ĐẠI HỌC BÁCH KHOA HÀ NỘI

———————— * ————————

BÁO CÁO PROJECT I – IT3150

ĐỀ TÀI: HỆ THỐNG QUẢN LÝ TIỀN THÔNG MINH

Họ và tên sinh viên: Trần Đức Mạnh

Mã số sinh viên: **20235156**

Ngành: Khoa học máy tính

Email: Manh.TD235156@sis.hust.edu.vn

Giảng viên hướng dẫn: B.S. Nguyễn Duy Tùng

Hà Nội, 2026

Mục lục

1 Giới thiệu đề tài

1.1 Bối cảnh

Các công cụ quản lý tài chính truyền thống như bảng tính hoặc các ứng dụng di động hiện có thường đòi hỏi người dùng phải nhập liệu thủ công một cách tẻ nhạt và thiếu tính định hướng. Sự thiếu trực quan trong việc theo dõi tiến độ và thiếu các gợi ý hành động cụ thể khiến người dùng dễ mất động lực và từ bỏ sau một thời gian ngắn. Điều này dẫn đến tình trạng chi tiêu không kiểm soát và khó khăn trong việc đạt được các mục tiêu tài chính dài hạn. Nhận thấy khoảng trống này, đề tài được đề xuất với mục tiêu giải quyết triệt để những nhược điểm của các phương pháp truyền thống.

1.2 Mục tiêu của đề tài

1.2.1 Mục tiêu tổng quát

Phát triển một ứng dụng Web hỗ trợ quản lý tài chính cá nhân, giúp người dùng theo dõi chi tiêu, lập ngân sách và đạt được các mục tiêu tiết kiệm một cách hiệu quả và thông minh.

1.2.2 Mục tiêu cụ thể

- Xây dựng hệ thống backend Node.js và frontend React.
- Triển khai các chức năng CRUD (Create, Read, Update, Delete) cho các mục: Giao dịch (Transaction), Ngân sách (Budget), và Mục tiêu (Goal).
- Tích hợp phương thức nhập liệu thông minh thông qua việc phân tích hình ảnh hóa đơn, giảm thiểu tối đa thao tác nhập liệu thủ công.
- Triển khai một thuật toán lập kế hoạch tiết kiệm, có tính giải thích được, có khả năng tạo ra kế hoạch tiết kiệm cá nhân hóa dựa trên dữ liệu thực tế của người dùng.

2 Thông tin mã nguồn

Toàn bộ mã nguồn của dự án được quản lý tập trung trên nền tảng GitHub, cho phép theo dõi lịch sử thay đổi và hỗ trợ triển khai nhanh chóng trên các môi trường máy tính khác nhau.

2.1 Thông tin Repository

- **Tên dự án:** SmartMoney - Hệ thống quản lý tiền thông minh
- **Link:** https://github.com/ducmanhhb2005/Smart_Money
- **Cấu trúc mã nguồn:** Dự án được chia thành hai thư mục chính: `/backend` (Node.js/Express) và `/frontend` (React.js/Vite).

2.2 Yêu cầu hệ thống

Để cài đặt và chạy dự án, máy tính cần được cài đặt sẵn các công cụ sau:

- **Node.js:** Phiên bản v16.0 trở lên.
- **MySQL Server:** Phiên bản 8.0 trở lên.
- **Trình quản lý gói:** npm hoặc yarn.
- **Git:** Để thực hiện sao chép kho lưu trữ.

2.3 Quy trình cài đặt và triển khai

2.3.1 Bước 1: Sao chép mã nguồn

Sử dụng lệnh Git để tải mã nguồn về máy tính cá nhân:

```
git clone https://github.com/ducmanhhb2005/Smart_Money.git
cd Smart_Money
```

2.3.2 Bước 2: Cấu hình môi trường

Người dùng cần tạo tệp `.env` trong thư mục `/backend` để thiết lập các thông số kết nối:

```
DATABASE_URL="mysql://root:password@localhost:3306/smartmoney"
JWT_SECRET="YOUR_SECRET_KEY"
GEMINI_API_KEY="YOUR_GOOGLE_GEMINI_API_KEY"
PORT=5000
```

2.3.3 Bước 3: Cài đặt thư viện và khởi tạo cơ sở dữ liệu

Thực hiện cài đặt các gói phụ thuộc và sử dụng Prisma để tạo cấu trúc bảng trong MySQL:

a. Cài đặt cho Backend

```
cd backend
npm install
npx prisma migrate dev - -name init
```

b. Cài đặt cho Frontend

```
cd ../frontend
npm install
```

2.3.4 Bước 4: Khởi chạy ứng dụng

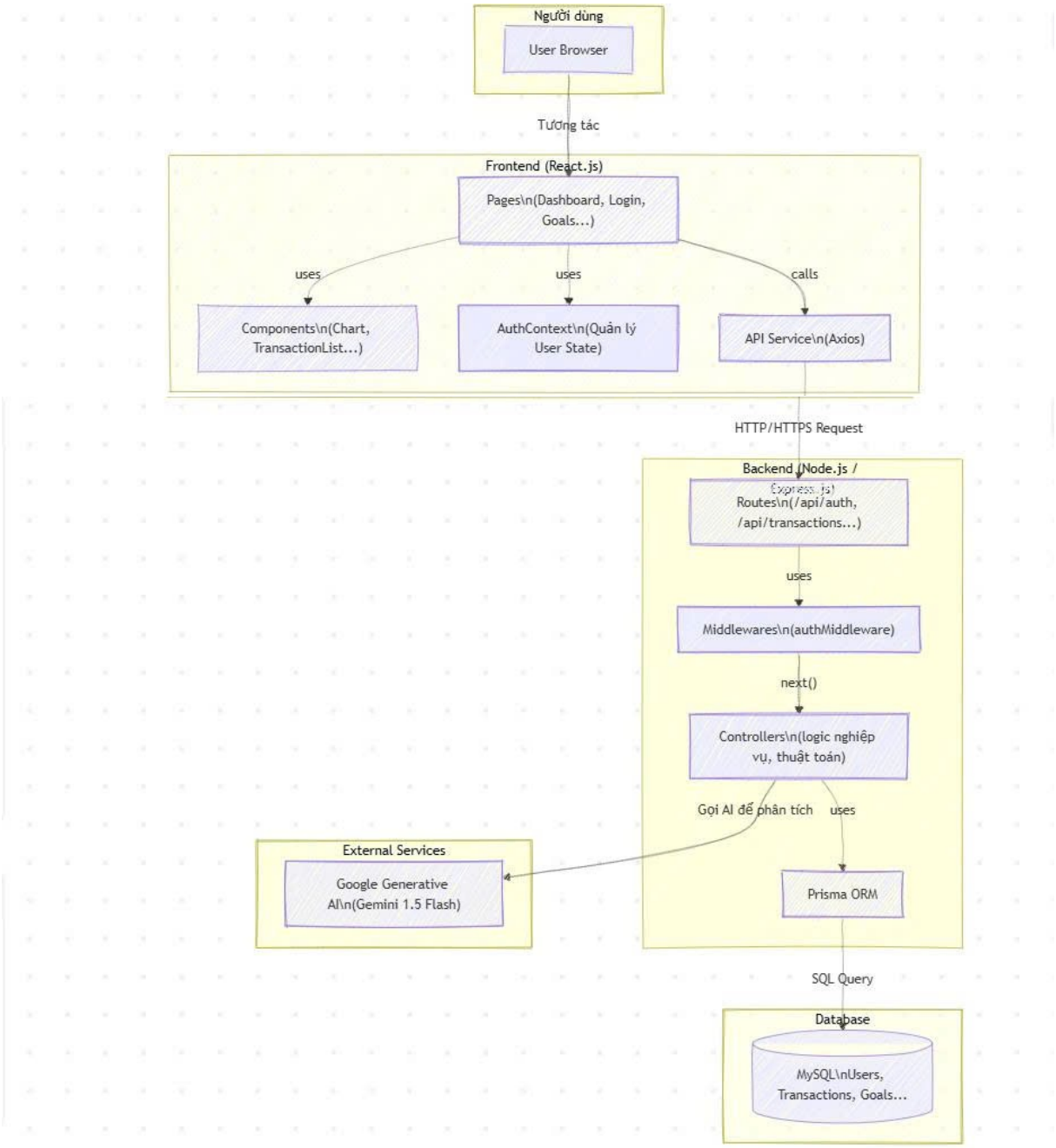
Để chạy toàn bộ hệ thống, cần mở hai cửa sổ terminal riêng biệt cho Backend và Frontend:

- **Tại thư mục /backend:** Chạy lệnh `npm run dev`. Server sẽ lắng nghe tại http://localhost:5000.
- **Tại thư mục /frontend:** Chạy lệnh `npm run dev`. Ứng dụng sẽ khả dụng tại http://localhost:5173.

3 Kiến trúc hệ thống và công nghệ sử dụng

3.1 Kiến trúc tổng quan

Hệ thống được xây dựng theo kiến trúc Client-Server. Phía client là một ứng dụng Single Page Application (SPA) viết bằng React, chịu trách nhiệm render giao diện và tương tác với người dùng. Phía server là một hệ thống API RESTful viết bằng Node.js và Express, có nhiệm vụ xử lý toàn bộ logic nghiệp vụ, xác thực người dùng và tương tác với cơ sở dữ liệu. Giao tiếp giữa client và server được thực hiện qua giao thức HTTP.



Hình 1: Sơ đồ kiến trúc tổng quan của hệ thống SmartMoney.

3.1.1 Thành phần Frontend

Phần Frontend được xây dựng dưới dạng một **Single Page Application (SPA)** sử dụng thư viện **React.js**. Đây là lớp chịu trách nhiệm hoàn toàn cho việc hiển thị giao diện và tương tác với người dùng. Các thành phần chính bao gồm:

- **Pages:** Các component lớn đại diện cho từng trang màn hình của ứng dụng: `RegisterPage`, `LoginPage`, `DashboardPage`, `AddTransactionPage`, `AddByReceiptPage`, `BudgetsPage`, `GoalsPage`.
- **Components:** Các thành phần giao diện nhỏ, có khả năng tái sử dụng cao, được kết hợp để xây dựng Dashboard: `Chart`, `TransactionList`, `SummaryCard`.
- **AuthContext:** Sử dụng React Context API để quản lý trạng thái xác thực (đăng nhập/đăng xuất) và thông tin người dùng trên toàn bộ ứng dụng.
- **API Service:** Một module trung gian sử dụng thư viện **Axios** để thực hiện các lời gọi API đến Backend. Module này đóng gói logic giao tiếp mạng, bao gồm cả việc tự động đính kèm token xác thực vào header của request.

3.1.2 Thành phần Backend

Phần Backend được xây dựng trên nền tảng **Node.js** và sử dụng framework **Express.js** để tạo ra một hệ thống **API RESTful**. Đây là khu xử lý toàn bộ logic nghiệp vụ và bảo mật.

- **Routes:** Định nghĩa các điểm cuối (endpoints) của API, ví dụ: `/api/auth`, `/api/transactions`. Thành phần này chịu trách nhiệm điều hướng các request đến đúng controller xử lý.
- **Middlewares:** Các hàm trung gian được thực thi trước khi request đến được controller. Middleware quan trọng nhất là `authMiddleware`, có nhiệm vụ xác thực JWT để bảo vệ các tài nguyên yêu cầu đăng nhập.
- **Controllers:** Nơi chứa toàn bộ logic nghiệp vụ và các thuật toán của ứng dụng. Controllers nhận request đã được xác thực, xử lý dữ liệu và trả về response cho client.
- **Prisma ORM:** Prisma cho phép lập trình viên tương tác với cơ sở dữ liệu bằng các đối tượng và phương thức JavaScript/TypeScript thay vì viết các câu lệnh SQL thuần, giúp tăng cường an toàn kiểu dữ liệu và năng suất.

Chi tiết API của backend: Sheet API

3.1.3 Các dịch vụ và lưu trữ

- **Dịch vụ ngoài:** Hệ thống tích hợp dịch vụ **Google AI Studio (Gemini 2.5 Flash)** để thực hiện các tác vụ thông minh. Backend sẽ gọi đến dịch vụ này để phân tích hình ảnh hóa đơn (kết hợp OCR và phân tích ngữ nghĩa).
- **Database:** Hệ thống sử dụng **MySQL** làm hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) để lưu trữ bền vững toàn bộ dữ liệu của ứng dụng, bao gồm `Users`, `Transactions`, `Budgets`, và `Goals`.

3.1.4 Luồng giao tiếp

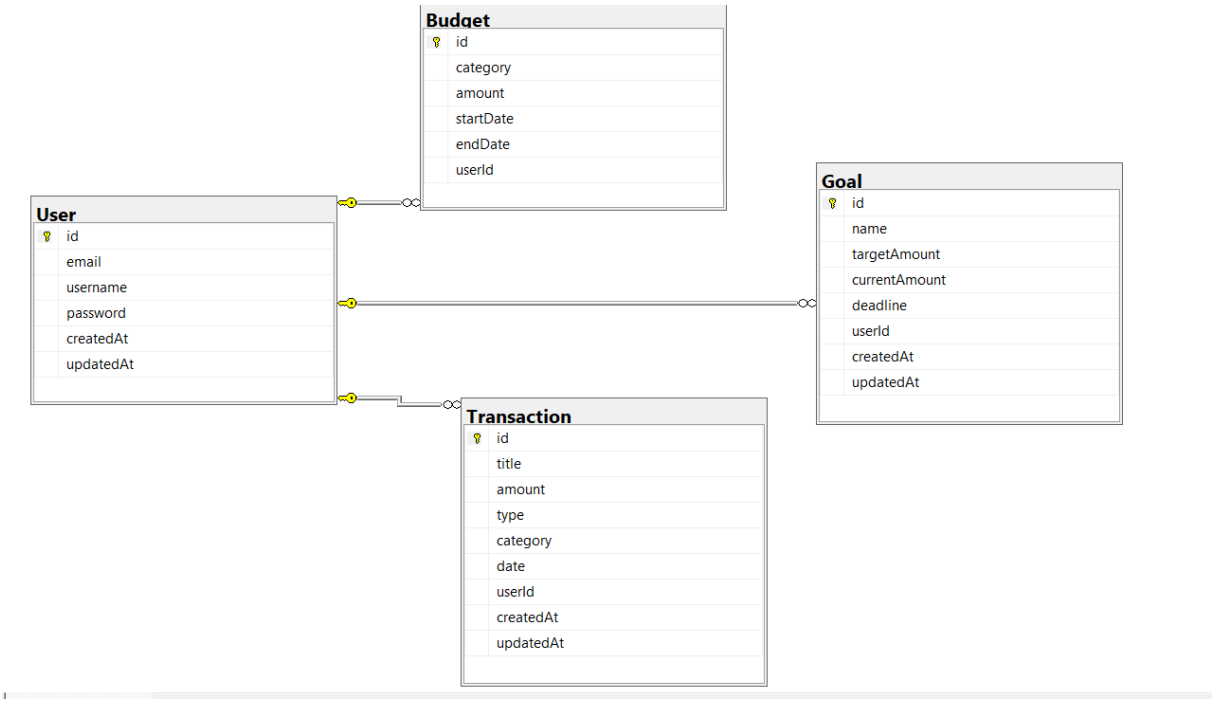
Luồng giao tiếp chính giữa Client và Server được thực hiện qua giao thức **HTTP/HTTPS**. Frontend gửi các request (ví dụ: `POST /api/transactions`) đến Backend. Backend xử lý và trả về các response dưới dạng JSON (ví dụ: `{status: 201, data: ...}`). Prisma ORM giao tiếp với cơ sở dữ liệu MySQL.

3.2 Công nghệ sử dụng

- **Backend:**
  - **Nền tảng:** Node.js
  - **Framework:** Express.js
  - **ORM:** Prisma
  - **Cơ sở dữ liệu:** MySQL
- **Frontend:**
  - **Thư viện:** React
  - **Build tool:** Vite
  - **Styling:** CSS Modules
  - **Routing:** React Router DOM
- **Xác thực:** JSON Web Tokens (JWT).
- **Dịch vụ AI:** Google AI Studio (Gemini 2.5 Flash).

4 Thiết kế Cơ sở dữ liệu

4.1 Sơ đồ Quan hệ Thực thể (ERD)



Hình 2: Sơ đồ quan hệ thực thể của CSDL SmartMoney.

4.2 Chi tiết Schema (Prisma)

Mô hình dữ liệu được định nghĩa và quản lý tập trung thông qua file `schema.prisma`. Cách tiếp cận Schema-first này giúp đảm bảo an toàn về kiểu dữ liệu, định nghĩa rõ ràng các mối quan hệ và tự động sinh ra một client tối ưu cho việc truy vấn cơ sở dữ liệu. Dưới đây là các model chính đã được triển khai:

Listing 1: Định nghĩa các Model chính trong file schema.prisma

```
% User
model User {
  id          Int       @id @default(autoincrement())
  email       String    @unique
  username    String    @unique
  password    String % hashed password

  % 1-n relationship
  transactions Transaction[]
  budgets      Budget[]
  goals        Goal[]

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}
```

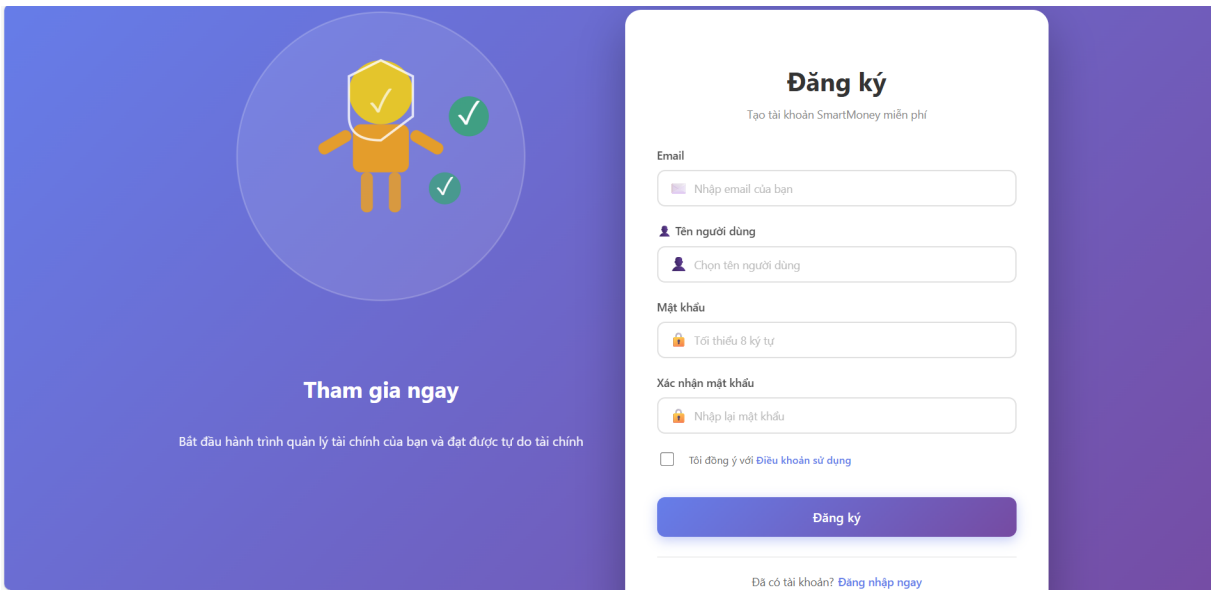
```
model Transaction {
  id          Int       @id @default(autoincrement())
  title       String
  amount      Float
  type        TransactionType % ENUM: INCOME | EXPENSE
  category    String
  date        DateTime

  % FK and relationship with User
  userId      Int
  user        User      @relation(fields: [userId], references: [id])

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}


model Budget {
  id          Int       @id @default(autoincrement())
  category    String
  amount      Float
  startDate DateTime
  endDate   DateTime

  % FK and relationship with User
  userId      Int
  user        User      @relation(fields: [userId], references: [id])
}


model Goal {
  id              Int       @id @default(autoincrement())
  name            String
  targetAmount    Float
  currentAmount Float      @default(0)
  deadline        DateTime?

 % FK and relationship with User
  userId          Int
  user            User      @relation(fields: [userId], references: [id])

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}


enum TransactionType {
```

INCOME
EXPENSE
}

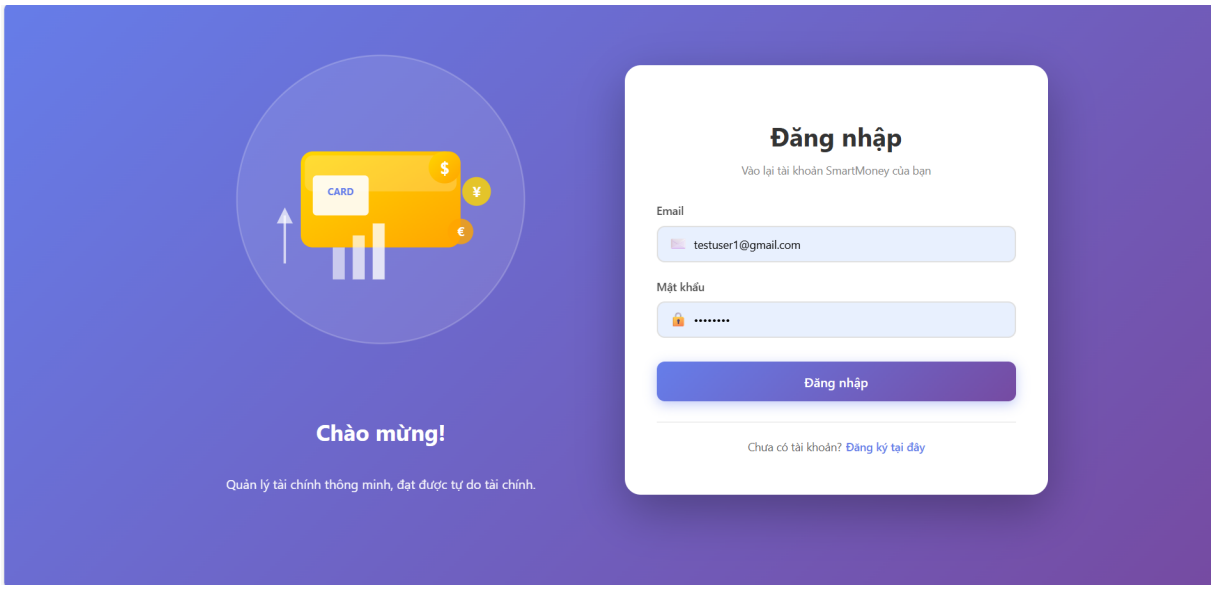
Các mối quan hệ một-nhiều (one-to-many) được định nghĩa rõ ràng, ví dụ một `User` có thể có nhiều `Transaction`. Khóa ngoại `userId` trong các model con kết hợp với chỉ thị `@relation` để Prisma hiểu và tạo ra các API truy vấn lồng nhau (nested queries) một cách hiệu quả, thay thế cho các câu lệnh JOIN phức tạp trong SQL truyền thống.

5 Giao diện

5.1 Trang đăng nhập, đăng ký

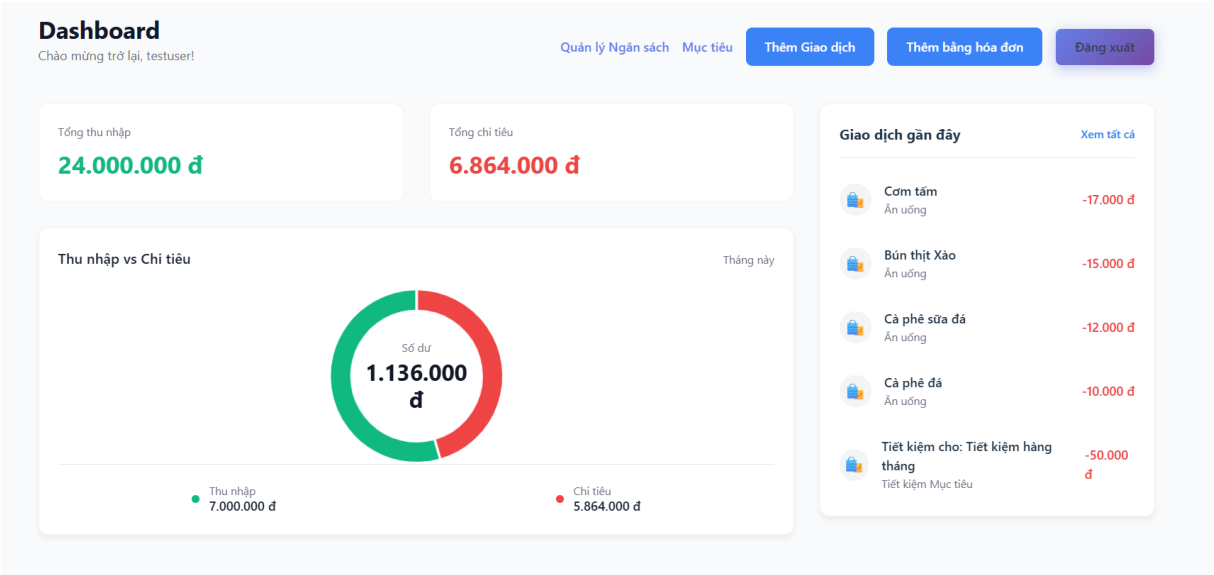


Hình 3: Trang đăng ký tài khoản



Hình 4: Trang đăng nhập tài khoản

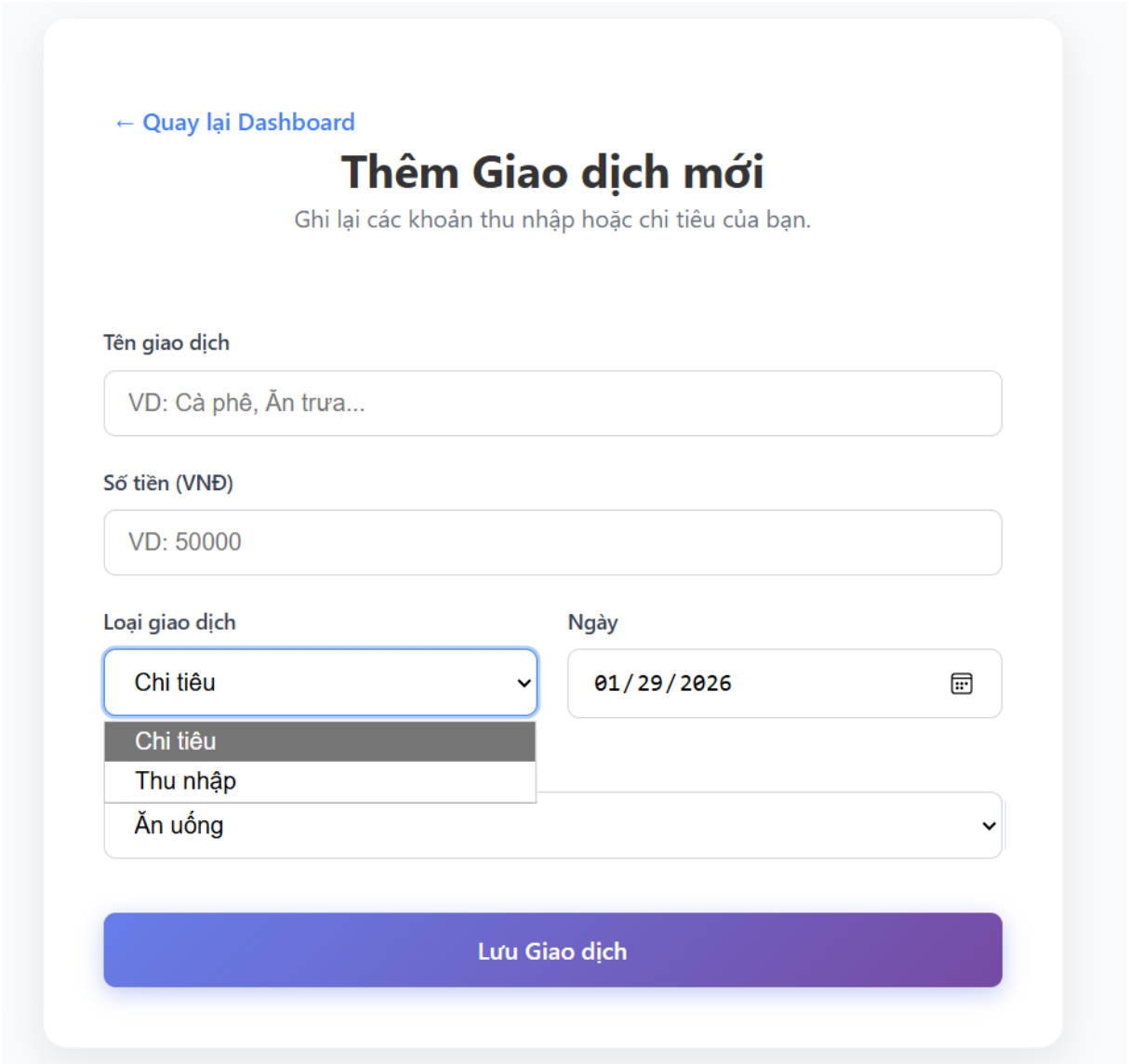
5.2 Trang Dashboard



Hình 5: Dashboard của web

- Summary Card: Hiển thị tổng toàn bộ giao dịch (thu nhập + chi tiêu) của người dùng từ trước đến nay.
- Chart: Hiển thị tổng toàn bộ giao dịch (thu nhập + chi tiêu) của người dùng trong mỗi tháng hiện tại.
- Các ô chức năng để người dùng có thể sử dụng nằm ở phía trên bên phải.
- Bảng tổng hợp các giao dịch gần đây để người dùng có thể biết các giao dịch gần đây.

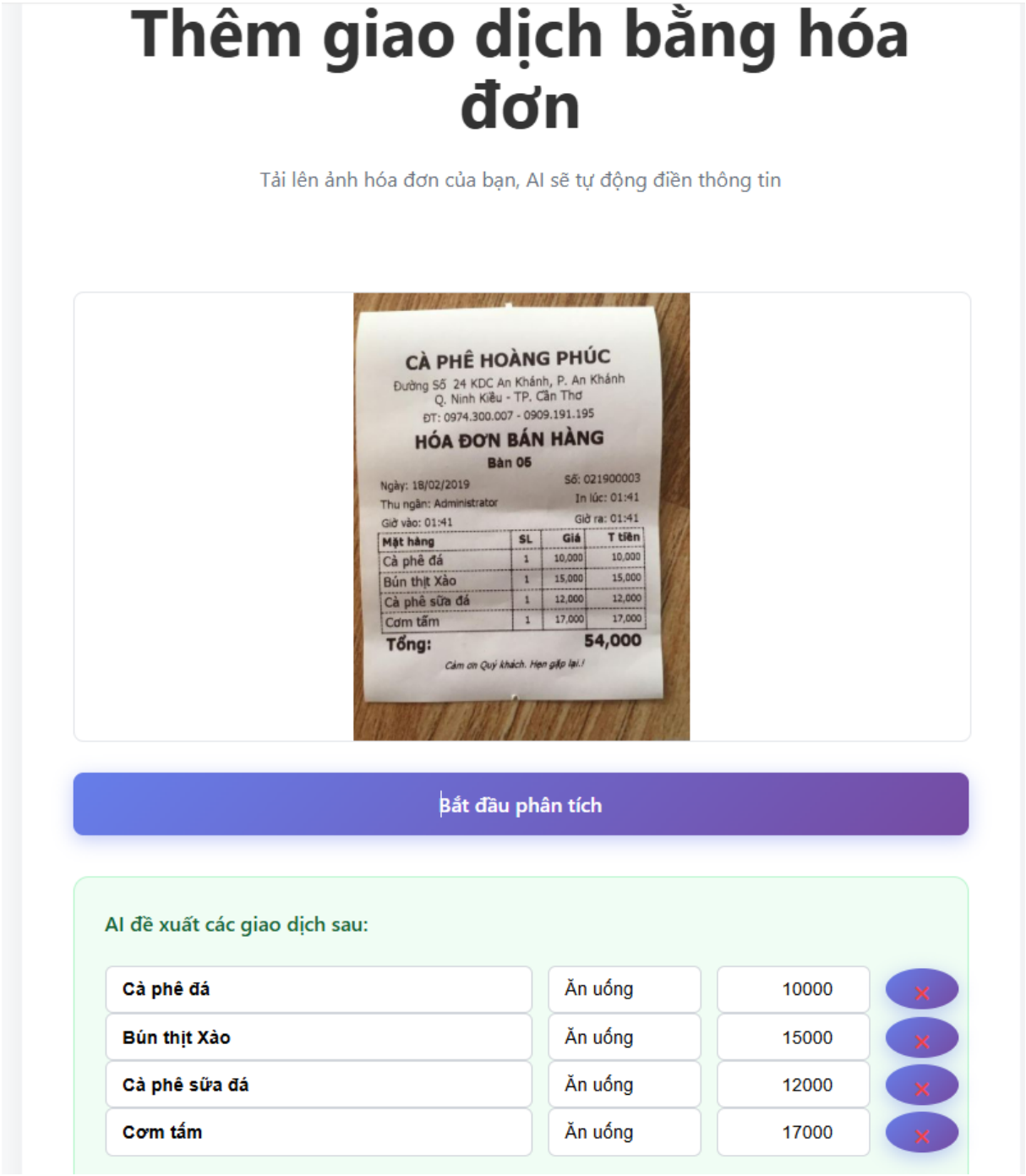
5.3 Biểu mẫu thêm các giao dịch (Thu nhập + Chi tiêu)



Hình 6: Biểu mẫu thêm giao dịch

- Người dùng nhập tên giao dịch cùng số tiền chi/nhận.
- Chọn một trong hai loại giao dịch: Chi tiêu, Thu nhập - Chọn ngày thực hiện giao dịch.
- Chọn một trong các hạng mục sau nếu là Chi tiêu: Ăn uống, Hóa đơn, Di chuyển, Mua sắm, Giải trí, Sức khỏe, Giáo dục, Gia đình, Quà tặng & Từ thiện, Khác.

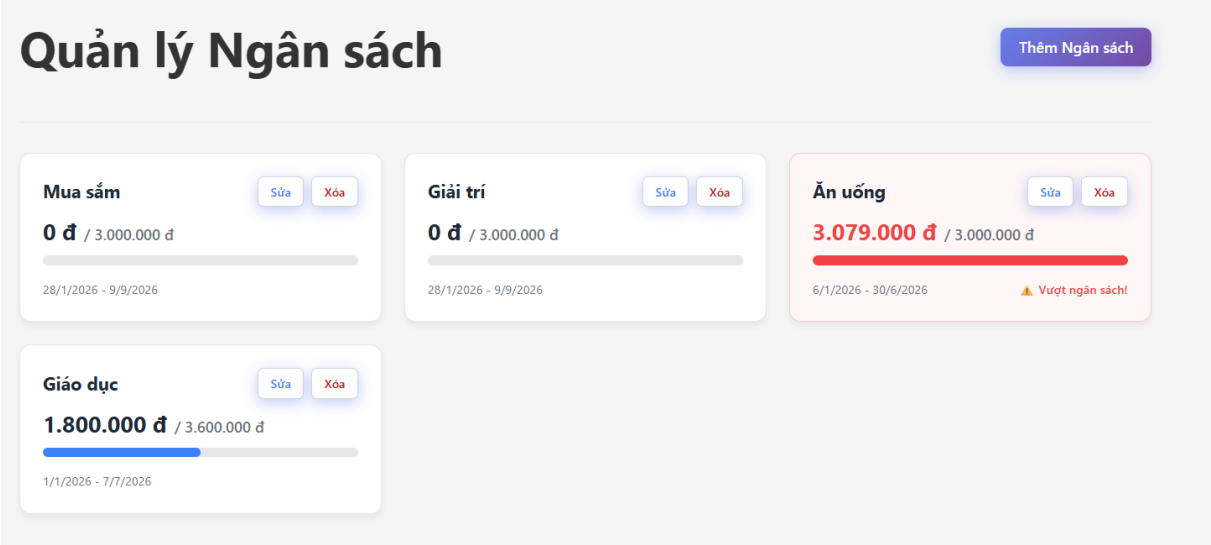
5.4 Nhập liệu thông minh bằng hóa đơn



Hình 7: Giao diện nhập liệu thông minh bằng hóa đơn

- Người dùng tải hóa đơn lên và chọn **Bắt đầu phân tích**
- AI làm việc và trích xuất ra các giao dịch có trong hóa đơn.
- Người dùng kiểm tra, có thể sửa để khớp với tình hình thực tế nếu AI trích xuất chưa chính xác.
- Cuối cùng, người dùng duyệt và tất cả giao dịch này sẽ vào hệ thống.

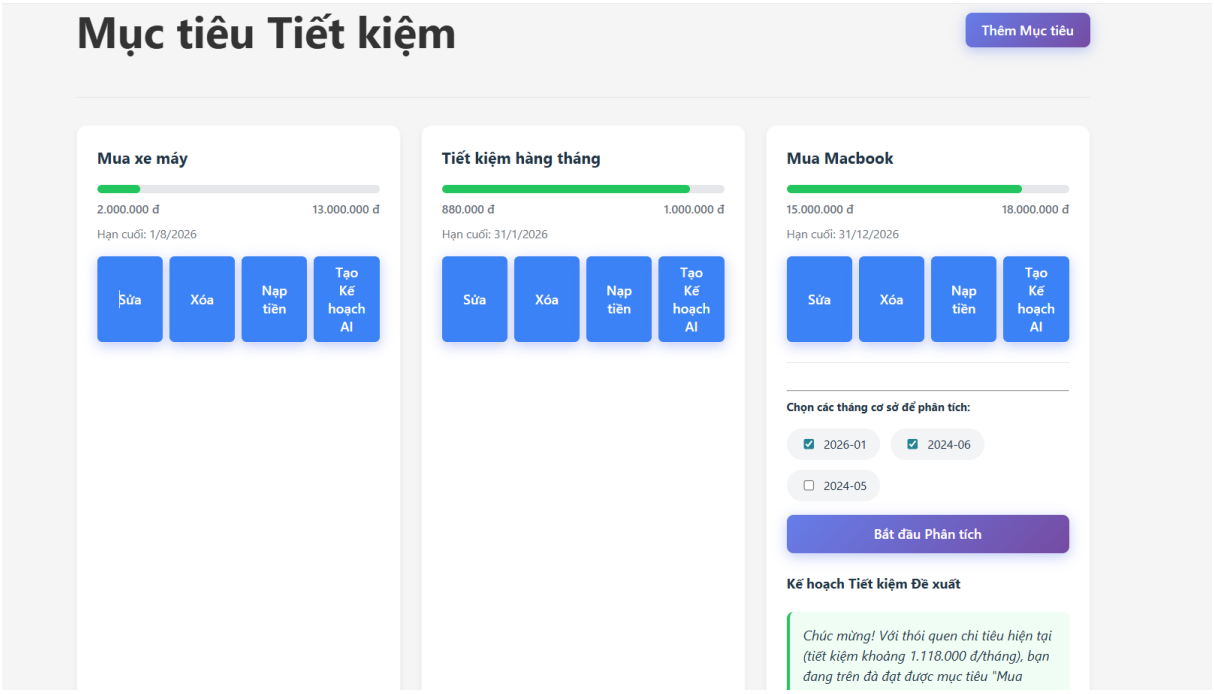
5.5 Quản lý ngân sách



Hình 8: Giao diện ngân sách

- Ngân sách để thực hiện kiểm soát chi tiêu người dùng không vượt ngưỡng cho phép trong khoảng thời gian đối với một trong các danh mục sau: Ăn uống, Hóa đơn, Di chuyển, Mua sắm, Giải trí, Sức khỏe, Giáo dục, Gia đình, Quà tặng & Từ thiện, Khác.
- Người có thể thêm, xóa, sửa các ngân sách.
- Mỗi khi người dùng chi tiêu trong khoảng thời gian đó thì số tiền tương ứng cho từng hạng mục sẽ được cập nhật.
- Nếu người dùng tiêu quá ngân sách cho một hạng mục thì hệ thống đưa ra cảnh báo.

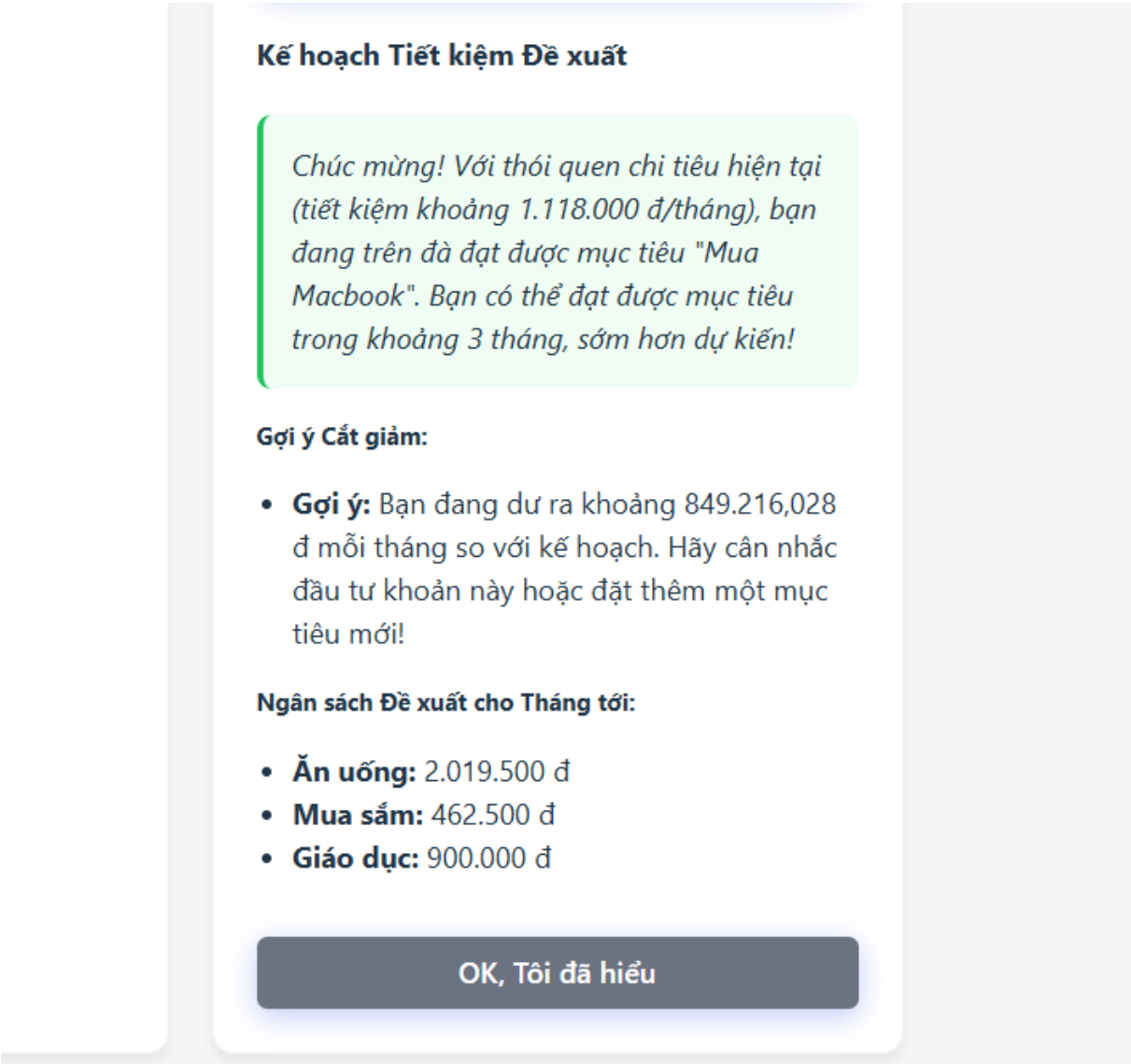
5.6 Quản lý mục tiêu



Hình 9: Quản lý mục tiêu

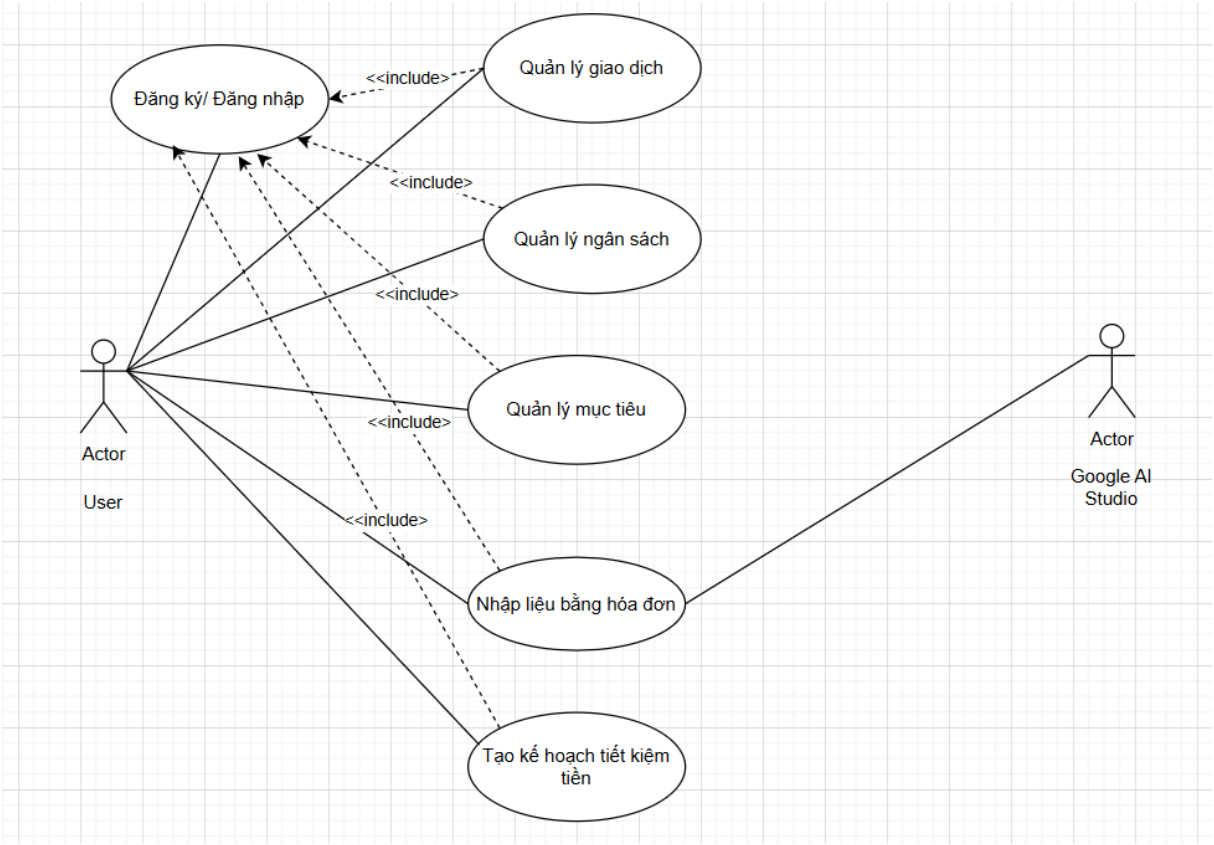
- Người dùng có thể thêm mục tiêu mới.
- Với mỗi mục tiêu:
+ Người dùng sửa, xóa hoặc nạp thêm tiền tiết kiệm vào mục tiêu đó. Khi nạp tiền tiết kiệm, một giao dịch đặc biệt có tên là "Tiết kiệm Mục tiêu"được tạo ra.

+ Tạo kế hoạch AI: Người dùng chọn các tháng cơ sở để hệ thống dựa vào đó tiến hành phân tích. Hệ thống trả về cho người dùng kết quả và đưa ra tư vấn về ngân sách người dùng giới hạn để chi tiêu trong một tháng tới.



Hình 10: Kết quả phân tích của hệ thống

6 Triển khai chức năng

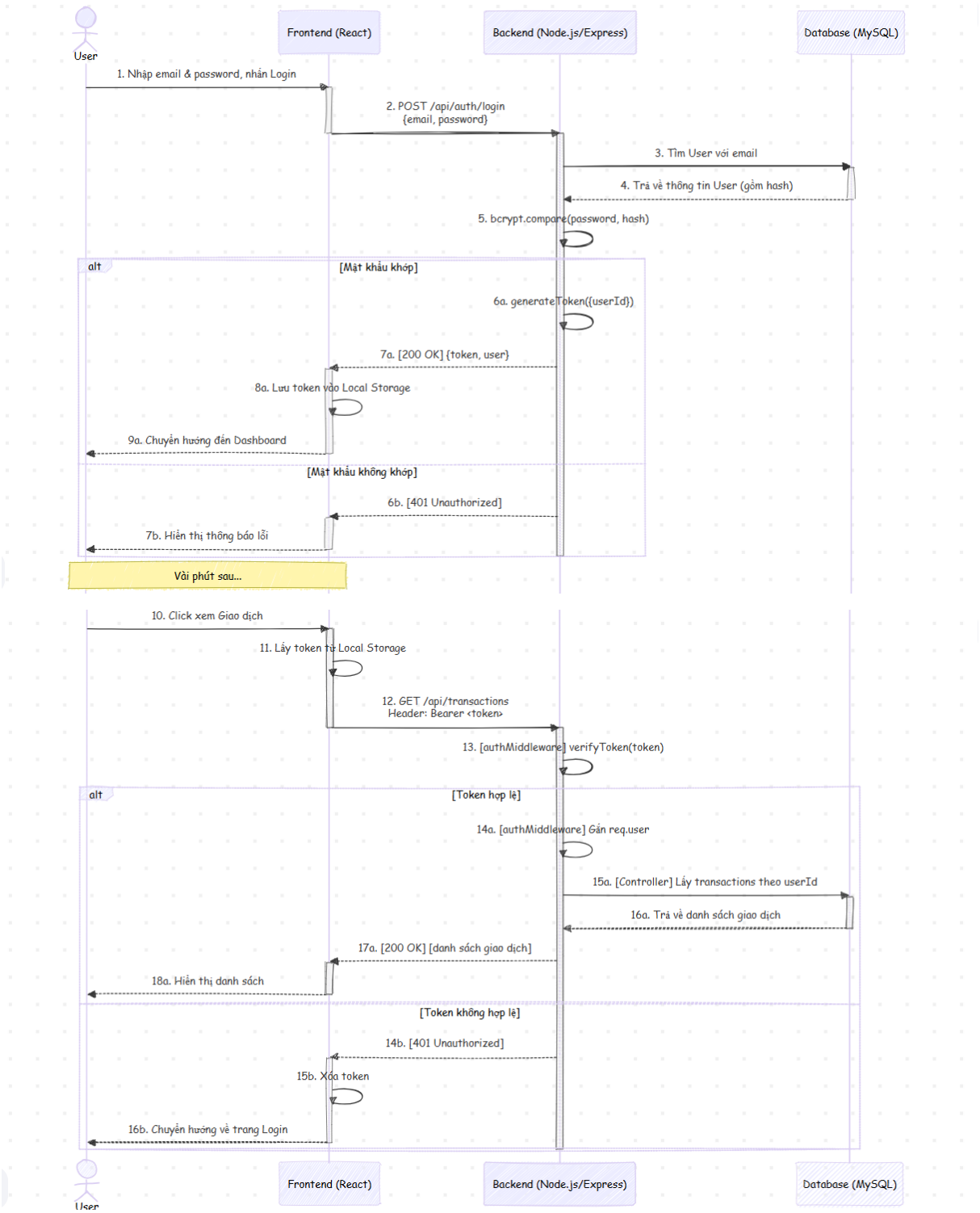


Hình 11: Biểu đồ Use Case mô tả chức năng toàn hệ thống

6.1 Xác thực người dùng

Hệ thống sử dụng phương pháp xác thực dựa trên token (JWT). Luồng hoạt động chi tiết:

1. Người dùng đăng ký tài khoản mới hoặc đăng nhập bằng email và mật khẩu.
2. Mật khẩu của người dùng được hash an toàn bằng bcrypt trước khi được lưu trữ hoặc so sánh trong cơ sở dữ liệu.
3. Sau khi đăng nhập thành công, server sẽ tạo ra một JWT. Token này chứa thông tin định danh (payload) như ‘userId‘ và được ký bằng một chuỗi bí mật (‘JWT_SECRET‘).
4. Token được trả về cho client. Client sẽ lưu trữ token này (ví dụ: trong Local Storage) và tự động đính kèm nó vào header ‘Authorization‘ (với tiền tố ‘Bearer‘) của mỗi request cần được bảo vệ.
5. Một middleware ‘authMiddleware‘ phía backend sẽ chặn các request này, giải mã token để xác thực người dùng và gắn thông tin người dùng vào đối tượng ‘req.user‘, sẵn sàng cho các controller xử lý.



Hình 12: Sequence Diagram cho xác thực người dùng và ví dụ minh hoa cho trường hợp người dùng muốn xem danh sách các giao dịch

6.2 Quản lý CRUD

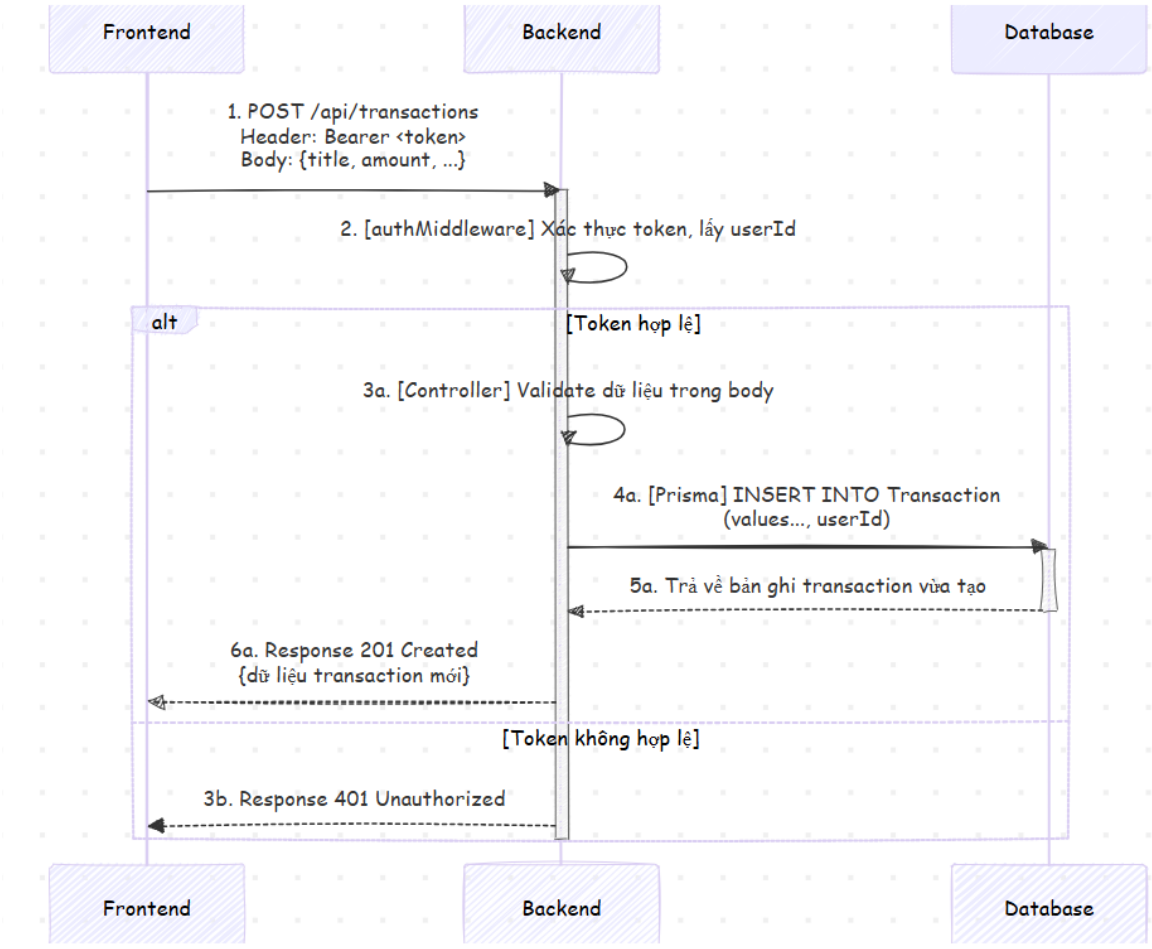
Các chức năng quản lý tài chính cốt lõi đã được triển khai dưới dạng các API RESTful đầy đủ, tuân thủ các nguyên tắc thiết kế hướng tài nguyên (resource-oriented design). Mỗi thực thể chính (Transaction, Budget, Goal) được xem như một tài nguyên và được

quản lý thông qua các phương thức HTTP tiêu chuẩn (GET, POST, PUT, DELETE). Tất cả các endpoint này đều được bảo vệ bởi middleware xác thực, đảm bảo người dùng chỉ có thể truy cập và thao tác trên dữ liệu của chính mình.

6.2.1 Tài nguyên Giao dịch (Transaction Resource) - Endpoint: `/api/transactions`

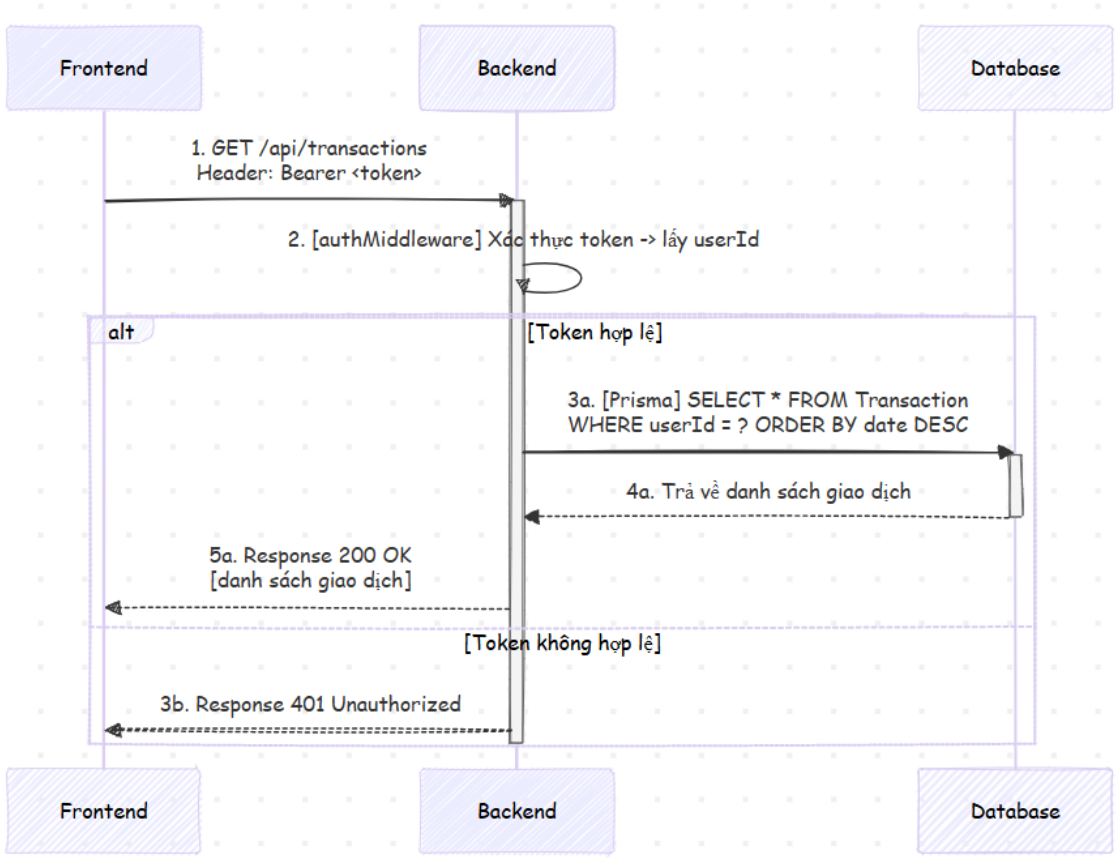
Đây là tài nguyên cốt lõi, ghi lại mọi dòng tiền ra vào của người dùng.

- **POST /api/transactions:** Tạo một giao dịch mới. Request body chứa các thông tin như `title`, `amount`, `type` (INCOME/EXPENSE), `category`, và `date`. Server sẽ tự động liên kết giao dịch này với `userId` của người dùng đang đăng nhập.



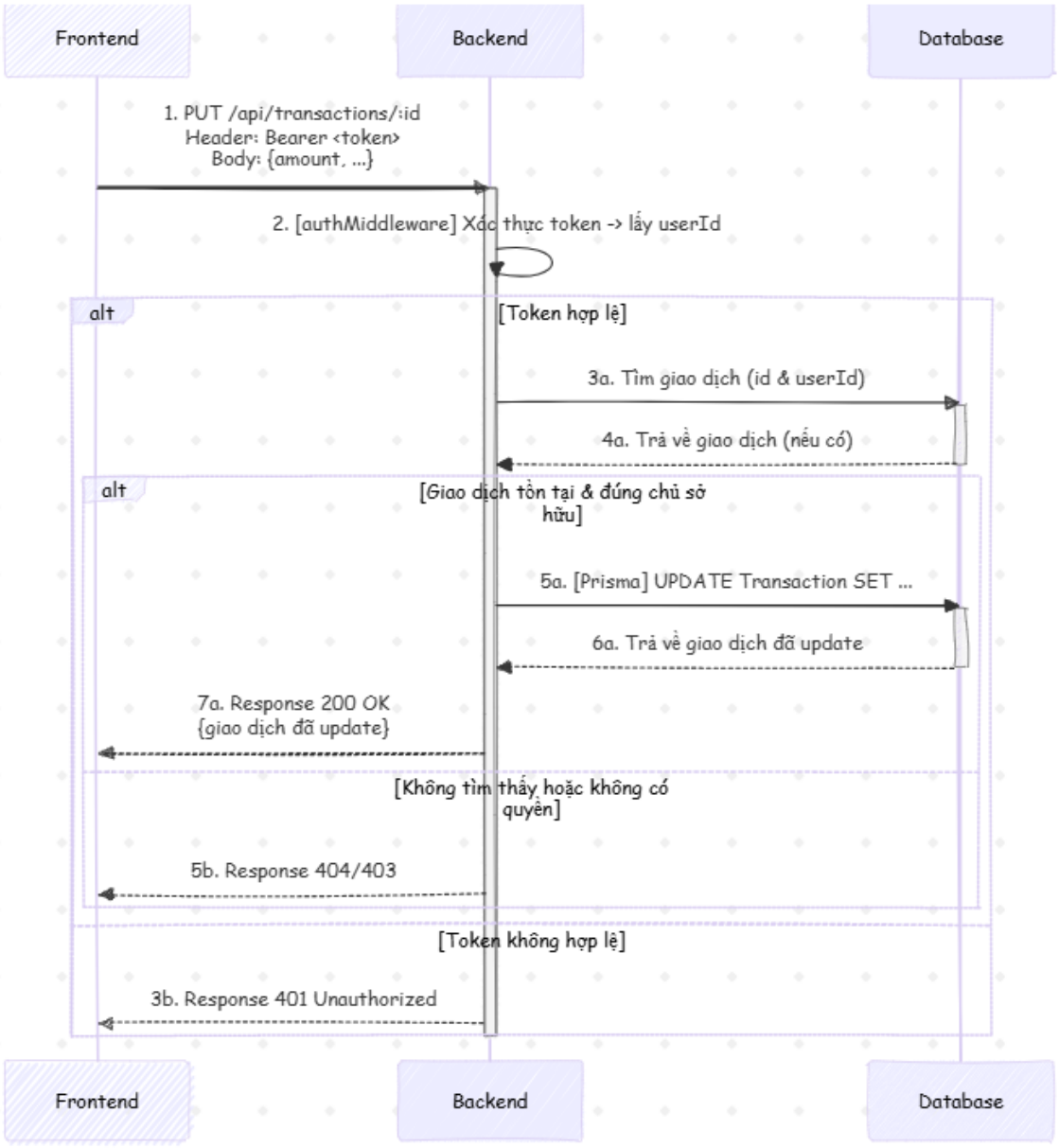
Hình 13: Sơ đồ Sequence biểu thị tạo giao dịch mới

- **GET /api/transactions:** Lấy danh sách toàn bộ giao dịch của người dùng, được sắp xếp theo ngày gần nhất. API này là nguồn dữ liệu chính cho trang Dashboard và trang lịch sử giao dịch.



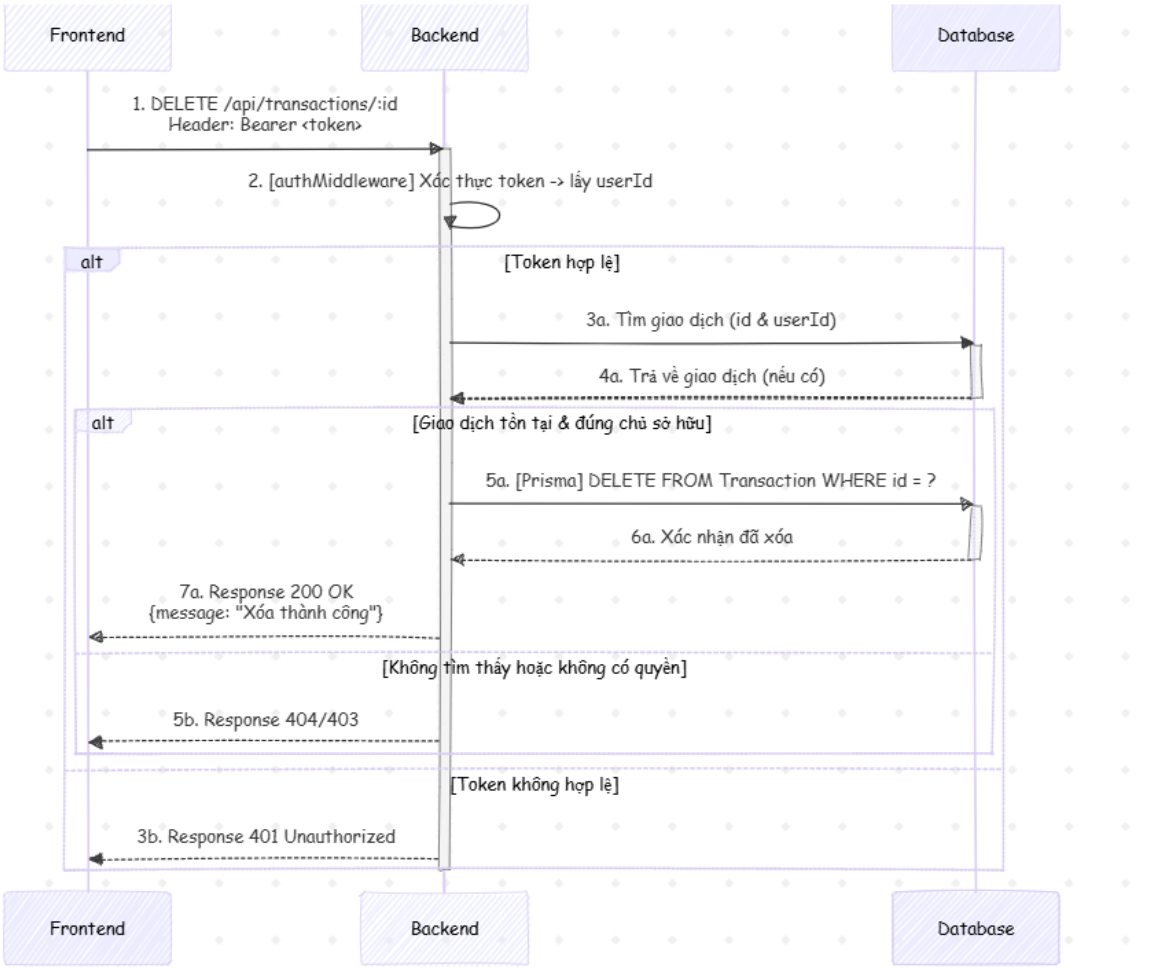
Hình 14: Sơ đồ Sequence biểu thị tạo giao dịch mới

- **PUT /api/transactions/:id:** Cập nhật thông tin cho một giao dịch cụ thể dựa trên ID của nó. Backend sẽ kiểm tra quyền sở hữu, đảm bảo người dùng không thể sửa giao dịch của người khác.



Hình 15: Sơ đồ Sequence biểu thị sửa giao dịch

- **DELETE /api/transactions/:id:** Xóa một giao dịch khỏi hệ thống. Tương tự, quyền sở hữu sẽ được kiểm tra trước khi thực hiện xóa.

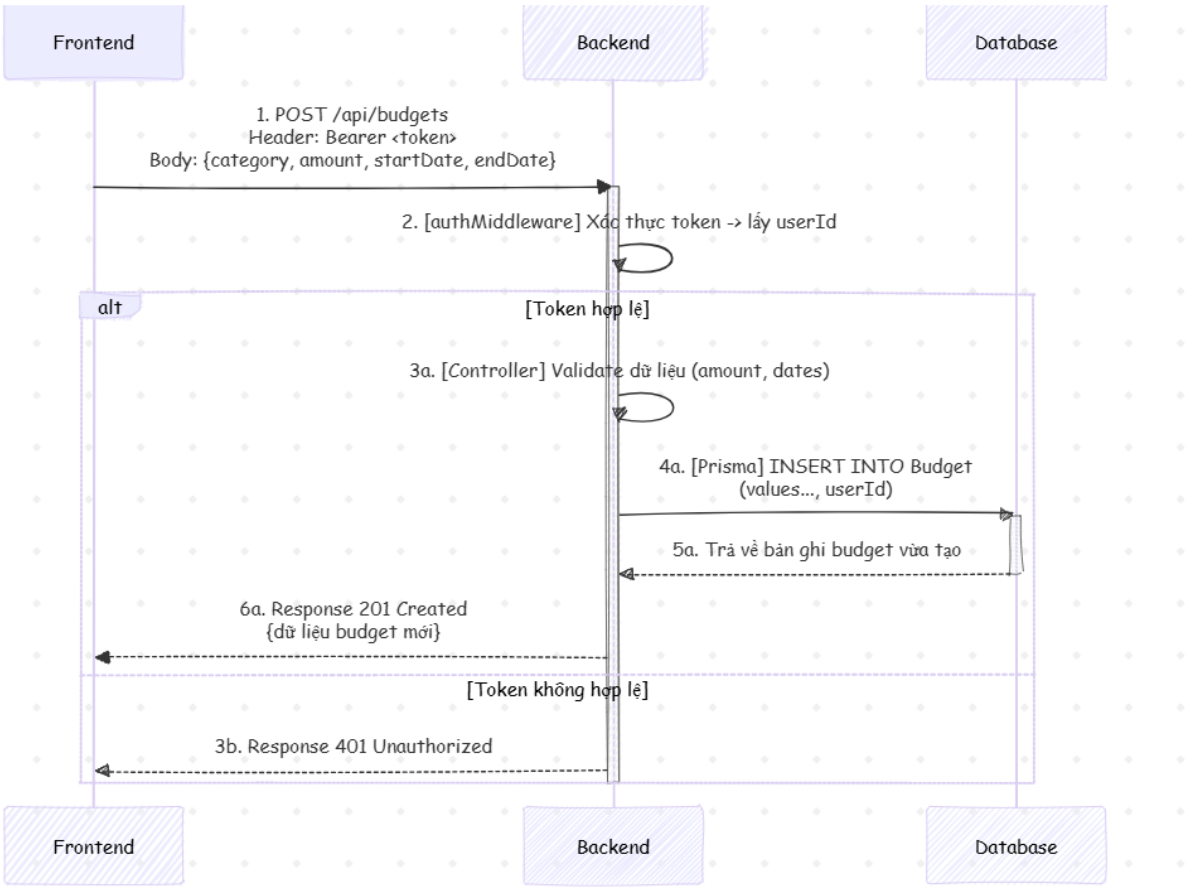


Hình 16: Sơ đồ Sequence biểu thị xóa giao dịch

6.2.2 Ngân sách (Budget) - Endpoint: `/api/budgets`

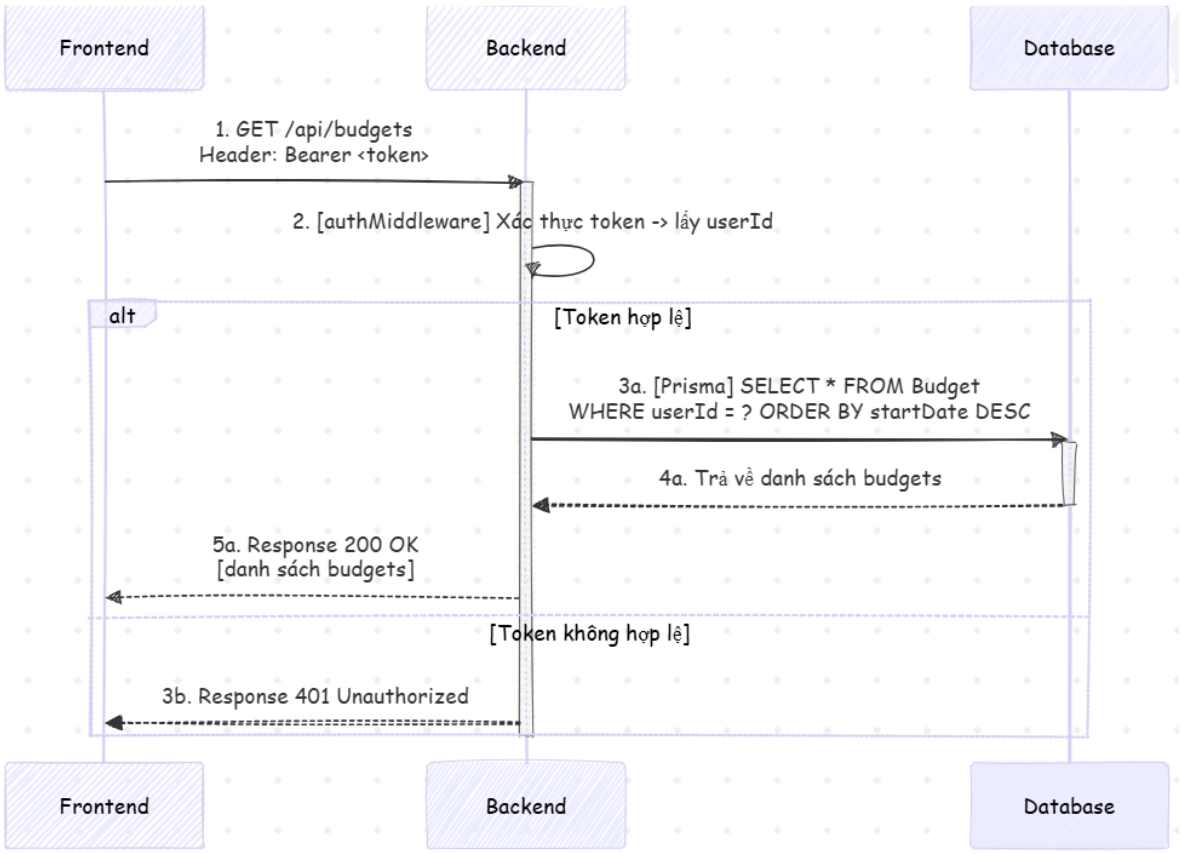
Tài nguyên này cho phép người dùng thiết lập giới hạn chi tiêu cho các danh mục.

- **POST /api/budgets:** Tạo một ngân sách mới cho một danh mục trong một khoảng thời gian xác định (`startDate`, `endDate`).



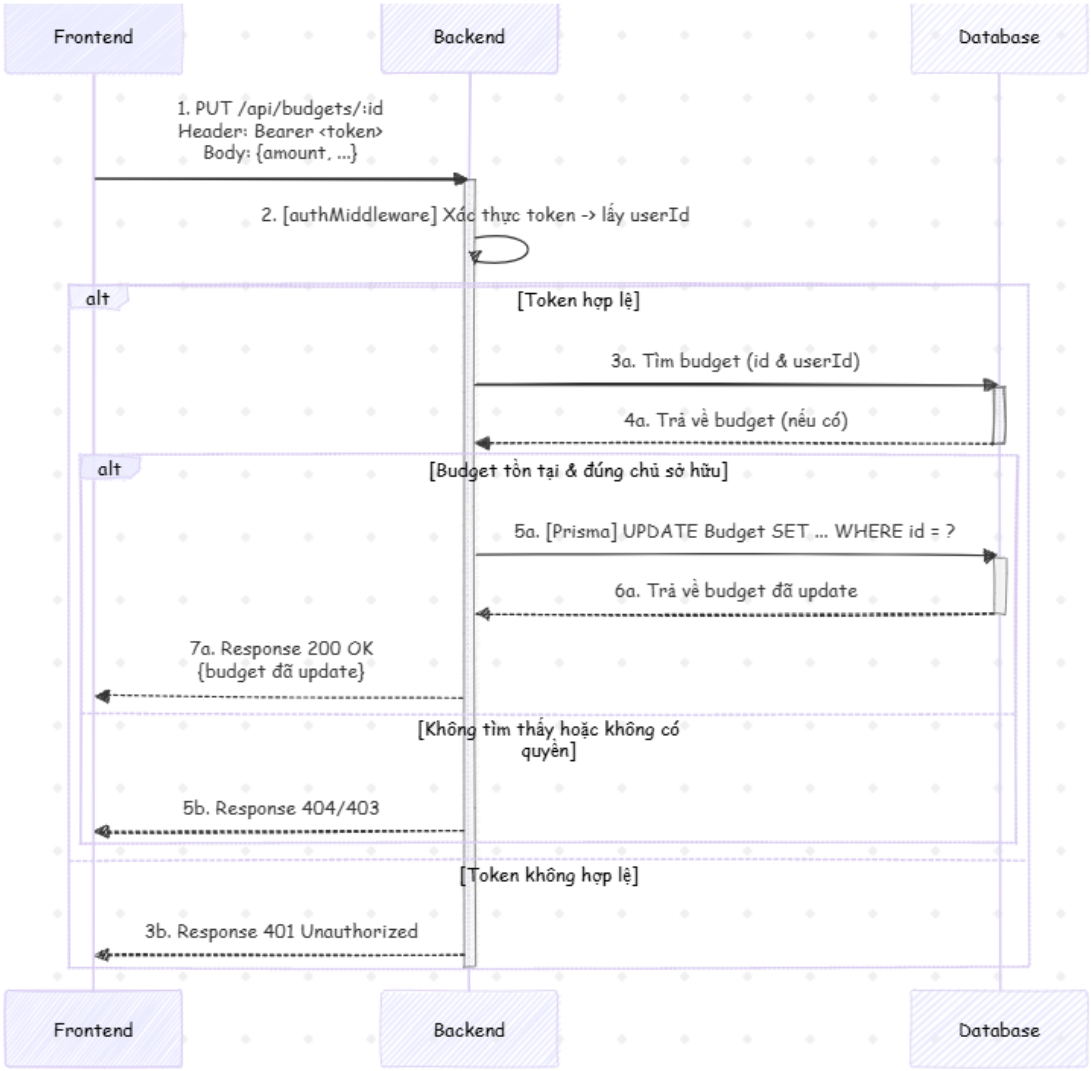
Hình 17: Sơ đồ Sequence biểu thị thêm ngân sách

- **GET /api/budgets:** Lấy danh sách tất cả các ngân sách mà người dùng đã thiết lập. Dữ liệu này được dùng để hiển thị trên trang Quản lý Ngân sách và có thể được dùng để so sánh với chi tiêu thực tế.



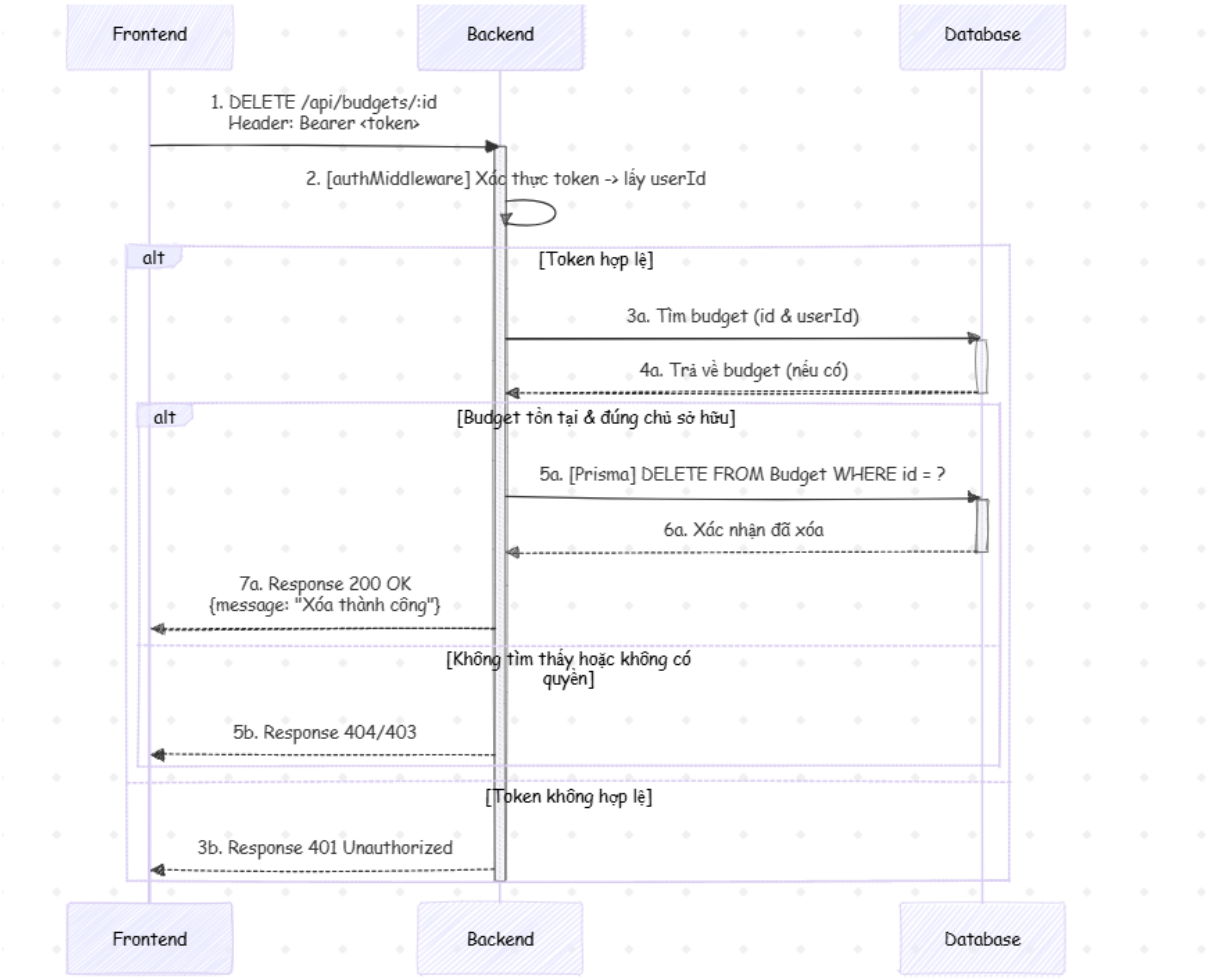
Hình 18: Sơ đồ Sequence biểu thị lấy về ngân sách

- **PUT /api/budgets/:id:** Cung cấp khả năng sửa đổi (ví dụ: thay đổi số tiền).



Hình 19: Sơ đồ Sequence biểu thị sửa ngân sách

- **DELETE /api/budgets/:id:** Xóa bỏ ngân sách đã được tạo.

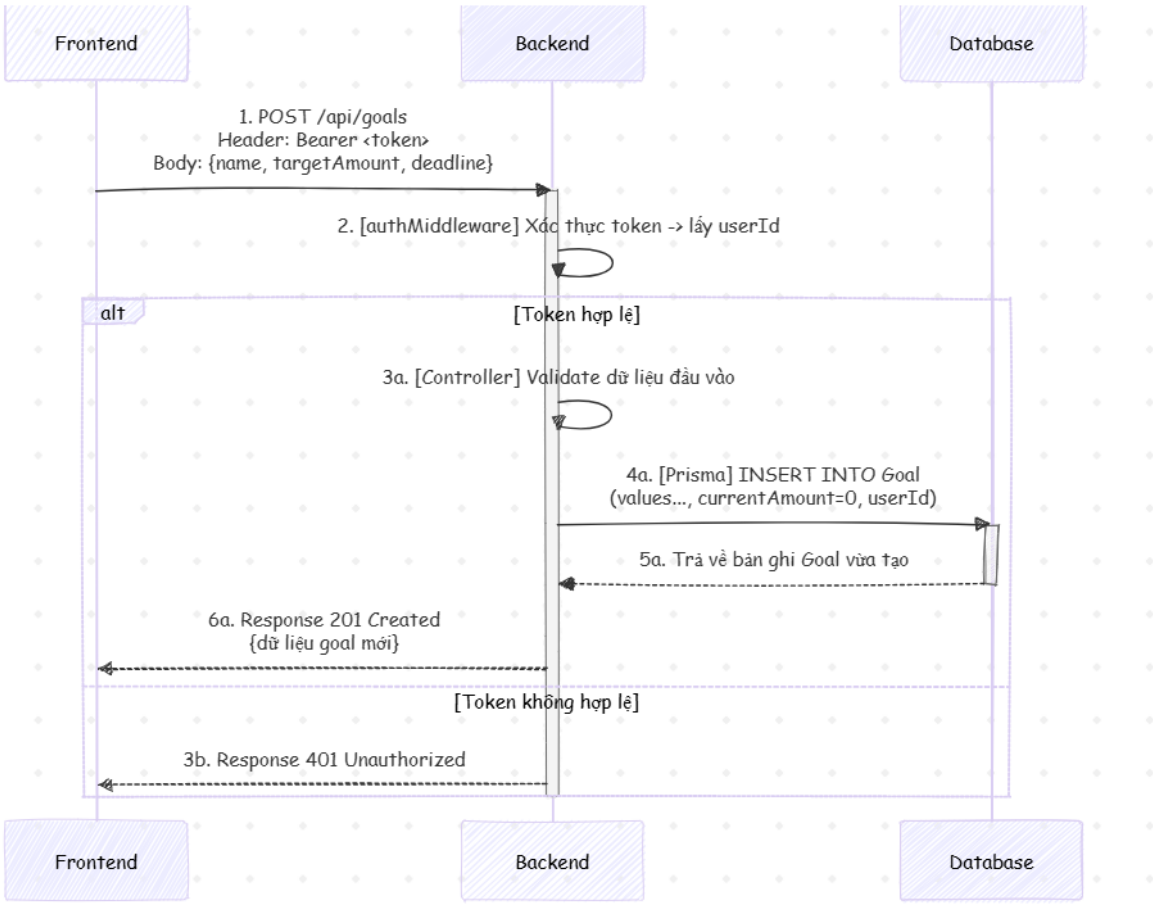


Hình 20: Sơ đồ Sequence biểu thị xóa ngân sách

6.2.3 Mục tiêu (Goal) - Endpoint: `/api/goals`

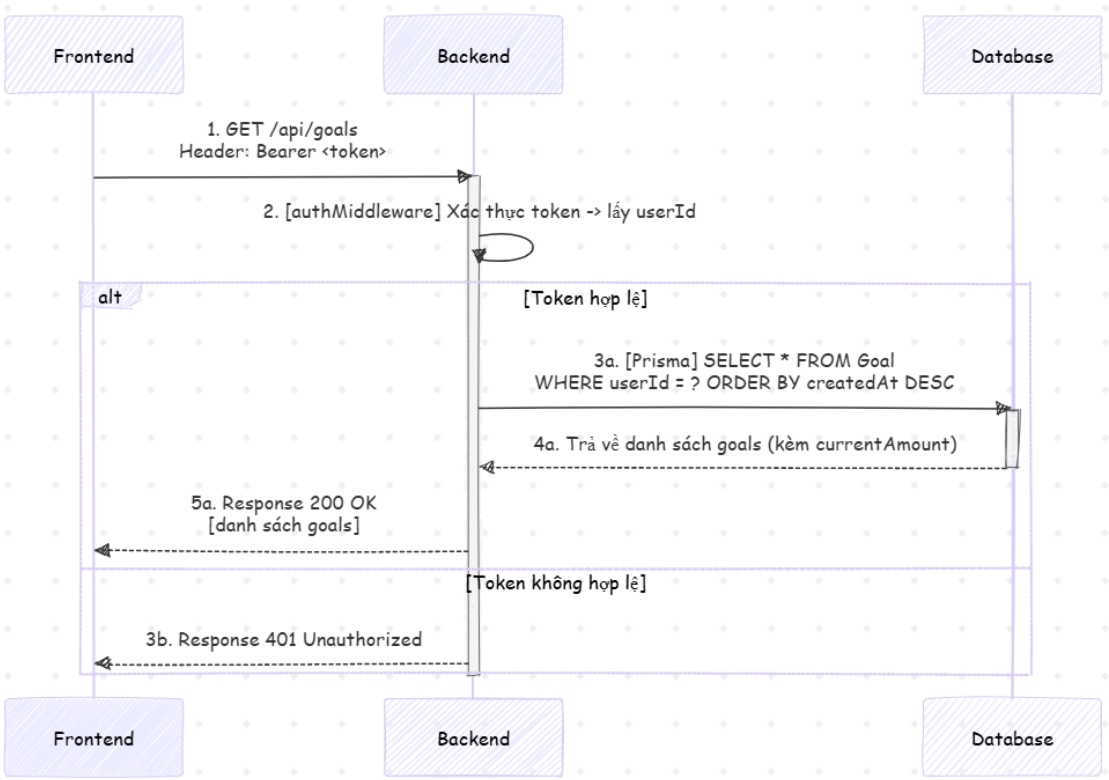
Tài nguyên này giúp người dùng theo dõi tiến độ hướng tới các mục tiêu tài chính dài hạn.

- **POST /api/goals:** Cho phép người dùng tạo một mục tiêu mới với các thông tin như tên mục tiêu (`name`), số tiền cần đạt (`targetAmount`), và hạn cuối (`deadline`). Trường `currentAmount` sẽ mặc định là 0.



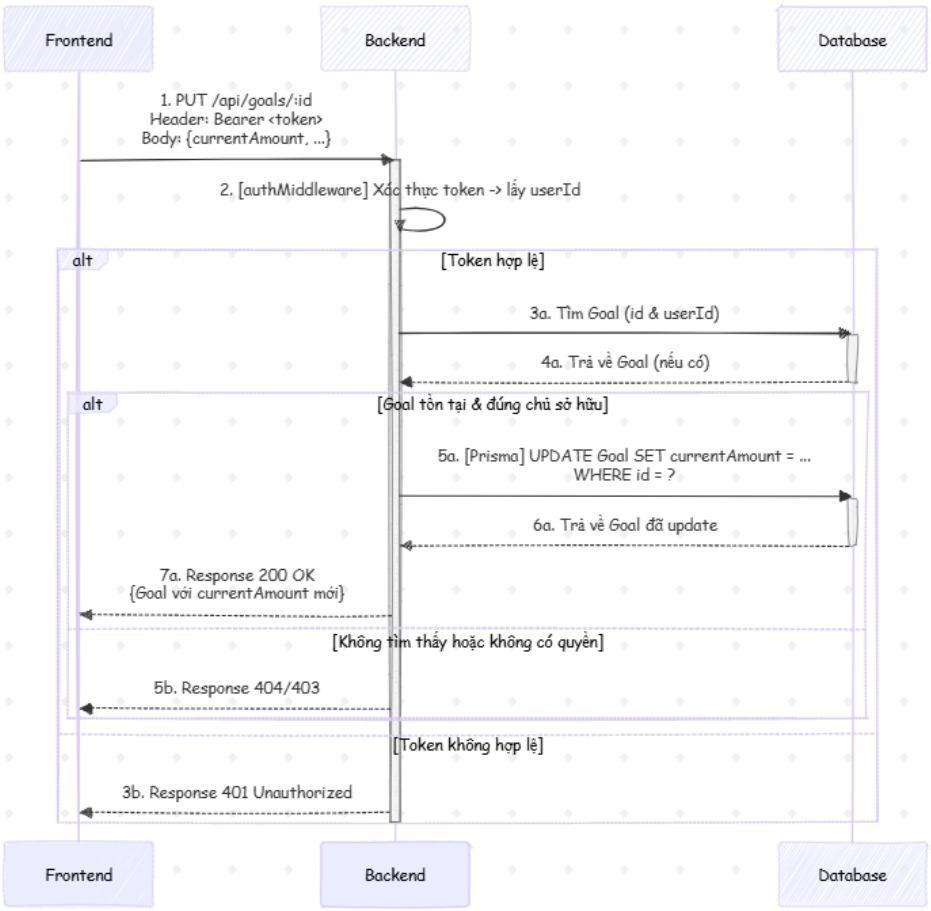
Hình 21: Sơ đồ Sequence biểu thị thêm mục tiêu mới

- **GET /api/goals:** Lấy danh sách tất cả các mục tiêu, bao gồm cả tiến độ hiện tại (`currentAmount`).



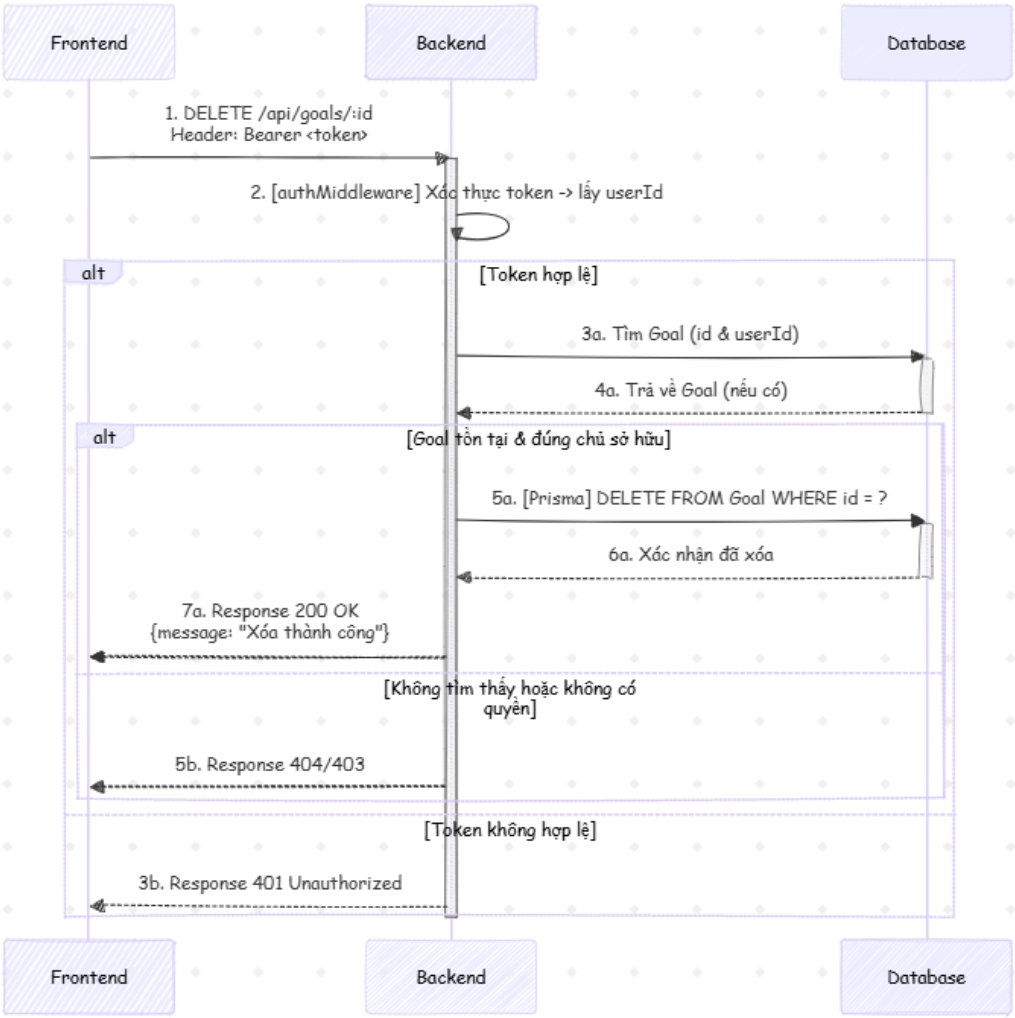
Hình 22: Sơ đồ Sequence biểu thị lấy mục tiêu

- **PUT /api/goals/:id:** Cập nhật thông tin của một mục tiêu. Chức năng này đặc biệt quan trọng, cho phép người dùng cập nhật `currentAmount` để ghi nhận số tiền họ đã tiết kiệm được cho mục tiêu đó.



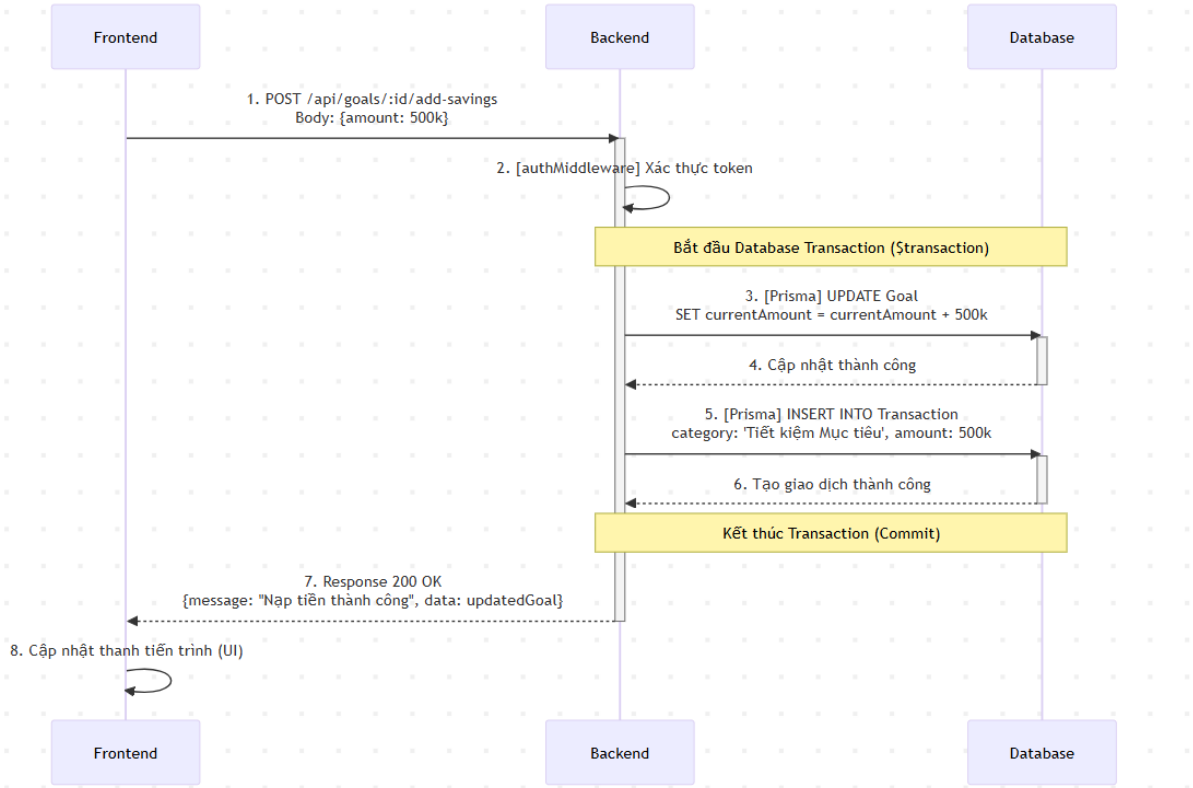
Hình 23: Sơ đồ Sequence biểu thị sửa mục tiêu

- **DELETE /api/goals/:id:** Xóa một mục tiêu khỏi danh sách.



Hình 24: Sơ đồ Sequence biểu thị xóa mục tiêu

- **POST /api/goals/:id/add-savings:** Chức năng nạp tiền vào mục tiêu tiết kiệm.
  - Để đảm bảo tính nhất quán, hệ thống thực hiện đồng thời hai hành động:
    1. Cập nhật bản ghi `Goal`: Sử dụng toán tử `increment` để cộng dồn số tiền nạp vào trường `currentAmount`.
    2. Tạo bản ghi `Transaction`: Tự động tạo một giao dịch chi tiêu (`EXPENSE`) với danh mục đặc biệt là "Tiết kiệm Mục tiêu" để trừ vào số dư tổng của người dùng.



Hình 25: Sơ đồ Sequence biểu thị việc nạp thêm tiền vào mục tiêu

6.3 Nhập liệu bằng hóa đơn tự động (OCR & AI Parsing)

Để giải quyết vấn đề nhập liệu thủ công tẻ nhạt và dễ sai sót, hệ thống tích hợp một luồng xử lý hóa đơn thông minh, tận dụng khả năng đa phương thức của các mô hình ngôn ngữ lớn hiện đại. Luồng xử lý này gộp hai giai đoạn OCR (Nhận dạng ký tự) và Parsing (Phân tích cú pháp) vào một bước duy nhất.

6.3.1 Luồng xử lý chi tiết

1. **Tải lên phía Client (Frontend):** Người dùng sử dụng giao diện web để chọn và tải lên một tệp hình ảnh chứa hóa đơn chi tiêu. Giao diện frontend sẽ hiển thị một bản xem trước của hình ảnh.

2. **Xử lý phía Server (Backend - Endpoint: `/api/ai/parse-receipt`):**

   - Server nhận file ảnh thông qua một request `POST` dạng `multipart/form-data`, được xử lý bởi middleware `multer`.
   - File ảnh được chuyển đổi thành chuỗi **base64** và chuẩn bị để đưa vào mô hình AI.
   - **Thiết kế Prompt:** Một prompt chi tiết được xây dựng, đóng vai trò như một bộ chỉ dẫn cho model AI. Prompt này bao gồm:
     - **Vai trò:** Chỉ định model đóng vai một chuyên gia phân tích hóa đơn.
     - **Nhiệm vụ:** Yêu cầu model xác định từng mục hàng riêng lẻ trên hóa đơn.

- **Trích xuất:** Yêu cầu trích xuất các trường thông tin cụ thể cho mỗi mục: `title`, `amount`, và `category`.
- **Ràng buộc:**
  * Yêu cầu model chỉ được phép gán nhãn cho các mục hàng bằng một trong các danh mục hợp lệ sau: `"Ăn uống"`, `"Thực phẩm"`, `"Di chuyển"`, `"Hóa đơn"`, `"Giải trí"`, `"Sức khỏe"`, `"Mua sắm"`, `"Khác"`. Việc này đảm bảo dữ liệu đầu ra từ AI tương thích với hệ thống phân loại của ứng dụng.
  * Chỉ thị rõ ràng cho model bỏ qua các dòng không phải là sản phẩm chi tiêu thực tế, ví dụ như các dòng tổng cộng ("Total", "Thành tiền"), thuế ("VAT", "Tax"), hoặc thông tin cửa hàng, giúp dữ liệu sạch và chính xác ngay từ đầu.
- **Định dạng đầu ra:** Yêu cầu model trả về kết quả dưới dạng một đối tượng JSON duy nhất với cấu trúc `{"transactions": [...] }`, trong đó giá trị là một mảng các đối tượng giao dịch.

- **Gọi mô hình:** Cả prompt văn bản và dữ liệu ảnh base64 được gửi đồng thời đến model **Gemini 2.5 Flash**.
- **Hậu xử lý:** Server nhận chuỗi JSON từ Gemini, thực hiện các bước làm sạch (trimming, removing markdown) và phân tích (parsing) để đảm bảo dữ liệu hợp lệ trước khi gửi về cho client.

3. **Hiển thị và xác nhận phía Client (Frontend):**

- Giao diện nhận về mảng các đối tượng giao dịch và hiển thị chúng dưới dạng một danh sách các form con có thể chỉnh sửa.
- Người dùng có toàn quyền kiểm tra, sửa đổi (ví dụ: sửa lại tên, số tiền, hoặc danh mục) hoặc xóa bỏ các giao dịch mà AI đã đề xuất.
- Sau khi xác nhận, một loạt các request `POST /api/transactions` được gửi đi song song (sử dụng `Promise.all`) để lưu các giao dịch đã được xác thực vào hệ thống.

Cách tiếp cận này không chỉ giúp tự động hóa việc nhập liệu từ hóa đơn mà còn thể hiện được khả năng ứng dụng kỹ thuật Prompt Engineering để điều khiển và ràng buộc đầu ra của một mô hình AI phức tạp, đảm bảo dữ liệu trả về có cấu trúc và nhất quán.

6.4 Lập kế hoạch tiết kiệm

Áp dụng thuật toán lập kế hoạch tiết kiệm được trình bày ở phần 7.

7 Thuật toán lập kế hoạch tiết kiệm cho tháng tới

7.1 Tổng quan phương pháp

Để giải quyết bài toán tư vấn tài chính cá nhân, tôi không sử dụng các mô hình học sâu dạng hộp đen do hạn chế về dữ liệu huấn luyện cá nhân hóa. Thay vào đó, hệ thống áp dụng phân tích thống kê để đưa ra tư vấn. Phương pháp này đảm bảo tính minh bạch, khả năng giải thích và hoạt động ổn định ngay cả với tập dữ liệu nhỏ.

7.2 Giai đoạn 1: Trích xuất đặc trưng

Đầu vào là tập hợp các giao dịch T thuộc các tháng cơ sở M_{base} do người dùng lựa chọn. Với mỗi danh mục chi tiêu c_i, hệ thống tính toán vector đặc trưng $V_i = [\mu_i, \sigma_i, R_i]$:

- **Chi tiêu trung bình (μ_i):**

$$\mu_i = \frac{\sum_{t \in T_{c_i}} amount(t)}{|M_{base}|} \tag{1}$$

  Việc sử dụng mẫu số là số tháng thực tế được chọn ($|M_{base}|$) giúp thuật toán thích nghi linh hoạt với dữ liệu đầu vào.

- **Độ biến động (σ_i):** Sử dụng độ lệch chuẩn (Standard Deviation) để đo lường mức độ ổn định của hành vi chi tiêu.

- **Trạng thái lặp lại (R_i):** Sử dụng thuật toán so khớp chuỗi thời gian để phát hiện các khoản chi cố định (như tiền thuê nhà). Một danh mục được gán nhãn $R_i = True$ nếu tồn tại các cặp giao dịch thỏa mãn:

$$\Delta_{days} \in [28, 32] \quad \text{và} \quad \frac{|\Delta_{amount}|}{amount} \leq 0.1 \tag{2}$$

7.3 Giai đoạn 2: Mô hình chấm điểm linh hoạt

Mỗi danh mục được gán một điểm số $S_i \in [0, 1]$ thể hiện khả năng cắt giảm chi tiêu. Điểm số cuối cùng S_i là sự kết hợp giữa bản chất danh mục (K_i) và hành vi thực tế của người dùng ($\bar{\sigma_i}$):

$$S_i = \begin{cases} 0 & \text{nếu } R_i = True \text{ (Khoản chi cố định)} \\ \alpha \cdot K_i + \beta \cdot \bar{\sigma_i} & \text{nếu } R_i = False \end{cases} \tag{3}$$

Trong đó:

- K_i: Điểm số từ Cơ sở tri thức (Knowledge Base Map). Tham số K_i đại diện cho bản chất của khoản chi tiêu. Hệ thống sử dụng một bảng tham chiếu cố định, được xây dựng dựa trên nguyên tắc phân loại *Nhu cầu (Needs) - Mong muốn (Wants)*:

Danh mục	**Điểm K_i**	**Giải thích**
Giải trí	0.9	Nhu cầu giải trí, dễ dàng cắt giảm nhất.
Mua sắm	0.8	Thường là chi tiêu cho sở thích, độ linh hoạt cao.
Quà tặng	0.7	Khoản chi không thường xuyên.
Ăn uống	0.6	Nhu cầu thiết yếu nhưng có thể điều chỉnh (tự nấu/ăn ngoài).
Di chuyển	0.4	Nhu cầu thiết yếu, khó điều chỉnh hơn.
Gia đình	0.3	Chi tiêu cho người thân, độ ưu tiên cao.
Sức khỏe	0.2	Rất quan trọng, hạn chế cắt giảm.
Giáo dục	0.1	Đầu tư dài hạn, không nên cắt giảm.
Hóa đơn	0.0	Các khoản cố định hàng tháng (điện, nước, internet).

Bảng 1: Bảng trọng số linh hoạt cho các danh mục chi tiêu (K_i)

- $\bar{\sigma_i}$: Độ biến động đã chuẩn hóa.

  Độ biến động gốc (σ_i) mang đơn vị tiền tệ, không thể kết hợp trực tiếp với các hệ số không đơn vị. Do đó, hệ thống thực hiện chuẩn hóa Min-Max để đưa σ_i về khoảng $[0, 1]$:

  $$\bar{\sigma_i} = \frac{\sigma_i}{\max_j(\sigma_j)} \tag{4}$$

  Trong đó $\max_j(\sigma_j)$ là độ lệch chuẩn lớn nhất tìm thấy trong tất cả các danh mục chi tiêu của người dùng. $\bar{\sigma_i}$ càng gần 1 nghĩa là danh mục đó có mức chi tiêu càng thất thường so với thói quen chung của người dùng.

- α, β: Trọng số điều chỉnh (Hệ thống sử dụng $\alpha = 0.6, \beta = 0.4$). Hệ thống sử dụng bộ trọng số $\alpha = 0.6$ và $\beta = 0.4$. Điều này phản ánh quan điểm: Bản chất của loại hình chi tiêu (Ăn uống hay Giải trí) đóng vai trò quyết định lớn hơn (60%), nhưng thói quen chi tiêu thất thường của người dùng cũng là một tín hiệu quan trọng (40%) để đề xuất cắt giảm.

7.4 Giai đoạn 3: Tối ưu hóa phân bổ thâm hụt

Hệ thống cần xác định chính xác số tiền người dùng cần cắt giảm thêm mỗi tháng. Quy trình tính toán gồm 3 bước:

1. Tính mục tiêu tiết kiệm tháng (S_{target}) Dựa trên tổng số tiền mục tiêu (G_{target}), số tiền hiện có ($G_{current}$) và thời gian còn lại tính bằng tháng (T_{remain}):

$$S_{target} = \frac{\max(0, G_{target} - G_{current})}{\max(1, T_{remain})} \tag{5}$$

Công thức sử dụng hàm max để đảm bảo mẫu số không bằng 0 và tử số không âm.

2. Tính khả năng tiết kiệm dự kiến ($S_{projected}$) Dựa trên thu nhập trung bình1 tháng (I_{avg}) và tổng chi tiêu trung bình của tất cả các danh mục ($\sum \mu_i$) đã tính ở Giai đoạn 1:

$$S_{projected} = I_{avg} - \sum_i \mu_i \tag{6}$$

3. Tính khoản thâm hụt ($D_{deficit}$) Thâm hụt là phần chênh lệch giữa mục tiêu cần đạt và khả năng hiện tại:

$$D_{deficit} = S_{target} - S_{projected} \tag{7}$$

Điều kiện dừng: Nếu $D_{deficit} \leq 0$, nghĩa là $S_{projected} \geq S_{target}$ (người dùng đang tiết kiệm đủ hoặc vượt mức). Thuật toán sẽ dừng lại và đưa ra lời khen ngợi thay vì gợi ý cắt giảm. Ngược lại, nếu $D_{deficit} > 0$, thuật toán chuyển sang bước phân bổ cắt giảm. Bước này như sau:
Hệ thống giải quyết bài toán phân bổ khoản thâm hụt tài chính ($D_{deficit}$) vào các danh mục dựa trên **Điểm đóng góp (C_i)**.

$$C_i = \mu_i \times S_i \tag{8}$$

Số tiền cần cắt giảm (Cut_i) cho danh mục i được tính theo tỷ trọng:

$$Cut_i = D_{deficit} \times \frac{C_i}{\sum_j C_j} \tag{9}$$

Mô hình này đảm bảo : Cắt giảm nhiều nhất ở những hạng mục vừa tiêu tốn nhiều tiền, vừa có độ linh hoạt cao, và bảo toàn các khoản chi thiết yếu.

4. Tính toán ngân sách đề xuất ($Budget_i$) Sau khi đã xác định được số tiền cần cắt giảm cho mỗi danh mục, hệ thống sẽ tính toán ngân sách đề xuất cho tháng tiếp theo. Ngân sách này được tính bằng cách lấy chi tiêu trung bình trước đó (μ_i) trừ đi phần cắt giảm đã được phân bổ (Cut_i).

$$Budget_i = \max(0, \mu_i - Cut_i) \tag{10}$$

Công thức sử dụng hàm $\max(0, ...)$ để đảm bảo rằng ngân sách đề xuất không bao giờ là một số âm, ngay cả trong trường hợp lý thuyết khi khoản cắt giảm đề xuất lớn hơn chi tiêu trung bình. Kết quả $Budget_i$ chính là giới hạn chi tiêu được khuyến nghị cho người dùng trong tháng tới để đạt được mục tiêu của họ.

7.5 Ví dụ thực nghiệm: Tính công bằng trong phân bổ

Xét một kịch bản: Người dùng có một khoản chi phí rất lớn dành cho Giáo dục (vượt trội so với các khoản khác). Theo logic thông thường, các hạng mục chi tiêu lớn thường là mục tiêu đầu tiên để cắt giảm. Tuy nhiên, Giáo dục là khoản đầu tư thiết yếu. Hãy xem thuật toán xử lý tình huống này như thế nào.

Giả định kịch bản:

- Người dùng cần cắt giảm thêm: $D_{deficit} = 1,000,000$ VNĐ.
- Danh mục 1: **Giáo dục** (Chi tiêu rất lớn: học phí, mua sách, khóa học bổ trợ).
- Danh mục 2: **Ăn uống** (Chi tiêu trung bình, linh hoạt vừa phải).
- Danh mục 3: **Mua sắm** (Chi tiêu thấp hơn, nhưng rất linh hoạt).

7.5.1 Bước 1: Dữ liệu đầu vào và trích xuất đặc trưng

Bảng dưới đây thể hiện các chỉ số μ_i (Chi tiêu trung bình) và độ biến động chuẩn hóa ($\bar{\sigma}_i$) sau giai đoạn 1:

Danh mục (i)	**Chi tiêu TB (μ_i)**	**Biến động ($\bar{\sigma}_i$)**
Giáo dục	**3,600,000**	0.1
Ăn uống	3,000,000	0.4
Mua sắm	1,500,000	0.8

Bảng 2: Dữ liệu đặc trưng của các danh mục

Nhận xét: Danh mục Giáo dục chiếm tỷ trọng chi tiêu lớn nhất (3.6 triệu VNĐ). Khoản này không cố định hoàn toàn ($R_i = False$) vì người dùng có thể mua thêm sách hoặc khóa học tùy tháng, dẫn đến độ biến động nhẹ ($\bar{\sigma} = 0.1$).

7.5.2 Bước 2: Tính điểm linh hoạt (S_i)

Áp dụng công thức tính điểm với trọng số $\alpha = 0.6$ (cho bản chất danh mục) và $\beta = 0.4$ (cho hành vi biến động):

- **Giáo dục:** Điểm tri thức $K_{giao\_duc} = 0.1$. $S_{giao\_duc} = 0.6(0.1) + 0.4(0.1) = \mathbf{0.10}$.
- **Ăn uống:** Điểm tri thức $K_{an} = 0.6$. $S_{an} = 0.6(0.6) + 0.4(0.4) = \mathbf{0.52}$.
- **Mua sắm:** Điểm tri thức $K_{mua} = 0.8$. $S_{mua} = 0.6(0.8) + 0.4(0.8) = \mathbf{0.80}$.

7.5.3 Bước 3: Tính điểm đóng góp (C_i) và Phân bổ cắt giảm (Cut_i)

Tính Điểm đóng góp ($C_i = \mu_i \times S_i$):

- $C_{giao\_duc} = 3,600,000 \times 0.10 = 360,000$.
- $C_{an} = 3,000,000 \times 0.52 = 1,560,000$.
- $C_{mua} = 1,500,000 \times 0.80 = 1,200,000$.

Tổng điểm đóng góp: $\sum C = 360,000 + 1,560,000 + 1,200,000 = 3,120,000$.

Áp dụng công thức phân bổ với $D_{deficit} = 1,000,000$:

Danh mục	**Chi tiêu (μ_i)**	**Tỷ trọng cắt giảm**	**Đề xuất cắt giảm (Cut_i)**
Giáo dục	**3,600,000**	**11.5%**	**≈ 115,000 VNĐ**
Ăn uống	3,000,000	50.0%	≈ 500,000 VNĐ
Mua sắm	1,500,000	38.5%	≈ 385,000 VNĐ

Bảng 3: Kết quả phân bổ cắt giảm cuối cùng

7.5.4 Bước 4: Tính toán ngân sách đề xuất cho tháng tới ($Budget_i$)

Dựa trên số tiền cắt giảm đã được phân bổ, hệ thống tính toán ngân sách giới hạn cho tháng tới theo công thức $Budget_i = \mu_i - Cut_i$.

Danh mục	**Chi tiêu TB (μ_i)**	**Cắt giảm (Cut_i)**	**Ngân sách đề xuất ($Budget_i$)**
Giáo dục	**3,600,000**	**115,000**	**3,485,000**
Ăn uống	3,000,000	500,000	2,500,000
Mua sắm	1,500,000	385,000	1,115,000
Tổng cộng	**8,100,000**	**1,000,000**	**7,100,000**

Bảng 4: Bảng tổng kết kế hoạch tiết kiệm đề xuất (đơn vị: VNĐ).

7.5.5 Kết luận về tính thông minh của thuật toán

1. **Bảo vệ khoản chi lớn nhưng thiết yếu:** Mặc dù Giáo dục chiếm nhiều tiền nhất (gần 45% tổng chi tiêu), nó chỉ phải gánh 11.5% trách nhiệm cắt giảm. Số tiền 115,000 VNĐ là rất nhỏ so với tổng ngân sách 3.6 triệu, đủ để người dùng tối ưu hóa bằng cách tìm nguồn mua sách rẻ hơn mà không ảnh hưởng đến việc học.

2. Gánh nặng cắt giảm được chuyển sang "Ăn uống" và "Mua sắm". Đặc biệt, "Mua sắm" tuy chi tiêu ít nhất (1.5 triệu) nhưng lại bị cắt giảm lượng tiền gần tương đương với "Ăn uống" (3 triệu), do tính chất quá linh hoạt ($S_i = 0.8$) của nó.

8 Kết luận và hướng phát triển

8.1 Kết quả đạt được

- Hoàn thành xây dựng hệ thống backend và frontend theo kế hoạch đề ra, đảm bảo các chức năng hoạt động ổn định.
- Triển khai các tính năng CRUD cho phép người dùng quản lý toàn diện các thực thể tài chính cá nhân.
- Tích hợp mô hình ngôn ngữ lớn Gemini để phân tích hóa đơn, giúp tự động hóa và nâng cao trải nghiệm nhập liệu.
- Xây dựng và triển khai một thuật toán lập kế hoạch có tính giải thích được cao, có khả năng tạo ra kế hoạch tiết kiệm cá nhân hóa, chi tiết dựa trên dữ liệu thực tế và lựa chọn của người dùng.

8.2 Hướng phát triển

- Hoàn thiện tính năng nhập liệu bằng giọng nói (Speech-to-Text).
- Cải tiến thuật toán lập kế hoạch bằng cách cho phép người dùng tùy chỉnh các trọng số hoặc loại trừ các gợi ý không phù hợp. Cải thiện thuật toán có thể dùng được trong trường hợp có nhiều mục tiêu cùng lúc.
- Xây dựng giao diện biểu đồ, thống kê trực quan hơn để người dùng theo dõi tiến độ theo thời gian.
- Triển khai hệ thống thông báo để nhắc nhở người dùng về ngân sách hoặc tiến độ mục tiêu.