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.

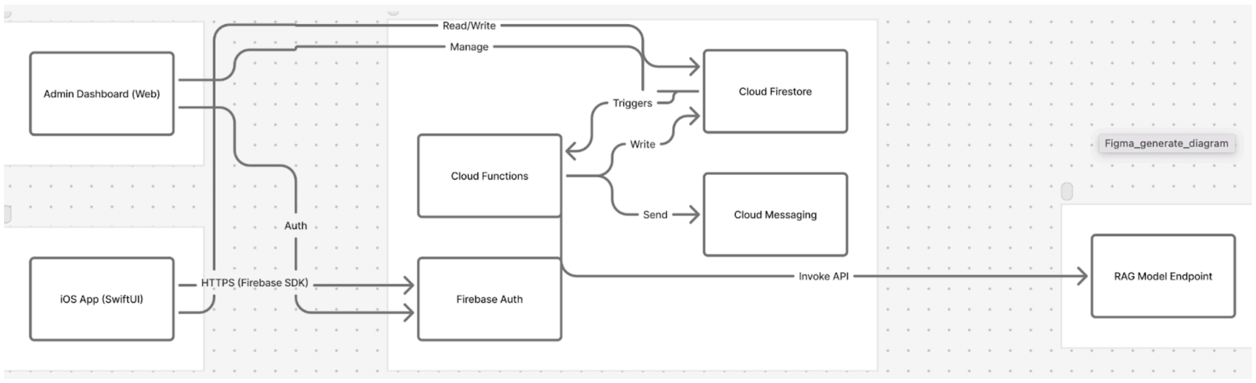
6.2.1 Major Software Components Section

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

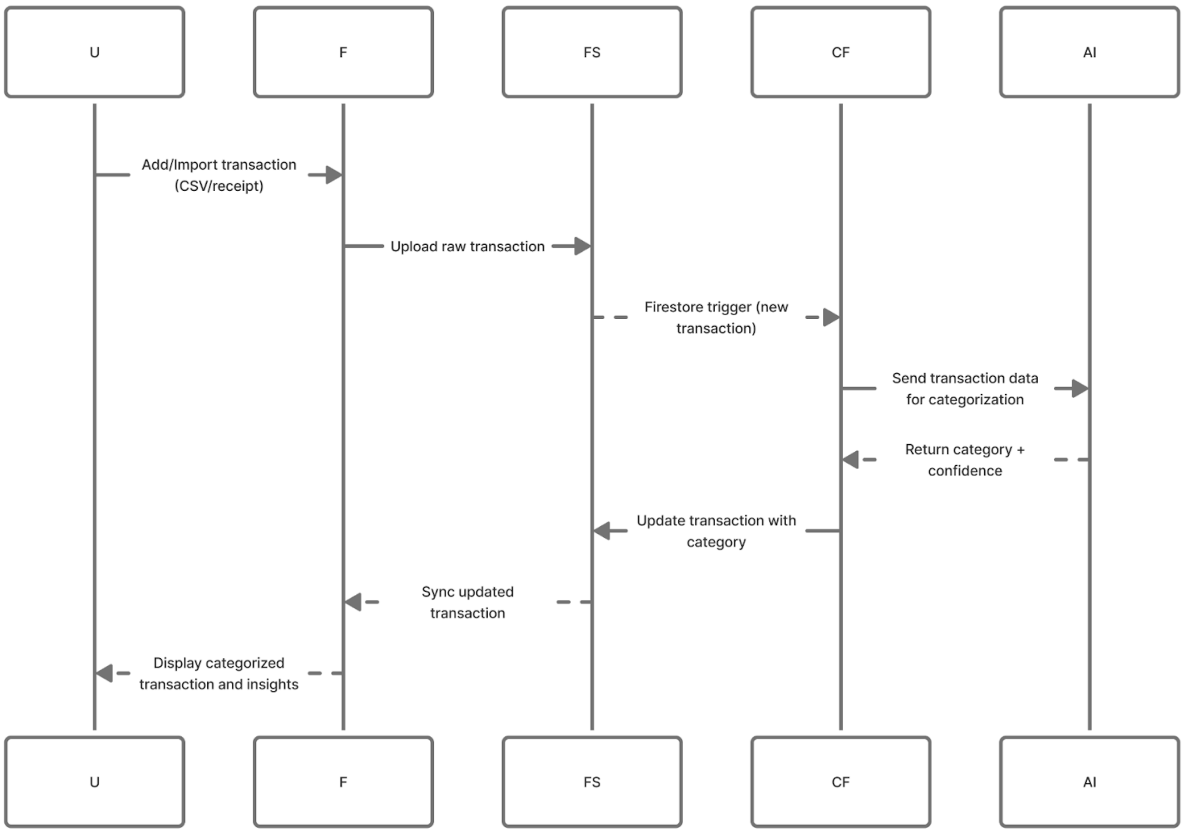
6.2.2 Major Software Interactions Section

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.

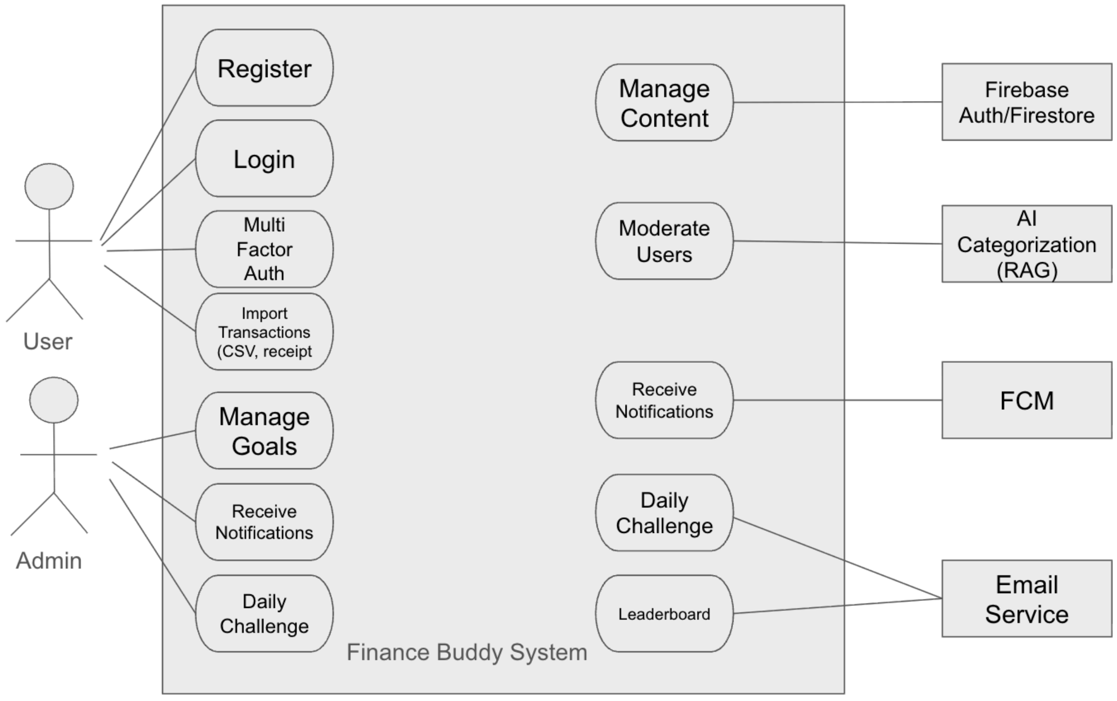
6.2.3 Architectural Design Diagrams Section



Component Diagram


U
F
FS
CF
AI
Add/Import transaction (CSV/receipt)
Upload raw transaction
Firestore trigger (new transaction)
Send transaction data for categorization
Return category + confidence
Update transaction with category
Sync updated transaction
Display categorized transaction and insights
U
F
FS
CF
AI

Sequence Diagram



User Case Diagram

6.3 Detailed CSC and CSU Descriptions Section

In this section, we describe the key CSCs associated with the Finance Buddy software architecture.

6.3.1 Authentication CSC

Manages user identity, secure access, MFA, and session lifecycle. Ensures that all data interactions occur under a verified Firebase Authentication context.

CSU Name	Description
AuthService	Wrapper around FirebaseAuth; handles user creation, login, logout, and token refresh.
UserSessionManager	Persists authenticated session state in the app; exposes @Published user state for SwiftUI.
MFAHandler	Handles multi-factor enrollment and verification using SMS/email links.
UserProfileRepository	Connects to Firestore's /users/{uid} document for profile settings and metadata.

Related Functional Requirements: FR1, FR7–FR9

6.3.2 Dashboard CSC

Displays real-time spending data, charts, AI categorizations, and aggregated insights. Updates live using Firestore listeners.

CSU Name	Description
DashboardViewModel	Retrieves expense summaries, categories, and updates UI state.
ExpenseRepository	Handles Firestore reads/writes for /expenses collections.
ChartsAdapter	Formats data for chart components (bar charts, donut charts, trends).
InsightsEngine	Calculates weekly summaries, spending streaks, overspending flags.

Related Functional Requirements: FR2, FR15–FR20

6.3.3 Goal Manager CSC

Manages creation, scheduling, tracking, and completion of user savings goals.

CSU Name	Description
GoalModel	Data model representing a savings or spending goal.
GoalRepository	CRUD operations for Firestore collection /goals/.
GoalProgressEngine	Computes progress percentage, estimated completion dates.

ReminderScheduler	Uses FCM and local notifications to trigger reminders.

Related Functional Requirements: FR3, FR31–FR32

6.3.4 AI Categorization CSC

Classifies user transactions using a Retrieval-Augmented Generation (RAG) engine. Supports both automatic and user-corrected categories.

CSU Name	Description
RAGRequestFormatter	Converts transactions into the prompt + metadata for the Python AI service.
CategorizationCloudFunction	Google Cloud Function that receives transactions and returns RAG predictions.
AIResponseParser	Parses JSON output into normalized labels + confidence values.
UserCorrectionHandler	Updates Firestore when a user overrides a category; feeds correction back to training logs.

Related Functional Requirements: FR4, FR10–FR14, FR24–FR26

6.3.5 Gamification CSC

Drives user engagement through lessons, streaks, badges, level-ups, and leaderboards.

CSU Name	Description
LessonModel	Represents the content and metadata for each financial literacy lesson.
LessonRepository	Fetches lessons from Firestore.
BadgeEngine	Determines when a badge unlocks; stores it under `/badges/{uid}`.
LeaderboardService	Computes ranking based on completed challenges and streaks.

Related Functional Requirements: FR5, FR27–FR30

6.3.6 Notification System CSC

Handles both local device notifications and remote push notifications sent through Firebase Cloud Messaging.

CSU Name	Description
NotificationManager	Schedules local reminders and handles notification permissions.
FCMTokenService	Registers and syncs device FCM tokens with Firestore.
NotificationTriggerCloudFunction	Server-side logic to push alerts based on Firestore events (goal deadlines, new insights).

Related Functional Requirements: FR6, FR31–FR33

6.3.7 Admin Dashboard CSC

Provides tools for administrators to manage app content, moderate users, and adjust lessons or badges.

CSU Name	Description
AdminWebPortal	Web interface built with React or Firebase Hosting.
AdminFirestoreService	Backend access to all moderated collections /lessons/, /users/, /reports/.
ContentManager	Allows admins to publish/edit lessons, badges, and challenges.

Related Functional Requirements: FR34–FR36

6.3.8 Reporting Service CSC

Generates monthly reports (CSV, PDF) and sends emails via Cloud Functions.

CSU Name	Description
ReportBuilder	Aggregates user expenses into a structured monthly dataset.
CSVGenerator	Creates exportable CSV from the aggregated dataset.

PDFGenerator	Creates PDF summaries (e.g., spending categories, charts).
EmailDeliveryFunction	Sends reports to the user via email using SendGrid or Gmail API.

Related Functional Requirements: FR21–FR23

6.3.1 Detailed Class Descriptions Section

The following sections provide brief descriptions of the core classes used in the Finance Buddy application. Each subsection describes a class from each major CSC, including its purpose, key fields, and essential methods. Together, these classes form the primary functional units supporting authentication, dashboards, goal tracking, AI categorization, gamification, notifications, administration, and file reporting.

6.3.1.1 AuthService (Authentication CSC)

Provides the app’s interface to Firebase Authentication, handling sign-up, login, logout, and session updates.

Fields

- currentUser: UserProfile? — Cached authenticated user.
- authHandle: Any? — Listener for Firebase auth changes.

Methods

- signIn(email, password) — Authenticates a user.
- signUp(email, password) — Creates a new account.
- signOut() — Ends the session.

- observeAuthChanges() — Watches for login/logout events.

6.3.1.2 DashboardViewModel (Dashboard CSC)

Supplies dashboard data to the UI by retrieving expenses, summarizing them, and updating charts and insights.

Fields

- expenses: [Expense] — Current expense list.
- summary: SpendingSummary — Aggregated totals.

Methods

- loadExpenses() — Fetches user expenses.
- computeInsights() — Builds spending flags and trends.
- refreshDashboard() — Updates the dashboard state.

6.3.1.3 Goal (Goal Manager CSC)

Represents a user's savings or spending goal, tracking progress and deadlines.

Fields

- title: String — Goal name.
- targetAmount: Double — Goal target.
- currentAmount: Double — Amount saved so far.

Methods

- progress() — Returns percent complete.
- isCompleted() — Checks if goal is finished.
- addContribution(amount) — Updates progress.

6.3.1.4 AICategorizationService (AI Categorization CSC)
Sends expense data to the RAG model endpoint and returns category predictions with confidence scores.

Fields

- endpointURL: URL — AI API location.
- networkClient: Any — HTTP communication handler.

Methods

- categorize(expense) — Returns category + confidence.
- buildPayload() — Formats request body.
- parseResponse(data) — Extracts AI output.

6.3.1.5 Lesson (Gamification CSC)
Represents one educational lesson used in Finance Buddy's gamified learning system.

Fields

- title: String — Lesson name.
- content: String — Instructional text.
- difficultyLevel: Int — Lesson rating.

Methods

- isShort() — Checks estimated time.
- toFirestoreData() — Serializes lesson content.

6.3.1.6 NotificationManager (Notification System CSC)
Handles local reminders and integrates with Firebase Cloud Messaging for push alerts.

Fields

- fcmToken: String? — Registered device token.
- permissionGranted: Bool — Notification status.

Methods

- requestAuthorization() — Asks for permission.
- scheduleGoalReminder(goal) — Schedules reminders.
- registerFCMToken(token) — Syncs the token.

6.3.1.7 AdminFirestoreService (Admin Dashboard CSC)

Allows administrators to manage lessons, badges, and moderated user content stored in Firestore.

Fields

- lessonsPath: String — Base path for lesson data.
- usersPath: String — Admin view of user profiles.

Methods

- fetchLessons() — Loads all lessons.
- updateLesson(lesson) — Saves edits.
- fetchUserData(userId) — Retrieves user info.

6.3.1.8 ReportBuilder (Reporting Service CSC)

Generates monthly financial summaries exportable as CSV or PDF.

Fields

- expenses: [Expense] — Data to include.

- periodStart: Date — Reporting window start.
- periodEnd: Date — Reporting window end.

Methods

- buildSummary() — Computes totals + category breakdowns.
- generateCSV() — Produces CSV text.
- generatePDF() — Produces PDF output.

6.3.2 Detailed Interface Descriptions Section

In this section, we summarize the key interface descriptions at each CSU component of Finance Buddy.

6.3.2.1 Authentication Interface

The Authentication Interface defines how the UI and session manager communicate with Firebase Authentication. It passes user credentials to Firebase, receives login or logout updates, and notifies other components when authentication state changes. This interface ensures secure access to all other subsystems.

6.3.2.2 Dashboard Data Interface

The Dashboard Data Interface manages communication between the dashboard view model, the expense repository, and Firestore. It retrieves expenses, listens for real-time updates, and delivers summarized data back to the UI. This interface keeps dashboard information current without requiring manual refreshes.

6.3.2.3 Goal Management Interface

The Goal Management Interface connects the goal-tracking components with Firestore and the notification system. It sends new or updated goals to storage, returns progress updates to the UI, and triggers reminders when deadlines approach. This interface ensures goals stay synchronized across the app.

6.3.2.4 AI Categorization Interface

The AI Categorization Interface sends expense details to the RAG model endpoint and returns predicted categories. It formats requests, parses responses, and forwards user corrections back to Firestore. This interface provides consistent communication with the AI service.

6.3.2.5 Gamification Interface

The Gamification Interface handles how lessons, badges, and user activity interact. It retrieves lesson content, records completions, and notifies the badge system when achievements are unlocked. This interface links learning progress with reward updates.

6.3.2.6 Notification Delivery Interface

The Notification Delivery Interface coordinates alerts between the notification manager, Firebase Cloud Messaging, and other CSCs. It schedules reminders, handles token registration, and delivers incoming notification data to the appropriate views. This interface ensures timely and accurate alert delivery.

6.3.2.7 Admin Management Interface

The Admin Management Interface connects the admin dashboard with backend Firestore collections. It sends content updates, retrieves moderated user data, and propagates changes to related components such as lessons or badges. This interface supports controlled administrative actions.

6.3.2.8 Reporting Interface

The Reporting Interface links the report builder with expense data and output delivery. It collects transactions for the reporting period, formats the results, and returns generated CSV or PDF files to the requester. This interface streamlines the report-generation process.

6.3.3 Detailed Data Structure Descriptions Section

In this section, we summarize the core data structures in each of the CSUs used in Finance Buddy.

6.3.3.1 Authentication CSC Data Structures

The Authentication CSC primarily uses Firebase Authentication response objects and lightweight Firestore user documents. The main data structure is the UserProfile document, containing fields such as UID, email, display name, creation timestamp, and preference flags. Authentication also relies on standardized JSON packets returned by Firebase for login status, errors, and MFA steps. These structures allow secure and consistent transmission of identity data across the app.

6.3.3.2 Dashboard CSC Data Structures

The Dashboard CSC uses Firestore documents representing individual expenses and aggregated summary structures used for real-time updates. Each expense document includes amount, category, timestamp, raw description, and metadata tags. The dashboard also uses a SpendingSummary structure, which aggregates totals, category breakdowns, and time-based trends. These data structures support efficient querying and display of financial insights.

6.3.3.3 Goal Manager CSC Data Structures

The Goal Manager CSC relies on Firestore goal documents that store the title, target amount, current amount, dates, and completion status. It also makes use of a lightweight GoalProgress structure that calculates derived values such as percentage completion and remaining time. Reminders use simple metadata packets containing goal IDs, due dates, and message content. Together, these structures keep goal information consistent across storage, UI, and notifications.

6.3.3.4 AI Categorization CSC Data Structures

The AI Categorization CSC uses structured JSON payloads to communicate with the RAG model. Outgoing requests contain fields like amount, merchant string, currency, and transaction context. Returned responses follow a fixed format including predicted category, confidence score, and optional model explanations. A correction packet structure is also used when users override predictions, storing expense ID, new category, and timestamp. These formats maintain interoperability between the app, backend, and AI engine.

6.3.3.5 Gamification CSC Data Structures

The Gamification CSC uses Firestore lesson and badge documents. Lesson documents include fields such as title, content, estimated minutes, tags, and difficulty rating. Badge documents track badge title, icon, and unlocked timestamp. A small ProgressRecord data structure captures lesson completions and challenge results. These structures help organize learning content and reward states in a standardized format.

6.3.3.6 Notification System CSC Data Structures

The Notification System CSC uses small data packets for scheduling reminders and handling push notifications. Local reminder structures contain goal IDs, fire dates, and message text. FCM remote notifications use JSON envelopes with title, body, action type, and optional payload data such as a goal or expense reference. Device tokens are stored as simple string fields within Firestore user documents. These data formats ensure reliable and structured delivery of alerts.

6.3.3.7 Admin Dashboard CSC Data Structures

The Admin Dashboard CSC uses Firestore documents representing lessons, users, and content metadata. Lesson documents include editable fields such as titles, summaries, tags, and version numbers. Moderation logs use compact structures containing user IDs, timestamps, and admin actions. The admin panel communicates using JSON request/response packets that wrap content updates or validation results. These data structures support controlled content management across the platform.

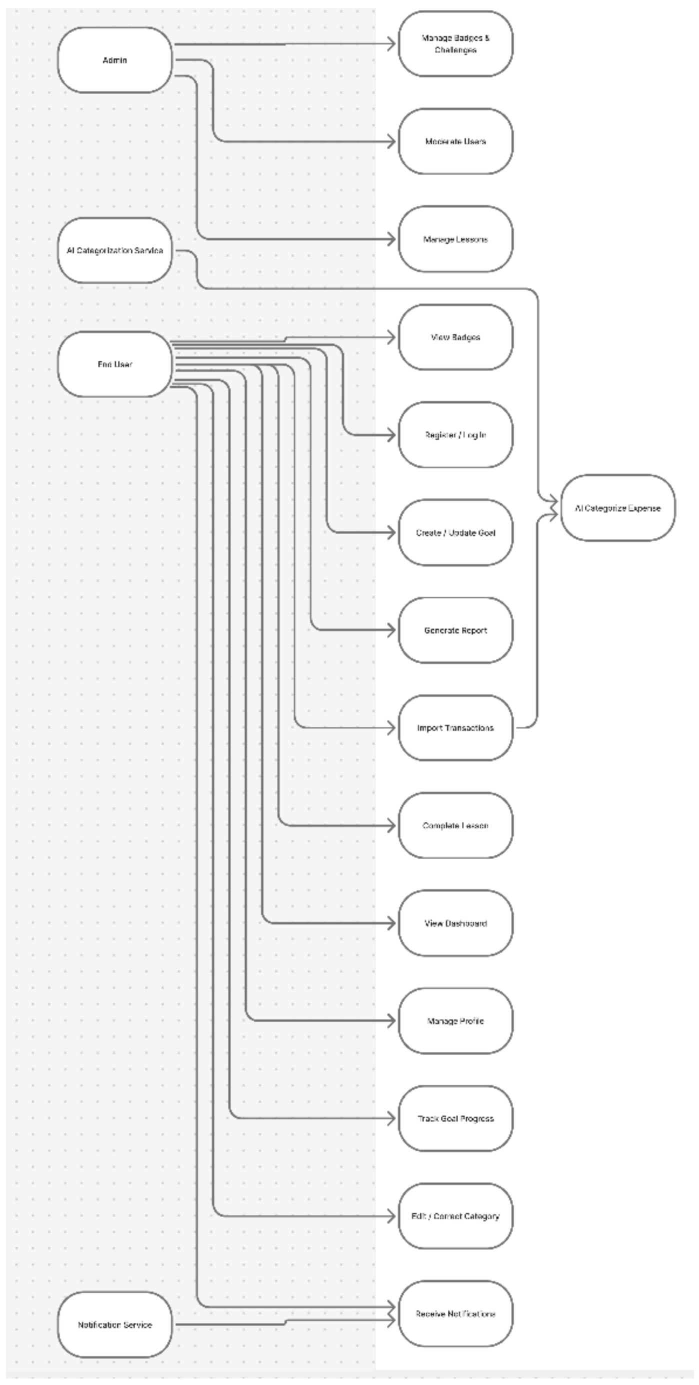
6.3.3.8 Reporting Service CSC Data Structures

The Reporting Service CSC aggregates expenses into structured report datasets used for export. It uses a MonthlyReport structure containing total spending, category totals, date ranges, and grouped expense lists. CSV exports convert these fields into delimited text rows, while PDF generation relies on formatted tables derived from the same data. These structures provide a consistent foundation for producing sharable financial summaries.

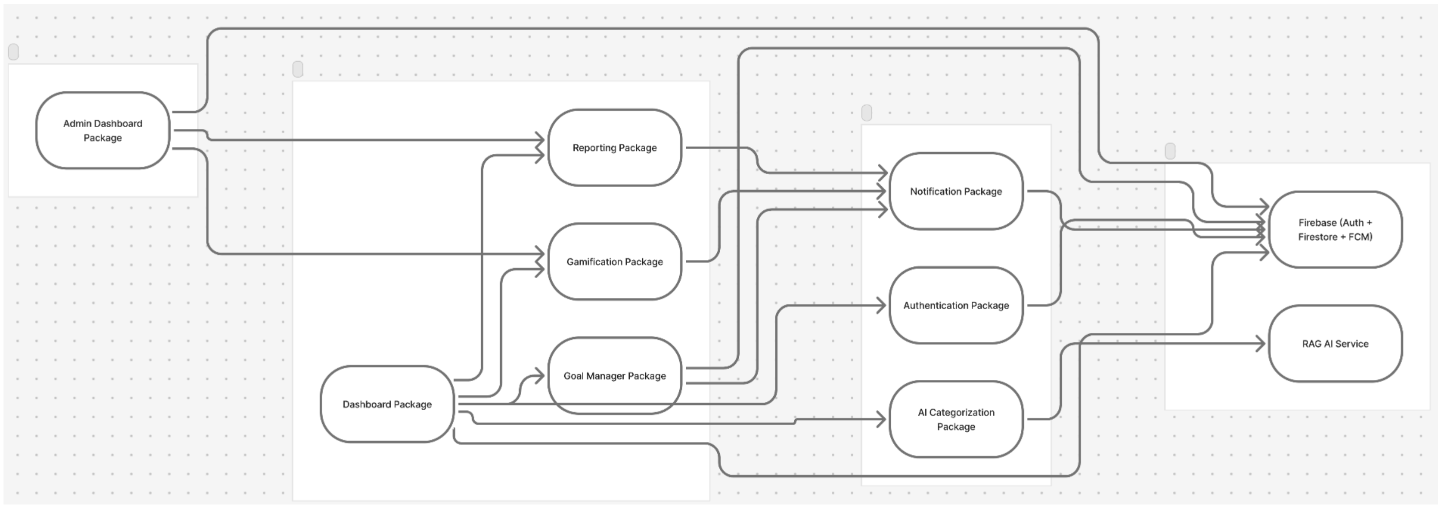
6.3.4 Detailed Design Diagrams Section

This section contains detailed diagrams for all facets of the functionality of Finance Buddy. Below are detailed use case diagrams, package diagrams, class diagrams, and state diagrams.

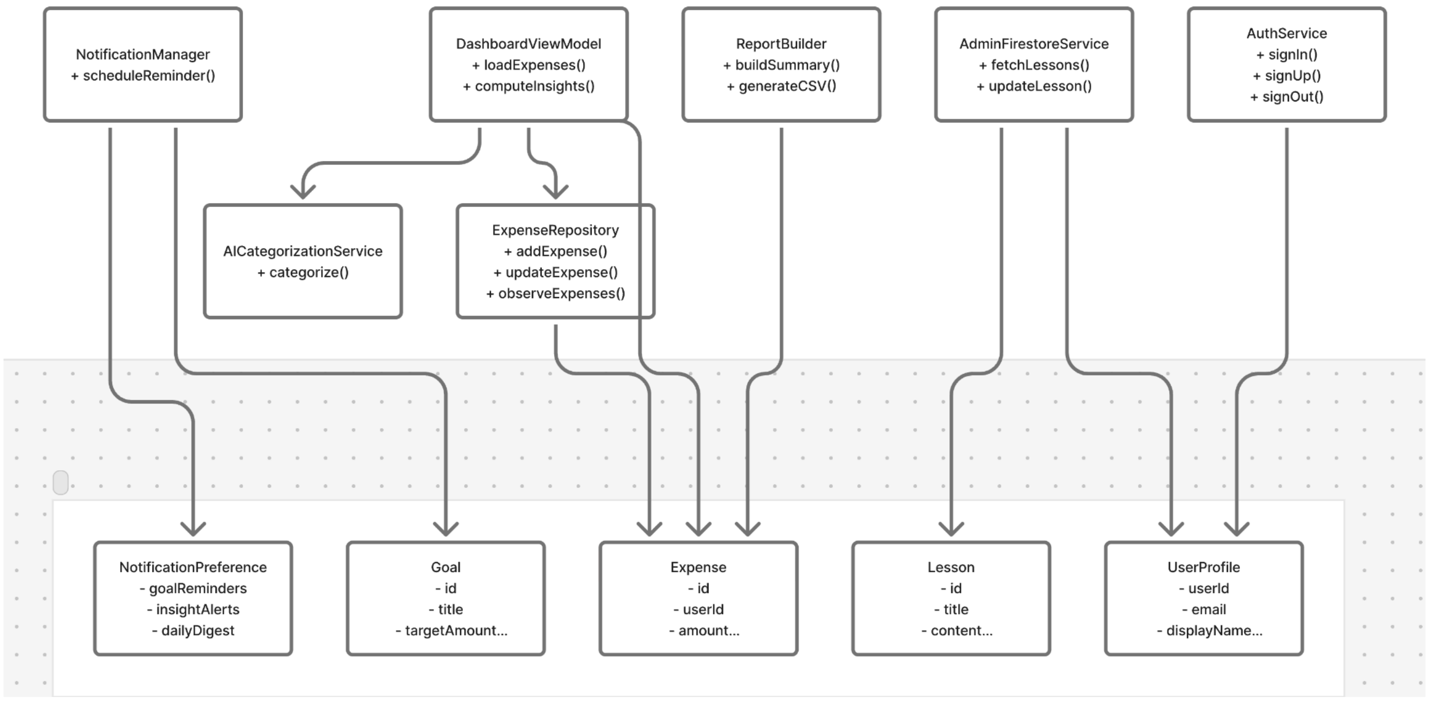
Use Case Diagram



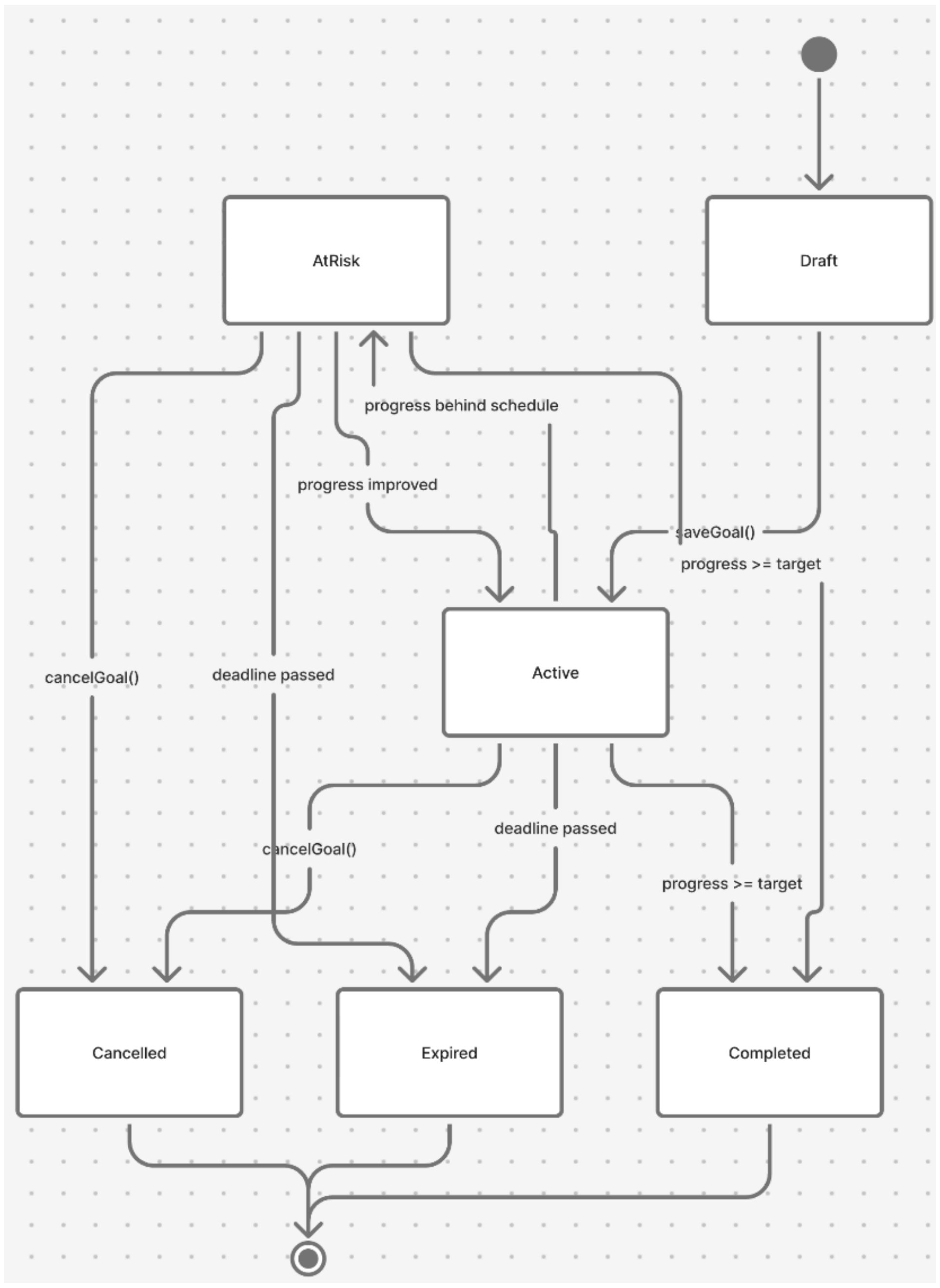
Package Diagram



Class Diagram



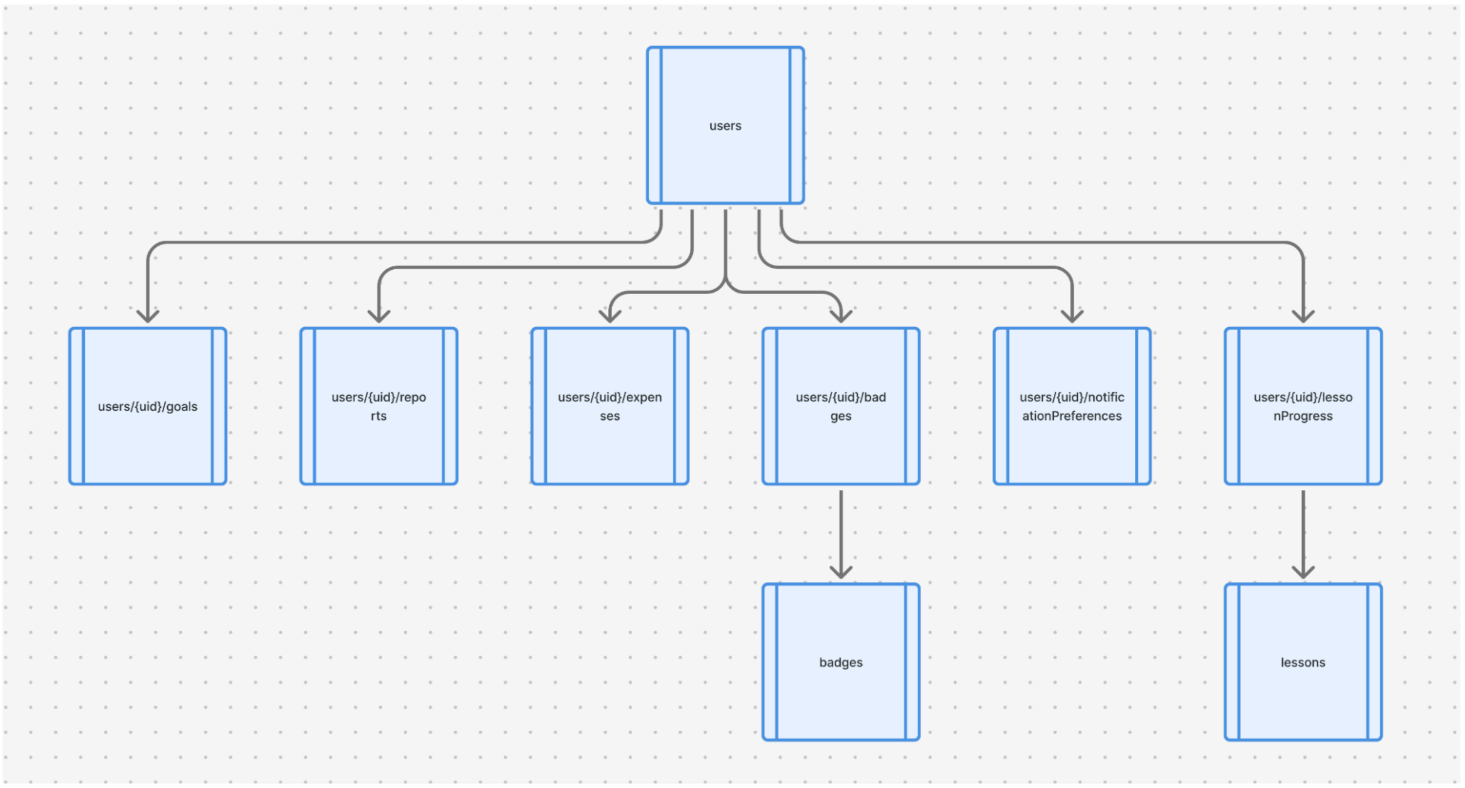
State Diagram



6.4 Database Design and Description Section

Finance Buddy uses Firebase Cloud Firestore as its primary database. Firestore is a NoSQL, document-oriented database that supports real-time synchronization, scalable storage, and secure access through Firebase Authentication. This section describes the structure of the database, how the application interacts with it, and how security is enforced across all modules.

6.4.1 Database Schema (Entity-Relationship Overview)



The Finance Buddy database schema is organized around a central users collection, with most user-specific data stored in subcollections underneath each user document. Global content such as lessons and badge definitions are stored in top-level collections and referenced by user-specific progress documents. This structure supports multi-tenant storage, clear data ownership per user, and efficient querying for both the mobile app and admin tools.

Top-level collections

- Users
  - Each document represents a single authenticated user, keyed by Firebase UID.
  - Typical fields: email, displayName, currencyCode, createdAt, riskProfile, and basic profile settings.
- Lessons
  - Global catalog of financial literacy lessons.
  - Fields: title, summary, content, difficultyLevel, estimatedMinutes, tags.
- Badges
  - Global definitions for achievements and badges.
  - Fields: name, description, iconName, unlockCriteria (e.g., required actions or thresholds).

<u>User subcollections</u>

Under each users/{uid} document, the following subcollections are defined:

- users/{uid}/expenses
  Stores all expense records for a user.
  Fields: amount, currencyCode, category, rawDescription, date, sourceType, aiConfidence, isUserCorrected, createdAt.
  Relationship: Many expenses belong to one user.

- users/{uid}/goals
  Stores the user's financial goals.
  Fields: title, targetAmount, currentAmount, startDate, endDate, status (e.g., Active, Completed, Expired, Cancelled).
  Relationship: Many goals belong to one user.
- users/{uid}/notificationPreferences
  Typically a single document that holds notification settings.
  Fields: goalRemindersEnabled, insightAlertsEnabled, dailyDigestEnabled, preferredHourOfDay.
  Relationship: One notification preference document per user.
- users/{uid}/badges
  User-specific badge state referencing global badges.
  Fields: badgeId (refers to badges), unlockedAt, progress (optional).

Relationship: Each user-badge document links one user to one global badge.
users/{uid}/lessonProgress
Tracks per-user progress on global lessons.
Fields: lessonId (refers to lessons), status (e.g., NotStarted, InProgress, Completed), lastViewedAt, completedAt, score (if quizzes are used).
Relationship: Many progress records for one user; each relates to a single lesson.

- users/{uid}/reports
  Stores metadata for generated monthly reports.
  Fields: month, year, totalSpending, topCategories, csvPath, pdfPath, createdAt.
  Relationship: A user can have many reports, one per reporting period.

Entity relationships (conceptual)

- User → Expenses: One user has many expenses (1:N).
- User → Goals: One user has many goals (1:N).
- User → NotificationPreferences: One user has one notification preference document (1:1).
- User → Badges (user-specific): One user can have many badge instances (1:N), each pointing to a global badge.
- User → LessonProgress: One user can have many lesson progress records (1:N), each referencing a global lesson.
- Lessons → LessonProgress: One lesson can be referenced by many progress records (1:N).
- Badges → UserBadges: One global badge can be unlocked by many users (1:N).
- User → Reports: One user can have many generated reports (1:N).

This schema leverages Firestore’s document/subcollection model to keep all user-specific data logically grouped under users/{uid}, while shared content (lessons, badges) remains centralized and reusable. It supports real-time listeners on subcollections (e.g., expenses and goals) and aligns cleanly with the MVC and CSCs described in earlier sections.