves its goals of illustrating best practices in Flutter development and provides a strong foundation for future expansion.