6.1 Introduction Section

This document presents the architecture and detailed design for the *Finance Buddy* mobile application.

Finance Buddy is an iOS-based personal finance app that combines AI-powered budget analysis and gamified financial learning to help users improve their money habits. The system integrates a Firebase backend, an iOS Swift front-end, and an AI categorization engine built around Retrieval-Augmented Generation (RAG). Together, these components deliver a smart dashboard, goal tracking, reminders, and personalized insights for everyday users seeking better financial literacy.ct.

6.1.1 System Objectives Section

The objectives of Finance Buddy are to:

1. **Promote financial literacy** through interactive, gamified lessons that keep users engaged while learning budgeting concepts.

2. **Enable smart expense management** by automatically categorizing transactions and generating real-time spending insights using AI.

3. **Support personalized goal setting**, allowing users to create, track, and visualize progress toward savings or spending objectives.

4. **Deliver accessibility and ease of use** via a clean, mobile-first interface optimized for iOS and later extensible to cross-platform frameworks.

5. **Protect user privacy and security** by encrypting data in transit and at rest, employing Firebase Authentication and optional multi-factor login.

Overall, the goal is to empower students, young professionals, and general users to make informed financial decisions through data visualization, adaptive recommendations, and gamified learning experiences.

6.1.2 Hardware, Software, and Human Interfaces Section

User Device: iPhone 8 or newer running iOS 16 or later; supports touch input, Face ID/Touch ID authentication, and push notifications.

Network Connectivity: Wi-Fi or cellular network required for synchronization with Firebase Cloud Firestore and receipt of remote notifications.

6.2 Architectural Design Section

The system architecture follows a modular MVC pattern integrated with Firebase. The design ensures separation of concerns and scalability for future cross-platform deployment.

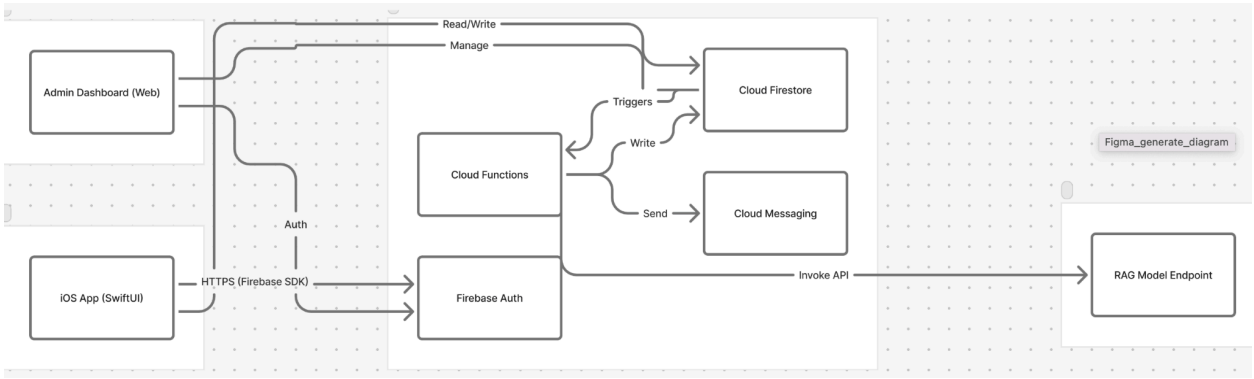
6.2.1 Major Software Components Section

Component	Description	Related Functional Requirements
Authentication Module	Handles user registration, login, logout, and MFA via Firebase.	FR1, FR7–FR9
Dashboard Module	Displays categorized expenses, visual charts, and progress summaries.	FR2, FR15–FR20
Goal Manager	Supports creation, tracking, and reminder scheduling for user savings goals.	FR3, FR31–FR32
AI Categorization Engine	Uses RAG model to classify expenses from CSV imports or receipts.	FR4, FR10–FR14, FR24–FR26
Gamification Subsystem	Handles lessons, badges, leaderboards, and daily challenges.	FR5, FR27–FR30
Notification System	Manages push alerts using Firebase Cloud Messaging.	FR6, FR31–FR33
Admin Dashboard	Enables content management and user moderation.	FR34–FR36
Reporting Service	Generates monthly CSV/PDF summaries and email deliveries.	FR21–FR23

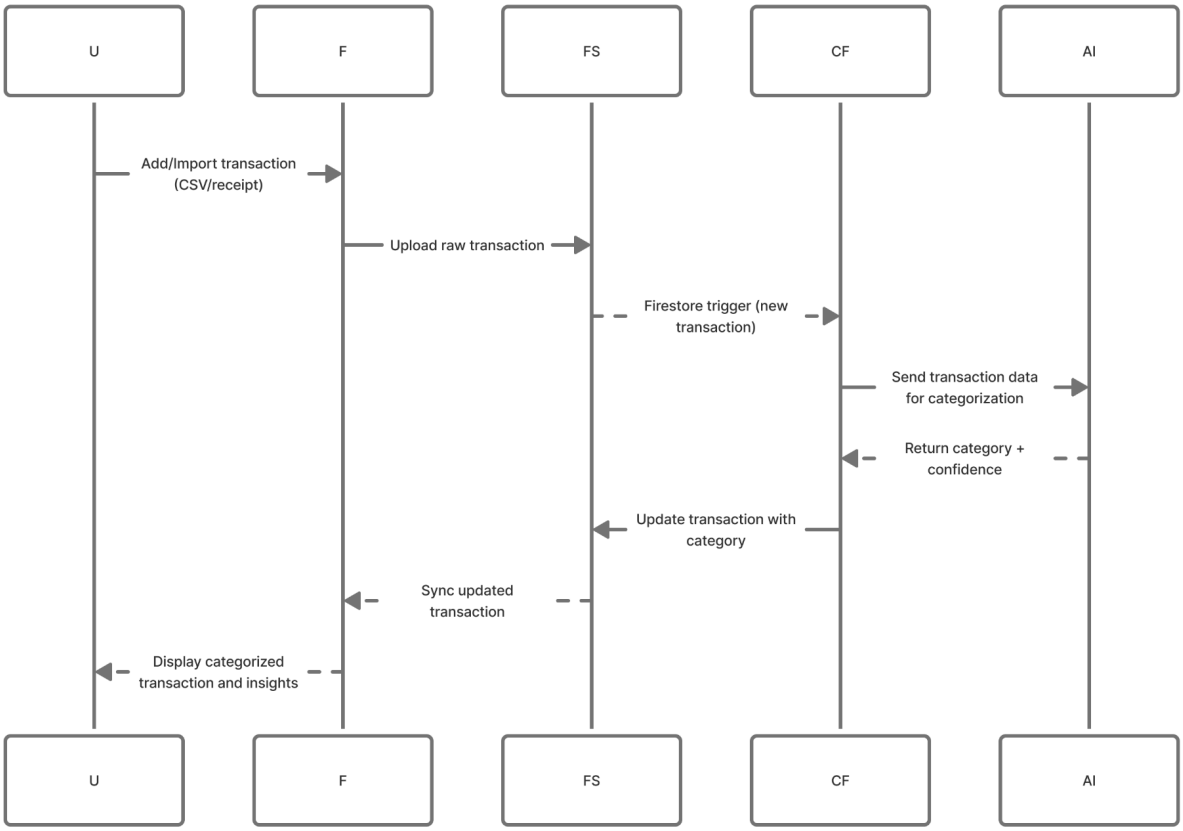
6.2.2 Major Software Interactions Section

1. Frontend ↔ Firebase Backend:
   The SwiftUI frontend communicates with Firebase Firestore via the Firebase SDK. Authentication tokens secure all requests using HTTPS.
2. Backend ↔ AI Service:
   Cloud Functions call a hosted Python RAG model endpoint with transaction data. The AI service returns categorized labels and confidence levels in JSON.
3. Backend ↔ Notification Service:
   Firebase Cloud Messaging sends push notifications triggered by Firestore events.
4. Backend ↔ Admin Panel:
   The admin dashboard accesses Firestore collections for lesson management and user accounts.
5. Frontend ↔ User:
   The GUI provides touch-driven interaction; all major views synchronize state with Firestore listeners for real-time updates.

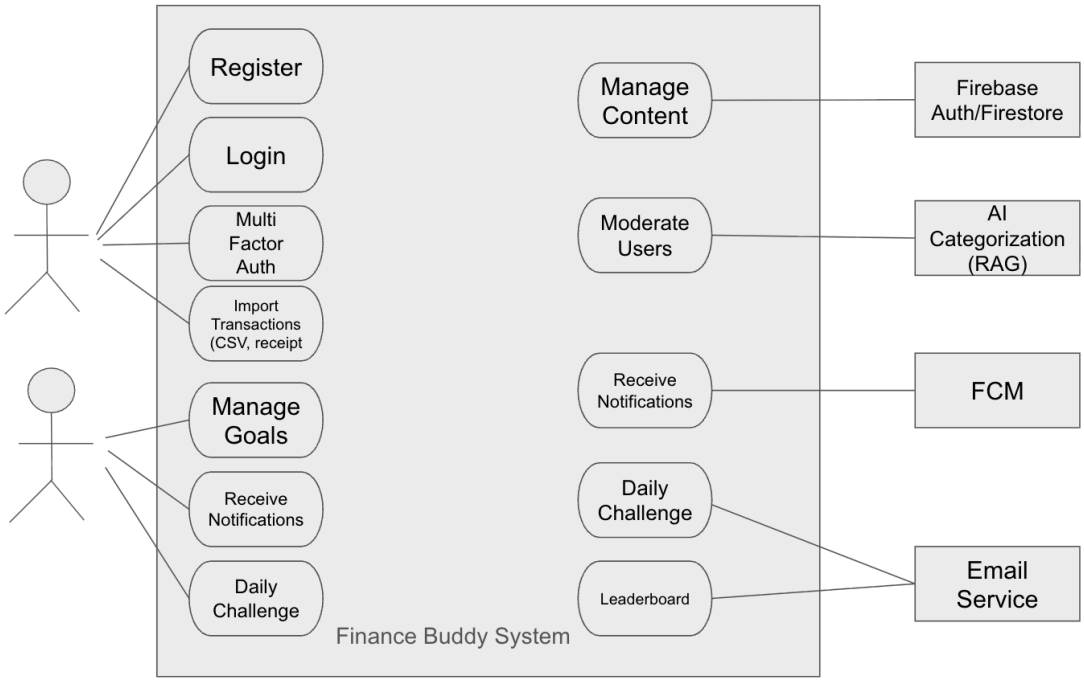
6.2.3 Architectural Design Diagrams Section



Component Diagram



Sequence Diagram



User Case Diagram