Team Assignment: Architecture and Design

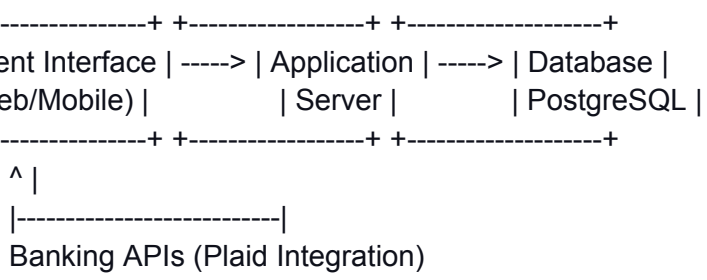
Architecture

Our Personal Finance Management App is designed as a cross-platform solution, using a three-tier architecture that includes a client layer, an application server, and a database layer.

Client Layer: This includes mobile and web-based interfaces that allow users to interact with the app. It handles all user inputs and displays data visualization charts and graphs.

Application Server: The server handles business logic, processes user requests, and communicates with the database. It categorizes transactions, calculates budgets, and provides insights into spending habits.

Database Layer: A secure relational database stores user information, transaction data, and budget plans. Additional external integrations, such as banking APIs like Plaid, allow transaction data to be imported automatically.



Architectural Style

Model-View-Controller (MVC): The app is organized using the MVC pattern to separate concerns. The client layer serves as the View, the server contains the Controller logic, and the database handles the Model layer.

Logical Components

System Components

User Management: Handles user authentication, profile settings, and preferences

Transaction Manager: Categorizes transactions, calculates income/expenses, and tracks spending

Budgeting Module: Provides personalized budget insights and savings recommendations

Bank Integration: Facilitates secure communication with external APIs like Plaid for transaction imports

Data Visualization: Generates charts and graphs for spending patterns and financial goals

Database Manager: Manages data storage, retrieval, and schema updates

Design

Key Classes and Functions

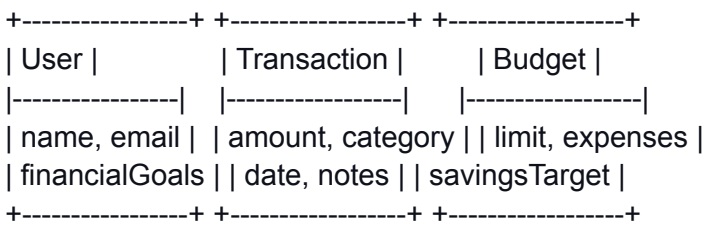
User Class: Represents the user, storing details such as name, email, and financial goals

Transaction Class: Manages individual transactions with properties for category, amount, date, and description

Budget Class: Calculates and tracks user-defined budgets

APIController: Handles requests from the client and communicates with external APIs.

Class Diagram



Database Layout
The app uses a relational schema to store data:

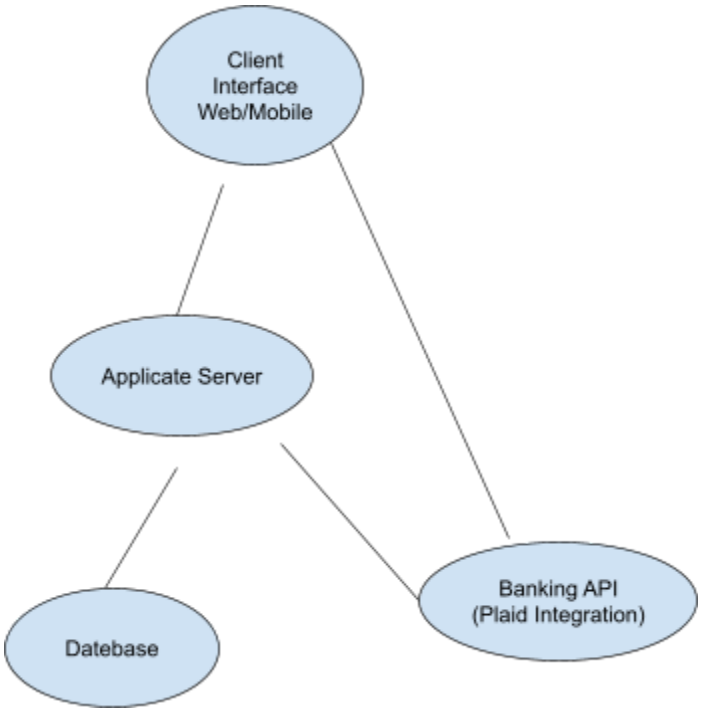
Table	Columns
Users	name, email, hashed_password
Transactions	amount, date, category, description
Budgets	limit, savings_goal, current_spending

Non-Obvious Design Decisions

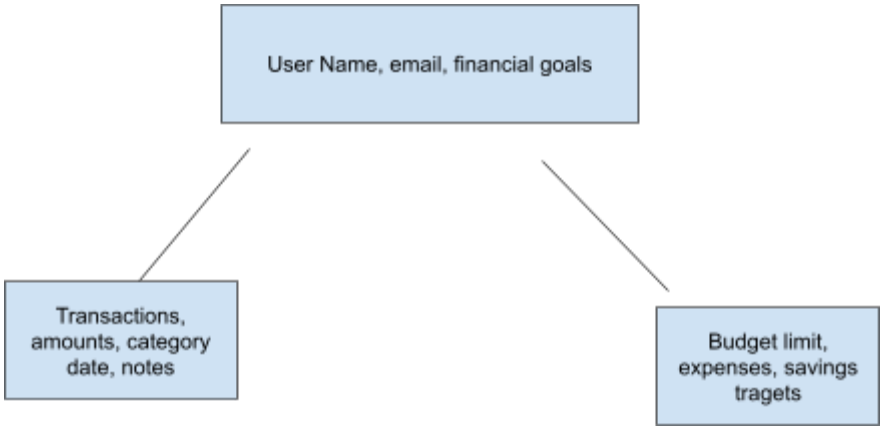
Transaction Categorization: Transactions are categorized based on keywords and patterns in descriptions. Future developers must ensure new keywords align with the existing structure.

Bank Integration Logic: The integration with Plaid must adhere to security protocols. API tokens are stored securely

Architecture Diagram



UML Diagram for Logical Components



Database Schema Diagram