CMSI 4071: Senior Project II

Finance Buddy — Software Design Description (SDD)

Authors: A'Kaia, Natasha, and Temi
Date: February 2026
Instructor: Dr. Brian Johnson
Course: CMSI 4071: Senior Project II

1.1 Introduction

This document presents the architecture and detailed design for the software system Finance Buddy. The project is an AI-powered personal finance assistant designed to help users track their expenses, analyze their financial goals, and receive personalized recommendations for improving budgeting and spending behavior.

Finance Buddy integrates machine learning and natural language interfaces to make financial management intuitive and proactive. It aims to bridge the gap between traditional budgeting apps and intelligent advisory systems, giving users meaningful insights and actions rather than static numbers.

1.1.1 System Objectives

The objective of Finance Buddy is to provide a smart, user-friendly financial management platform that:

- Tracks user income, expenses, and saving goals in real time.
- Uses natural language inputs (chat interface) to record and interpret financial actions.
- Generates personalized recommendations using an AI model trained on spending patterns.
- Encourages users through gamified progress tracking and visual progress bars.
- Offers automated notifications, reminders, and adaptive goal recommendations.

Ultimately, Finance Buddy strives to empower users to achieve financial wellness through intelligent automation and data-driven insights.

1.1.2 Hardware, Software, and Human Interfaces

1.1.2.1 Hardware Interfaces

- **Client Devices:** Mobile (iOS).
- **Server:** Google Firebase
- **Database:** Google Firebase
- **AI-Powered Chatbot:** OpenAI API

1.1.2.2 Software Interfaces

- **Frontend:** Developed using SwiftUI (for iOS)
- **Backend:** Firebase
- **Machine Learning:** Python-based recommendation engine (using scikit-learn or TensorFlow).
- **Database:** Firebase

1.1.2.3 Human Interfaces

- **User Interface:** Intuitive dashboard with progress bars, transaction summaries, and goal panels.
- **Chat Interface:** LLM-powered conversational interface for financial guidance.
- **Notifications:** Adaptive alerts (push/email) for spending habits and milestone updates.

1.2 Architectural Design

The architectural design of Finance Buddy emphasizes modularity, scalability, and data security. The system follows a client–server architecture with a microservices-inspired backend that separates financial logic, AI recommendations, and user management.

1.2.1 Major Software Components

1. **User Management Module**
   Handles account creation, authentication, and user profile data.

2. **Transaction Engine**
   Processes user-entered expenses, categorizes spending, and aggregates monthly summaries.

3. **Goal & Budget Manager**
   Allows users to set financial goals, visualize progress, and receive AI-driven adjustments.

4. **AI Recommendation System**
   Analyzes transaction history and provides tailored recommendations for saving, spending, and investment.

5. **Analytics Dashboard**
   Displays charts, progress widgets, and insights

6. **Notification System**
   Generates alerts for budget thresholds, missed goals, and milestone completions.

1.2.2 Major Software Interactions

- The Frontend UI communicates with the Spring Boot API using HTTPS and JSON payloads.

- The **Backend** interacts with the **PostgreSQL database** for persistent storage.

- The **AI Engine** (Python microservice) receives anonymized transaction data via API calls to generate predictions.

- The **Notification Service** subscribes to database events to push updates to users.

- The **Chat Interface** sends user queries to a lightweight LLM endpoint (e.g., OpenAI API or HuggingFace model).

1.2.3 Architectural Design Diagrams

1. Use Case Diagram

- Actors: User, AI Engine, Database.

- Use cases: “Track Expenses,” “Set Goal,” “View Recommendations,” “Receive Notifications.”

2. Top-Level Class Diagram

- Classes: `User`, `Transaction`, `Goal`, `Budget`, `Recommendation`, `Notification`, `AIModel`.
- Relationships:
    - `User` → has many → `Transactions`
    - `User` → has many → `Goals`
    - `AIModel` → generates → `Recommendations`

3. Component Diagram

- Frontend (React/SwiftUI) ↔ REST API (Spring Boot) ↔ Database (PostgreSQL)
- REST API ↔ AI Engine (Python microservice)
- REST API ↔ Notification Service (Kafka or Cloud Pub/Sub)

1.3 CSC and CSU Descriptions

The *Finance Buddy* application is a single **CSCI**, composed of multiple **CSCs**:

1. User Management CSC
2. Finance Engine CSC
3. AI Recommender CSC
4. Notification CSC
5. Frontend Interface CSC

Each CSC comprises several **CSUs** (classes and interfaces).

1.3.1 Class Descriptions

The following sections describe the classes used in *Finance Buddy*.

1.3.1.1 Class: User

- **Purpose:** Manages authentication and user profile data.
- **Fields:**
    - `id: UUID`
    - `username: String`
    - `email: String`
    - `hashedPassword: String`
    - `createdAt: DateTime`
- **Methods:**
    - `authenticate()`
    - `updateProfile()`
    - `deleteAccount()`

1.3.1.2 Class: Transaction

- **Purpose:** Represents an expense or income record.
- **Fields:**
    - `id: UUID`
    - `userId: UUID`
    - `category: String`
    - `amount: Float`
    - `timestamp: DateTime`
- **Methods:**

- `categorize()`
- `summarizeByMonth()`

1.3.1.3 Class: Goal

- **Purpose:** Stores and manages user-defined financial goals.
- **Fields:**
  - `goalName: String`
  - `targetAmount: Float`
  - `currentProgress: Float`
- **Methods:**
  - `updateProgress()`
  - `evaluateGoalStatus()`

1.3.1.4 Class: Recommendation

- **Purpose:** Stores AI-generated insights.
- **Fields:**
  - `message: String`
  - `category: String`
  - `priority: Int`
- **Methods:**
  - `generateForUser(userId)`
  - `displayRecommendation()`

1.3.2 Detailed Interface Descriptions

- **REST Endpoints (Spring Boot):**
    - `POST /api/user/register`
    - `GET /api/user/{id}/transactions`
    - `POST /api/transaction/add`
    - `GET /api/ai/recommendations`
- **AI Microservice Interface:**
    - Input: User's recent transactions (JSON).
    - Output: Personalized tips and spending classifications (JSON).

1.3.3 Detailed Data Structure Descriptions

- **Transaction Table (PostgreSQL):**

Column	Type	Description
id	UUID	Primary key
user_id	UUID	Foreign key
category	TEXT	Expense category
amount	FLOAT	Transaction amount
timestamp	TIMESTAMP	Date/time of transaction

Recommendation JSON Format:

```
 {
  "userId": "123",
  "advice": "Reduce dining out by 10% to save $120/month",
  "priority": "medium"
}
```

1.3.4 Detailed Design Diagrams

- **Sequence Diagram:** User submits transaction → API logs transaction → AI model analyzes spending → Recommendation returned → Displayed on UI.
- **Activity Diagram:** Data flow for “Goal Progress Update.”
- **Class Diagram:** All core data and service relationships.

1.4 Database Design and Description

1.4.1 Database Design ER Diagram

Entities: `User`, `Transaction`, `Goal`, `Recommendation`
Relationships:

- `User` 1–* `Transaction`
- `User` 1–* `Goal`
- `AIModel` 1–* `Recommendation`

1.4.2 Database Access

Accessed through a Spring Boot ORM layer using **JPA** repositories with **Hibernate**. All queries are parameterized to prevent SQL injection.

1.4.3 Database Security

- Encrypted credentials via AES encryption.
- Role-based access control.
- Periodic backups and anonymized training data exports.