# CUSTOM WEB APPLICATION

# VUE WALLET

COS30043 - Interface Design & Development

Lecturer: Nguyen Van Cong

Author: Luu Tuan Hoang

ID: 104180391

# Introduction

Vue Wallet is a customized fintech web application designed for the COS30043 - Interface Design & Development unit. This application demonstrates the application of key concepts from the course, showcasing the depth of understanding and the practical implementation of interface design and development principles. The project includes both frontend and backend components, each tailored to meet specific requirements and functionalities.

# Project Structure

The project is structured into two main directories: frontend and backend.

**Frontend**

The frontend of the application is built using Vue.js and utilizes the Composition API with Typescript for a more modern and efficient approach to state management using Pinia and component development. The frontend is responsible for the user interface, including the registration form, transaction charts using Apexchart, and overall user experience.

**Backend**

The backend of the application is built with Node.js and Express.js, providing a robust API for managing user data and transactions. It includes endpoints for user registration, login, fetching financial data, and more. The backend also handles PostgreSQL database migrations and other server-side operations using Prisma.

# Features of the Project

## Index

- **Show Ratings and Comments:** Display ratings and comments from other users to provide insights and feedback on transactions and categories.

## User Authentication

- **Login:** Implement a secure login system for users to access their accounts.
- **Registration:** Allow users to create new accounts with a straightforward registration process.

## Dashboard

- **Summary Display:** Present a comprehensive summary of the user's remaining balance, income, and expenses to provide an at-a-glance view of their financial status.

- **Pie Chart:** Utilize pie charts to visually represent the proportion of different financial categories.
- **Line Chart:** Incorporate line charts to show financial trends and changes over time, helping users track their income and expenses.

### Transactions

- **CRUD Operations:** Enable users to create, read, update, and delete transactions easily, ensuring they can manage their financial data effectively.
- **Filter:** Implement advanced filtering options to allow users to search and sort transactions based on various criteria, such as date, amount, and category.

### Category

- **CRD Operations:** Allow users to create, read, and delete categories to organize their transactions better.
- **Find Category:** Provide search functionality to help users quickly locate specific categories.

### Currency

- **Conversion Rate:** Display the most updated conversion rates for various currencies, allowing users to manage international transactions and financial planning.
- **Pagination and Search:** Implement client-side pagination and search capabilities to enhance the user experience by allowing easy navigation and search within currency data.

### Rating

- **Ratings:** Allow users to rate transactions and categories, providing feedback and contributing to the overall user experience.
- **Edit Ratings:** Enable users to edit their ratings, ensuring they can update their feedback as needed.

### Responsive Design

- **Tailwind CSS:** Utilize Tailwind CSS to ensure the application is fully responsive, providing an optimal user experience across different devices, including desktops, tablets, and smartphones.

## Functional Requirements

### User Authentication

**Login:**

- Users must be able to log in with their username and password.
- Passwords must be hashed and stored securely.

**Registration:**

- Users must be able to register by providing a unique username and a password.
- Confirmation of the password during registration.

### Dashboard

**Summary Display:**

- The dashboard must display a summary of the remaining balance, total income, and total expenses.

**Pie Chart:**

- A pie chart must show the distribution of expenses across different categories.

**Line Chart:**

- A line chart must display income and expenses trends over time (last 7 days).

### Transactions

**Create, Read, Update, Delete (CRUD):**

- Users must be able to add new transactions, view transaction details, update existing transactions, and delete transactions.

**Filter:**

- Users must be able to filter transactions by date, category, and amount.

### Category

**CRUD Operations:**

- Users must be able to create, read, update, and delete categories.

**Find Category:**

- Users must be able to search for categories by name.

### Currency

**Conversion Rate:**

- The application must display the most updated conversion rates for multiple currencies.

**Pagination and Search:**

- The currency conversion rate list must support client-side pagination and search functionality.

### Rating

**Ratings:**

- Users must be able to rate transactions and categories.

**Edit Ratings:**

- Users must be able to edit their ratings for transactions and categories.

### Index

**Show Ratings and Comments:**

- The index page must show ratings and comments from other users.

### Responsive Design

**Tailwind CSS:**

- The application must be responsive, providing an optimal experience on desktops, tablets, and smartphones.

## Nonfunctional Requirements

### Usability

**User-Friendly Interface:**

- The interface must be intuitive and easy to navigate.

**Accessibility:**

- The application must comply with accessibility standards (e.g., WCAG 2.1).

### Security

**Data Protection:**

- User data must be encrypted in transit and at rest.

**Authentication and Authorization:**

- Only authenticated users must access their data.

## Technical Aspects

### Frontend

#### Frameworks and Libraries

**Vue.js:**
- Utilized for building the user interface using both Composition API and Options API.

**Tailwind CSS:**
- Employed for styling and ensuring responsive design across different devices.

**Axios:**
- Used for making HTTP requests to the backend.

#### State Management

**Pinia:**
- Used for managing global state, including user authentication and other shared data.

#### API Integration

**RESTful API:**
- Communication with the backend to fetch and submit data for various features like transactions, categories, currency rates, and user ratings.

### Backend

#### Frameworks and Libraries

**Express.js:**
- Used for building the RESTful API.

**Prisma:**
- ORM (Object-Relational Mapping) for database operations and migrations.

**Node.js:**
- Runtime environment for executing server-side code.

#### Endpoints

**User Authentication:**
- Endpoints for login and registration with password hashing and validation.

**Transactions:**
- CRUD operations and filtering functionality.

**Categories:**
- Endpoints for managing categories.

**Currency:**
- Endpoints for fetching the most updated currency conversion rates.

**Ratings:**
- Endpoints for adding and editing user ratings.

#### Database

1. **PostgreSQL:**
   - Relational database management system used for storing user data, transactions, categories, currency rates, and ratings.

## Innovation Features

**State Management using Pinia**

- **Efficient State Handling:** Leveraging Pinia for state management to ensure scalable and maintainable code. This approach simplifies state handling across components and improves overall application performance.

**TypeScript + Composition API in Vue.js**

- **Type-Safe Development:** Utilizing TypeScript to provide static typing, enhancing code quality and reducing runtime errors.
- **Modern Vue.js Practices:** Employing Vue.js Composition API for a more organized and modular code structure, making it easier to manage complex application logic.

**Financial-Related Algorithm for Dashboard**

- **Advanced Financial Calculations:** Integrating algorithms to compute key financial metrics, trends, and insights. This feature provides users with comprehensive data analysis and actionable financial insights directly on their dashboard.

**Search and Filtering**

- **Enhanced Data Navigation:** Implementing robust search and filtering capabilities to allow users to efficiently locate and manage financial data. This includes dynamic search functionality and customizable filters for transactions, categories, and other financial records.

**Responsiveness**

- **Adaptive Design:** Ensuring that the application is fully responsive using Tailwind CSS, providing an optimal user experience across a variety of devices, including desktops, tablets, and smartphones. This feature guarantees that the application maintains usability and aesthetics regardless of screen size or orientation.