Cashflow - Project 1

Nguyen Dang Minh Quan - 210257

March 1, 2025

Contents

1 Requirements

1.1 Users and Use Cases

1.1.1 Target Users

The **Cashflow** application is designed for students who study far from home. The app is an AI-powered budget management tool to help these students manage their expenses efficiently.

1.1.2 User Role

- **Students:** The only user role in the current system.

1.2 Functionalities

1.2.1 User Account Management

- **Registration:** Users can register for an account using the `/auth/signup` endpoint (HTTP POST).
- **Login:** Users log in using the `/auth/login` endpoint (HTTP POST).
- **Profile Retrieval:** Authenticated users can view their profile details via the `/users/me` endpoint (HTTP GET).

1.2.2 Expense Management

- **Manual Expense Entry:** Users can manually add an expense using the `/transactions/save` endpoint (HTTP POST).
- **View Transactions:** Users can retrieve a list of all their past transactions using the `/transactions` endpoint (HTTP GET).

1.2.3 Receipt Scanning and OCR

- **Receipt Scanning:** Users can scan receipts by taking a picture or selecting an image from their photo gallery.
- **OCR Processing:** The receipt image is processed asynchronously via the `/ocr/scan` endpoint (HTTP POST). The process involves:
  - Using the local Google Tesseract OCR engine to extract text.
  - Sending the OCR output to the Google Gemini API for further analysis and data extraction.
  - Automatically creating a transaction based on the extracted information.

1.2.4 Category Management

- **View Categories:** Users can view a list of default categories via the `/categories` endpoint (HTTP GET).
- **Default Categories:** Default categories have a `null` user ID. (A future feature will allow users to create new categories.) The `icon` and `color` fields are designed to be compatible with Flutter's Icon and Color modules.

2 Design

2.1 Architectural Design

2.1.1 Frontend Architecture

- **Design Pattern:** Model-View-Controller (MVC).
- **Components:**
  - **Model:** Defines the data structures and business logic in Flutter.
  - **View:** Consists of UI components, screens, and layouts using Flutter widgets.
  - **Controller:** Handles user interactions and communicates with backend RESTful API endpoints.

2.1.2 Backend Architecture

- **Current Architecture:** A monolithic backend built with Java Spring Boot.
- **Planned Evolution:** The architecture will be refactored into microservices once the project scales.
- **Communication:** RESTful API endpoints are used with JSON payloads.
- **Key API Endpoints:**
  - `/auth/signup` (POST) – User registration.
  - `/auth/login` (POST) – User login.
  - `/users/me` (GET) – Retrieve the logged-in user's information.
  - `/ocr/scan` (POST) – Submit receipt images for OCR processing.
  - `/transactions` (GET) – Retrieve all transactions for the user.
  - `/transactions/save` (POST) – Save a new transaction.
  - `/categories` (GET) – Retrieve default categories.
- **Asynchronous Processing:** The receipt scanning process is handled asynchronously, enqueuing a job for OCR processing and subsequent transaction creation.

2.2 Database Design

2.2.1 Database Engine

PostgreSQL is used as the primary database engine.

2.2.2 Entity-Relationship Diagram (ERD)

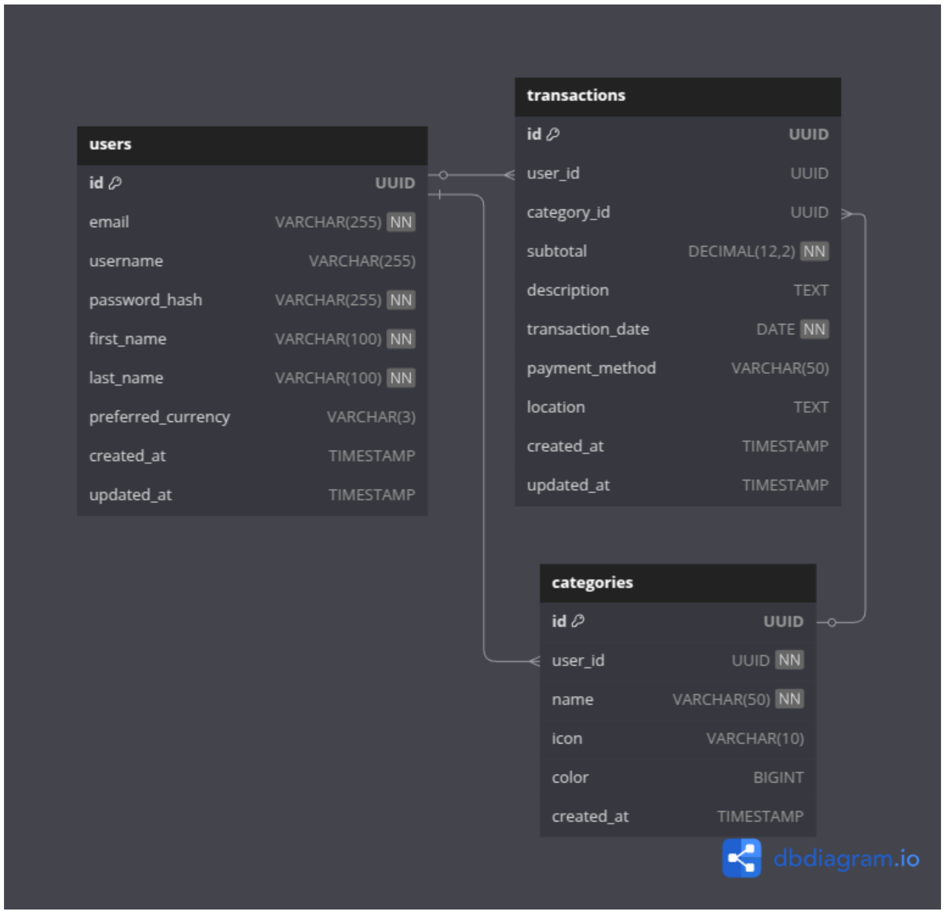

Figure 1: ERD for Cashflow

3 API Endpoints Summary

Since the table of all API Endpoints is too large to be included in this report, I have attached a link to a Google Sheets document below which contains all the API endpoints as well as how to test them. H

https://docs.google.com/spreadsheets/d/1LQVeCh6bFJSSYUjLgNCkAdbC0AY8ioNvFqb1Uigei28/edit?usp=sharing

4 Set up application

4.1 Requirements

1. Java 17
2. Apache Maven 3.6.3
3. PostgreSQL
4. Flutter SDK

4.2 Setting up

1. Initialize database
   - Run local PostgreSQL server
   - create table **cashflow_dev** and run **CREATE EXTENSION IF NOT EXISTS "uuid-ossp";** as super user
2. Run the backend server
   - cd backend/cashflow
   - Create .env file and include your credentials:

     ```
     DATABASE_URL=jdbc:postgresql://localhost:5432/cashflow_dev
     DATABASE_USERNAME=
     DATABASE_PASSWORD=

     JWT_SECRET_KEY=
     GEMINI_API_KEY=
     ```

   - Modify the username and
   - ./mvnw spring-boot:run
3. Run frontend
   - cd frontend/cashflow
   - Create .env file and include

     ```
     API_BASE_URL={Your BE base URL}
     ```

   - flutter run
   - Choose you device (mobile only)

5 Visualization and Screens

All screens and visualization of user flow is included in a demo video. The link is attached below.

https://youtu.be/qKEDsLBkbj0

6 Codebase

I have attached the link to this project repository below.

https://github.com/siddankthep/cashflow