House Wallet App Requirements Document

Table of Contents

1. Introduction

The purpose of this document is to outline the requirements for the development of the House Wallet web application. This document provides an overview of the system, its functions, and its requirements.

2. Glossary

- - Authentication: the process of verifying the identity of a user, typically through a username and password.
- - Budget: a plan for managing income and expenses over a specified period of time.
- - Business logic: the rules and processes that define how an application should function based on the business requirements.
- - Controller: a component of the Model-View-Controller architecture that manages user requests, retrieves data from the model, and returns responses to the view.
- - CSS (Cascading Style Sheets): a language used to define the presentation and layout of HTML documents.
- - Database: a structured collection of data that is stored and organized for easy retrieval.
- - HTML (Hypertext Markup Language): a markup language used to create web pages.
- - JavaScript: a scripting language used to add interactivity and dynamic behavior to web pages.
- - Model: a component of the Model-View-Controller architecture that represents the data and business logic of an application.
- - MVC (Model-View-Controller): an architectural pattern used to separate an application into three interconnected components: model, view, and controller.
- - Non-functional requirements: requirements that specify how well the system should perform its functions, rather than what functions the system should perform.
- - PHP: a server-side scripting language used to create dynamic web pages.

- - Performance: how well an application meets its response time, throughput, and resource usage requirements.
- - Scalability: the ability of an application to handle increased workload and data volume.
- - Security: the measures taken to protect an application and its data from unauthorized access and attacks.
- - Usability: how easy an application is to use for its intended audience.
- - View: a component of the Model-View-Controller architecture that presents data to the user and accepts user input.

3. User Requirements Definition

This section outlines the requirements from the user's perspective. It includes a list of functional and non-functional requirements.

3.1 Functional Requirements

- Authentication for different Users Registration.
- Stay logged in for a year.
- Editing profile and budgeting settings. Budget Tracking & Analysis Expense Tracking.
- Filtering purchasing.
- Financial Limit & Goal Setting.
- Financial Reporting.
- Storing old budgeting details.

3.2 Non-functional Requirements

- **Security**: protect user data.
- **Performance**: fast and responsive.
- **Usability**: easy to use and user friendly.
- **Scalability**: able to handle large amounts of data and users as the app grows.

- **Compatibility**: compatible with different devices and browsers.

4. Application Architecture

The House Wallet web application will follow the Model-View-Controller (MVC) architecture, consisting of three components:

4.1 Model

The model will represent the data and business logic of the application. It will consist of a database to store the user's financial data and functions to handle the data logic.The model will be implemented using MYSQL as the database .

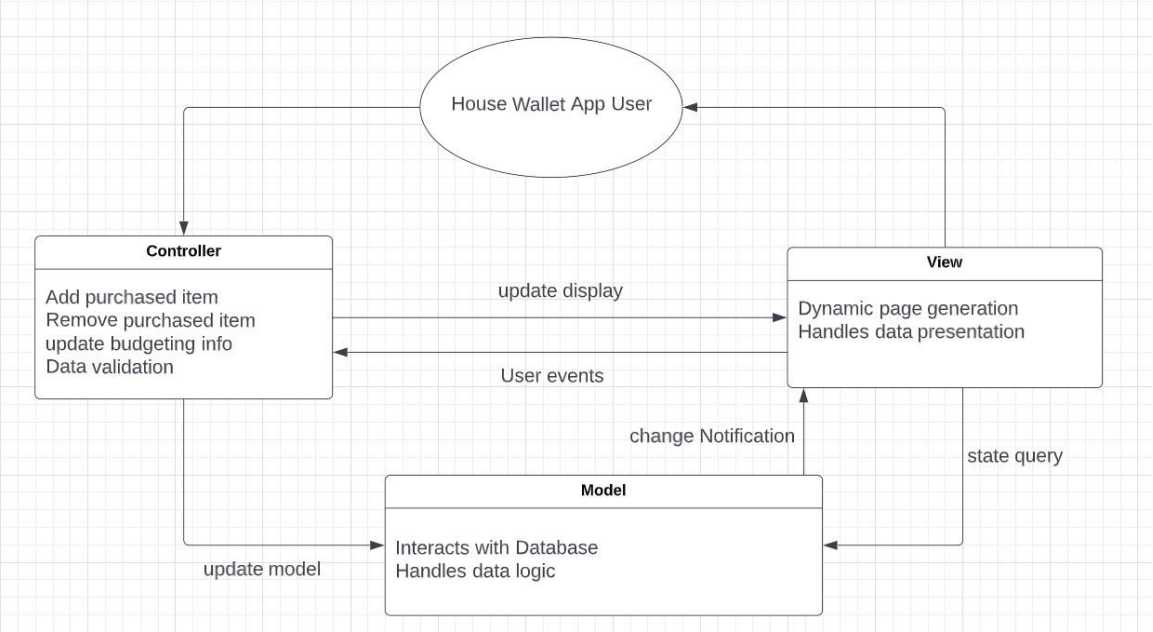
4.2 View

The view will be responsible for presenting data to the user and accepting user input. It will be built using HTML, CSS, and JavaScript on the front-end, and PHP on the back-end. The view will communicate with the controller to retrieve and update data.

4.3 Controller

The controller will handle user requests, retrieve and manipulate data from the model, and return responses to the view. It will be implemented using PHP. The controller will contain the business logic of the application and will communicate with the model to retrieve and manipulate data. The controller will also communicate with the view to present data to the user and receive user input.

The use of MVC architecture provides a clear separation of concerns between the model, view, and controller, allowing for easier maintenance and modification of the application.



5. System Requirements Specification

This section provides a detailed list of the system requirements, including functional and non-functional requirements.

5.1 Functional Requirements

- User authentication system and interfaces to allow users to create their accounts and log in securely.
- User profile management system to allow users to manage their account information and preferences.
- Session management system to keep the user logged in for a specified period.
- User profile management system to allow users to update their personal information and budgeting settings.

- Budget tracking system to allow users to create and manage their budgets.
- Budget analysis system to provide users with detailed reports and visualizations of their budgets.
- Expense tracking system to allow users to track their expenses and categorize them for better analysis.
- Purchase filtering system to allow users to filter their purchases by date, category, or other criteria.
- Goal & Limit setting system to allow users to set financial goals and track their progress towards them.
- Reporting system to generate reports and visualizations to help users understand their financial situation and progress towards their goals.
- Data storage system to store old budgeting details for future reference.

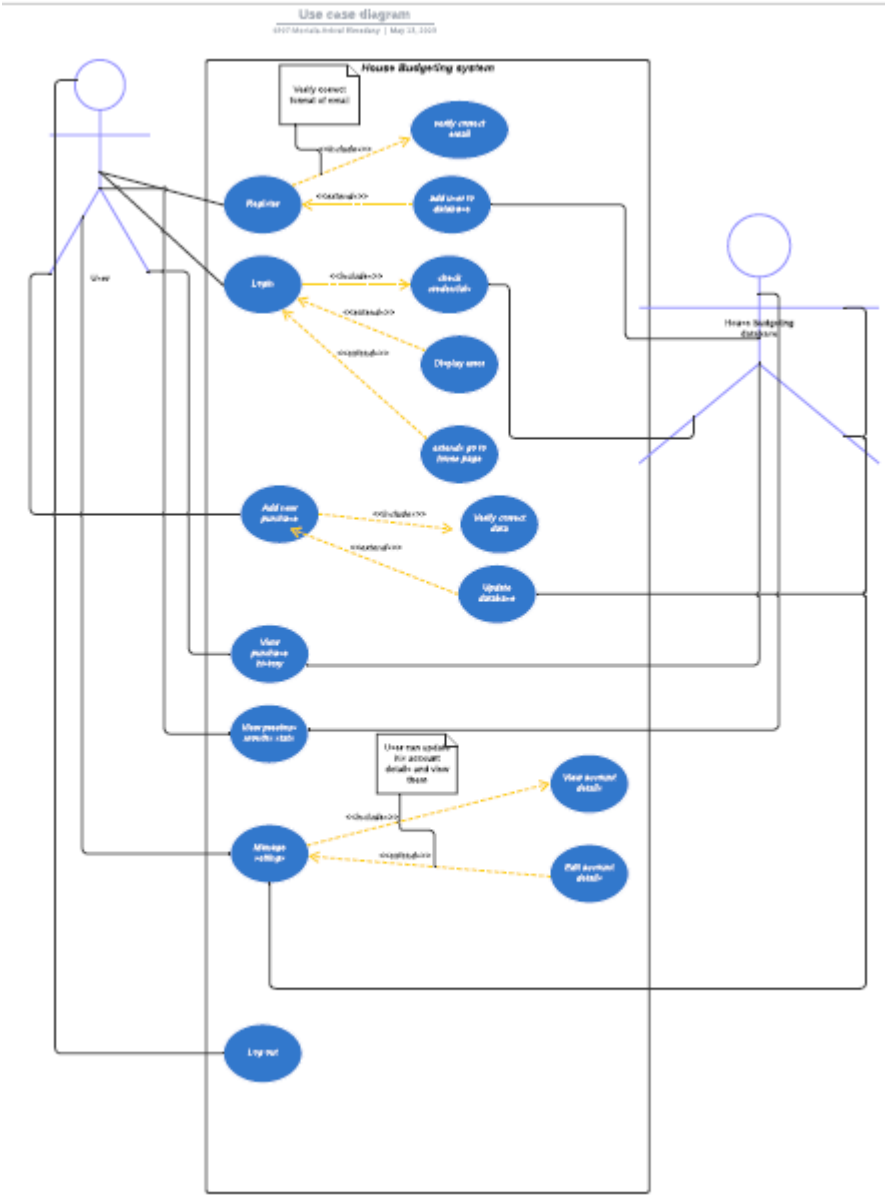
5.2 Non-functional Requirements

- Secure authentication system to protect user data, including login credentials and financial information.
- Secure data storage system to protect user data from unauthorized access.
- Fast and responsive application design to minimize lag and downtime.
- Efficient database design to handle large amounts of data.
- Scalable database design to handle an increasing number of users and data.
- User-friendly interface design to improve the user experience and ease of use.
- Responsive design to support different devices and screen sizes.

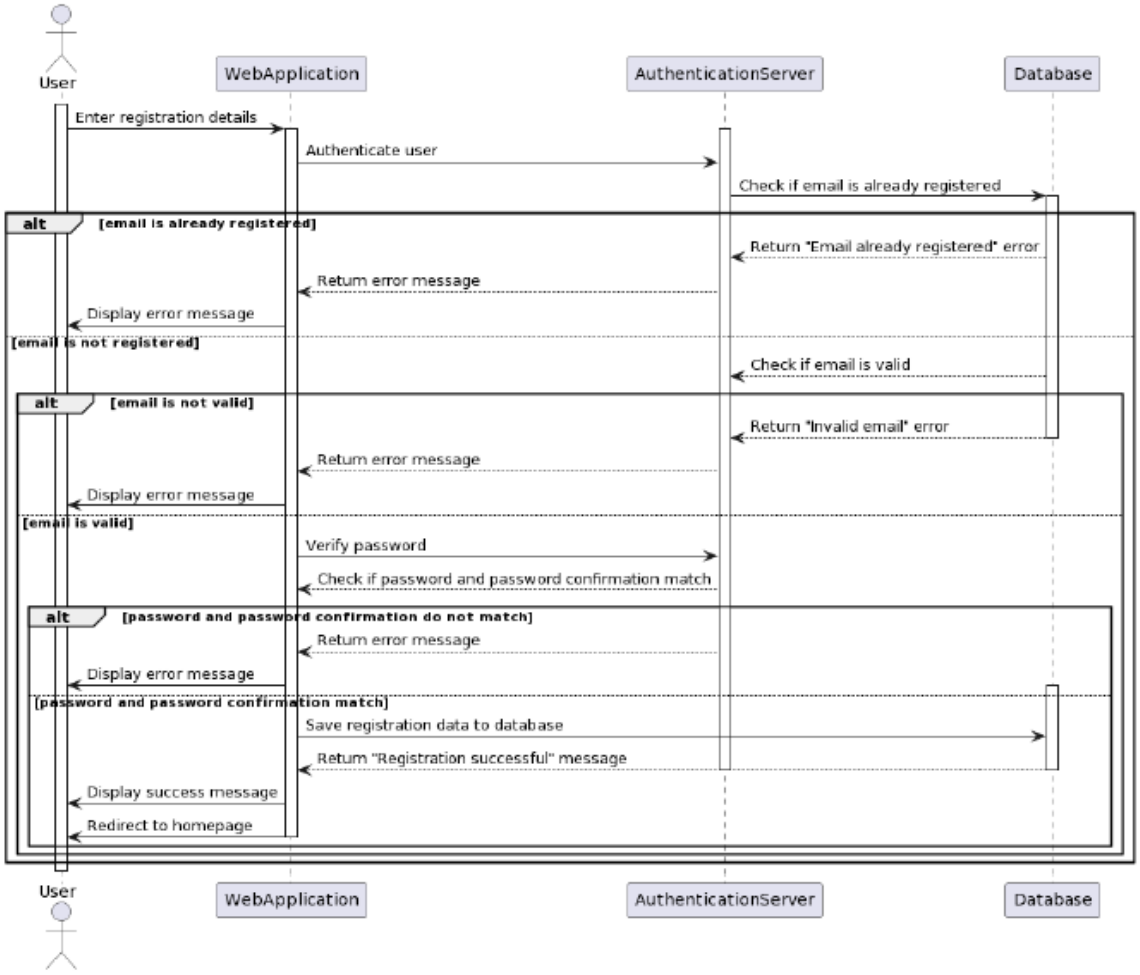
- Cross-browser compatibility to ensure the app works well on different browsers.
- Cross-device compatibility to ensure the app works well on different devices.

6. System Models

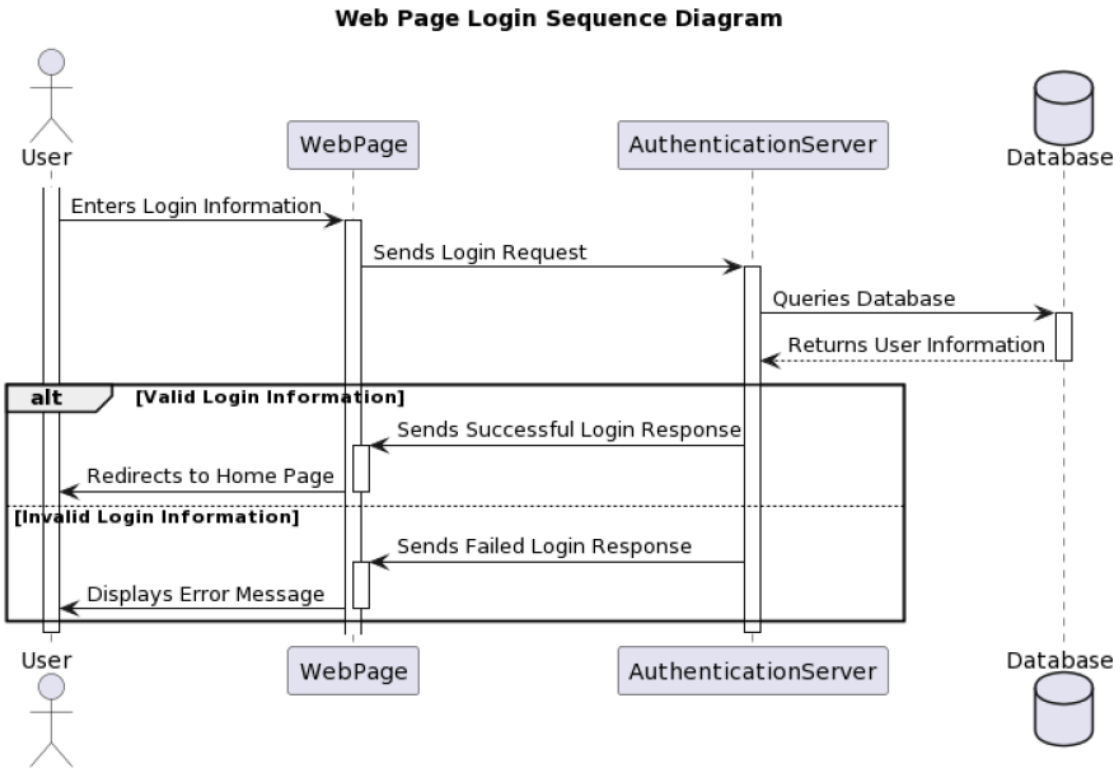
6.1 Use Case Diagram:



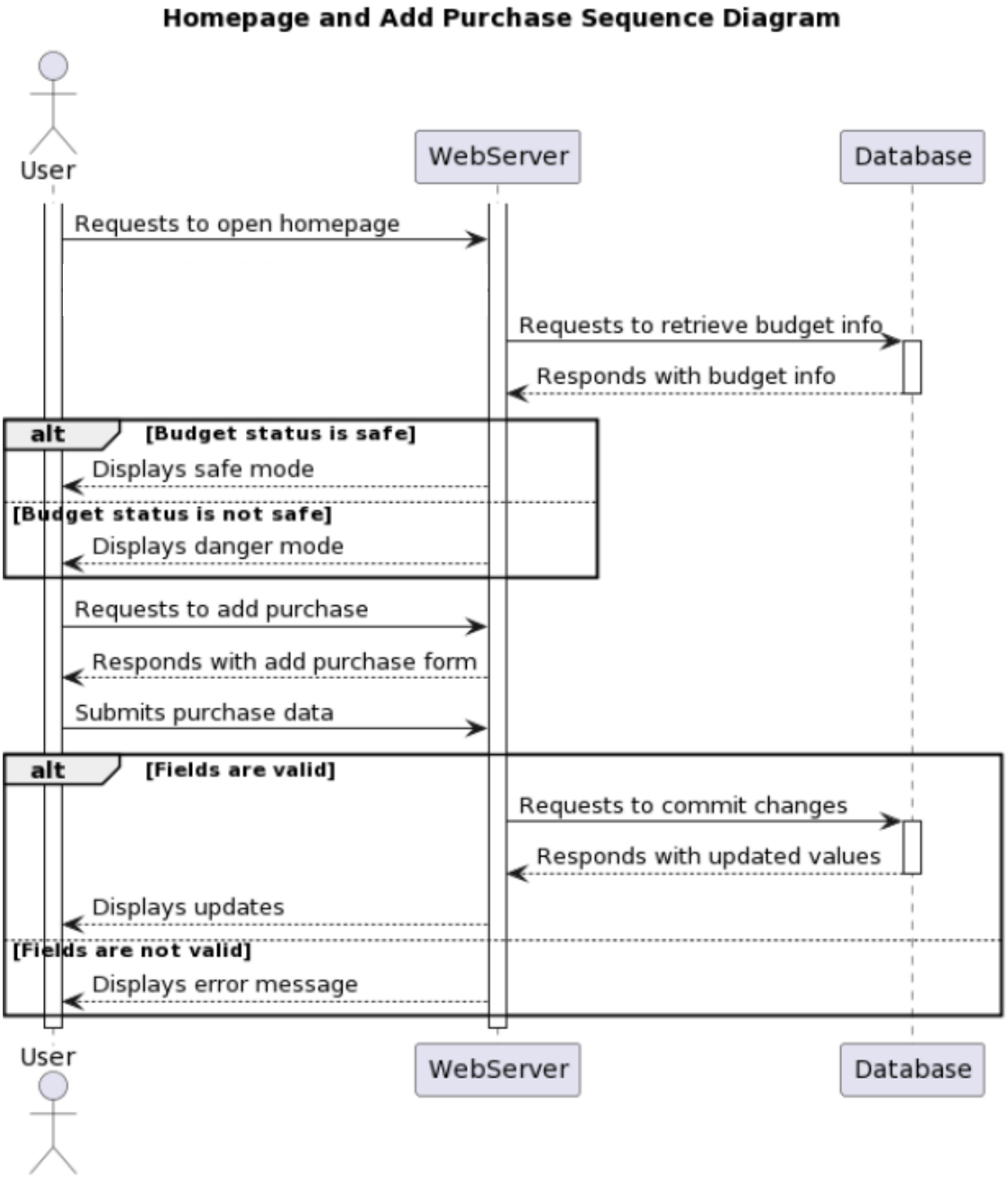
6.2 Registration Sequence Diagram:



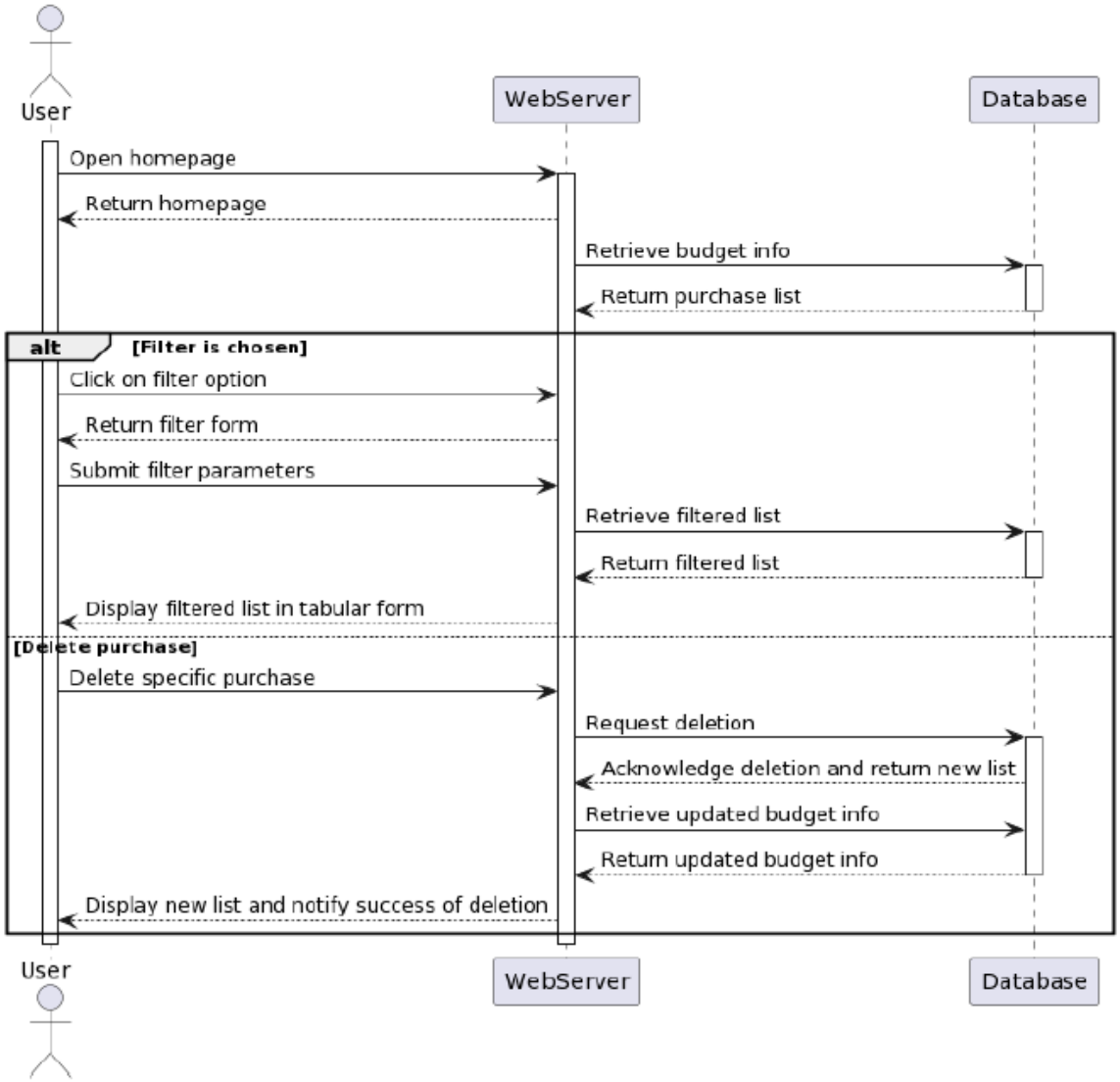
6.3 Login Sequence Diagram:



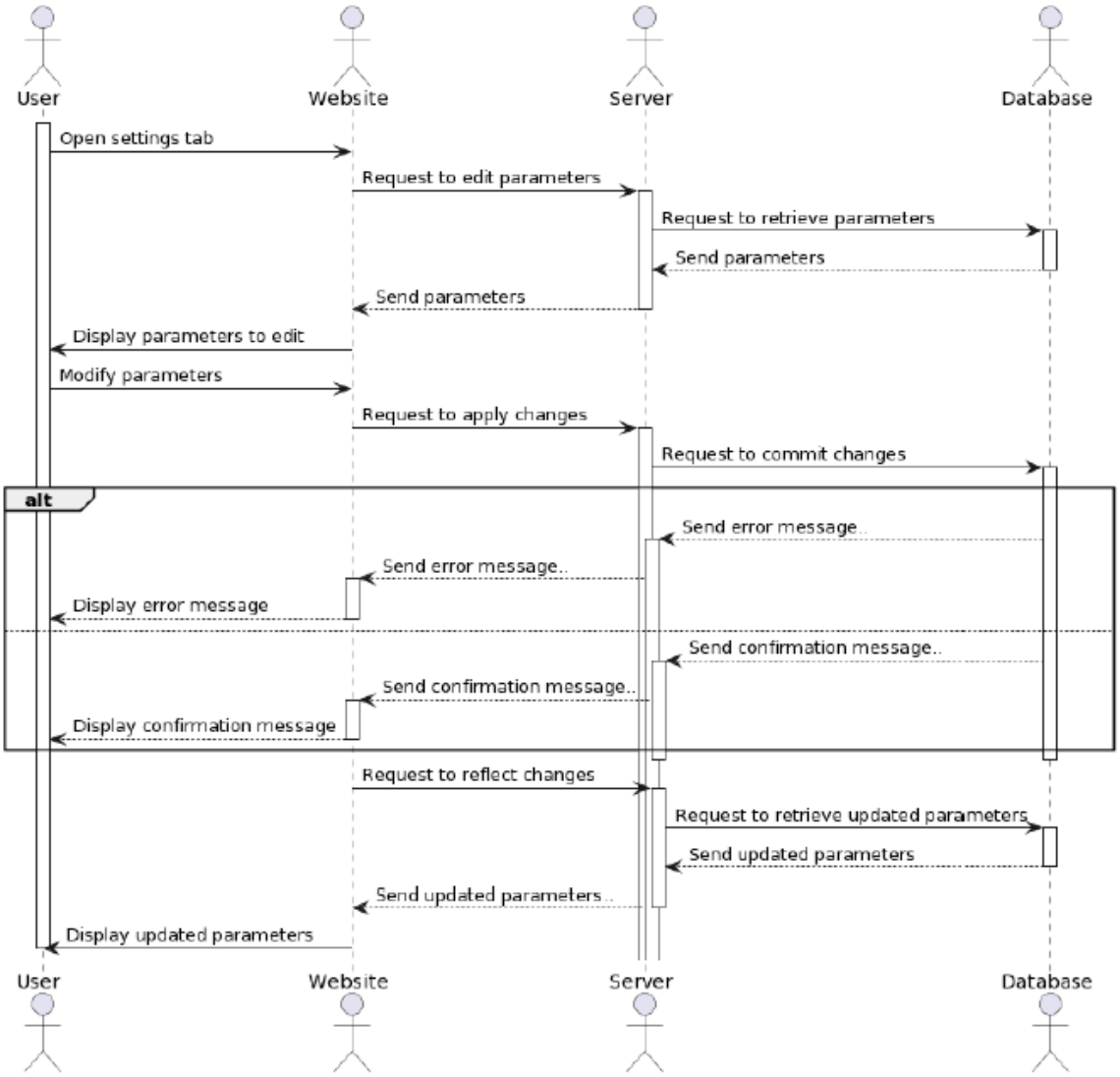
6.4 home and add purchase Sequence Diagram:



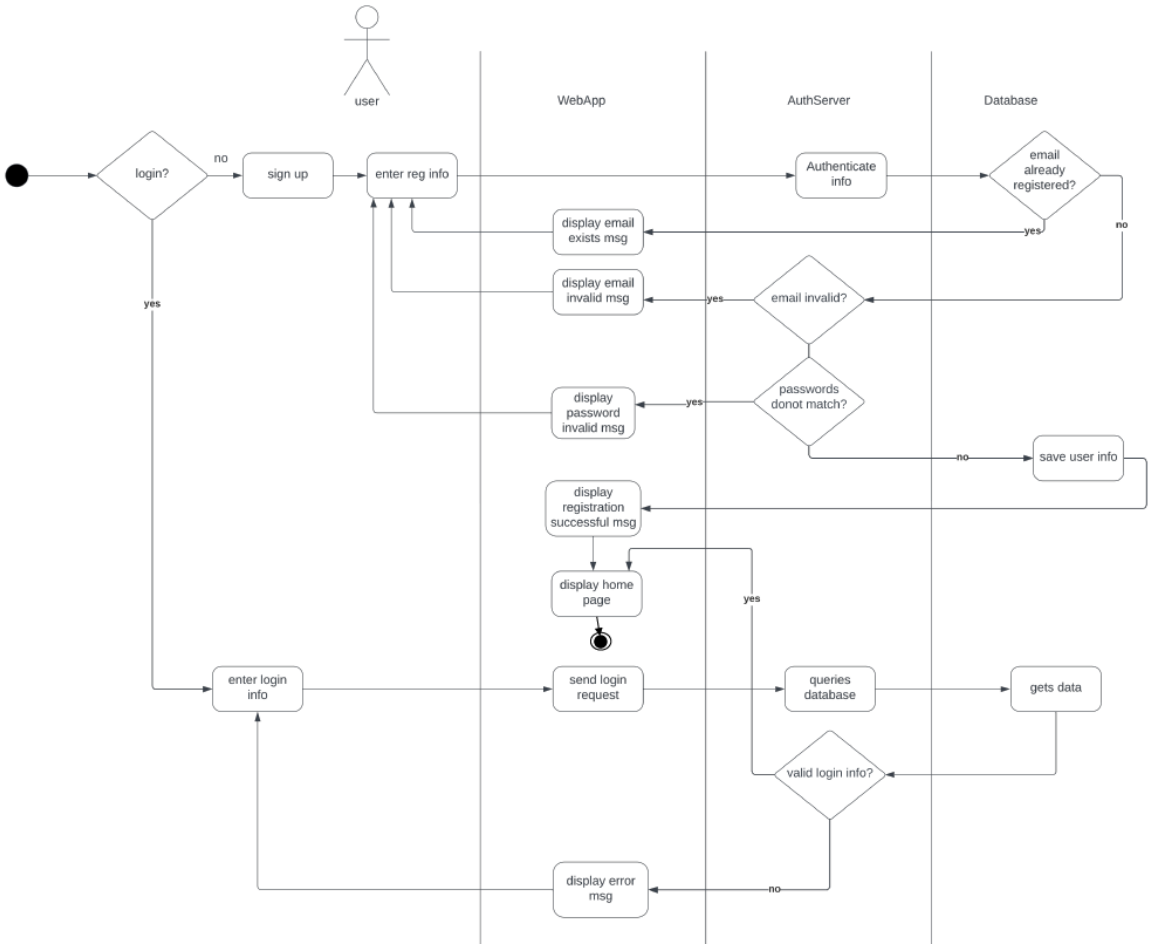
6.5 filter and delete purchase Sequence Diagram:



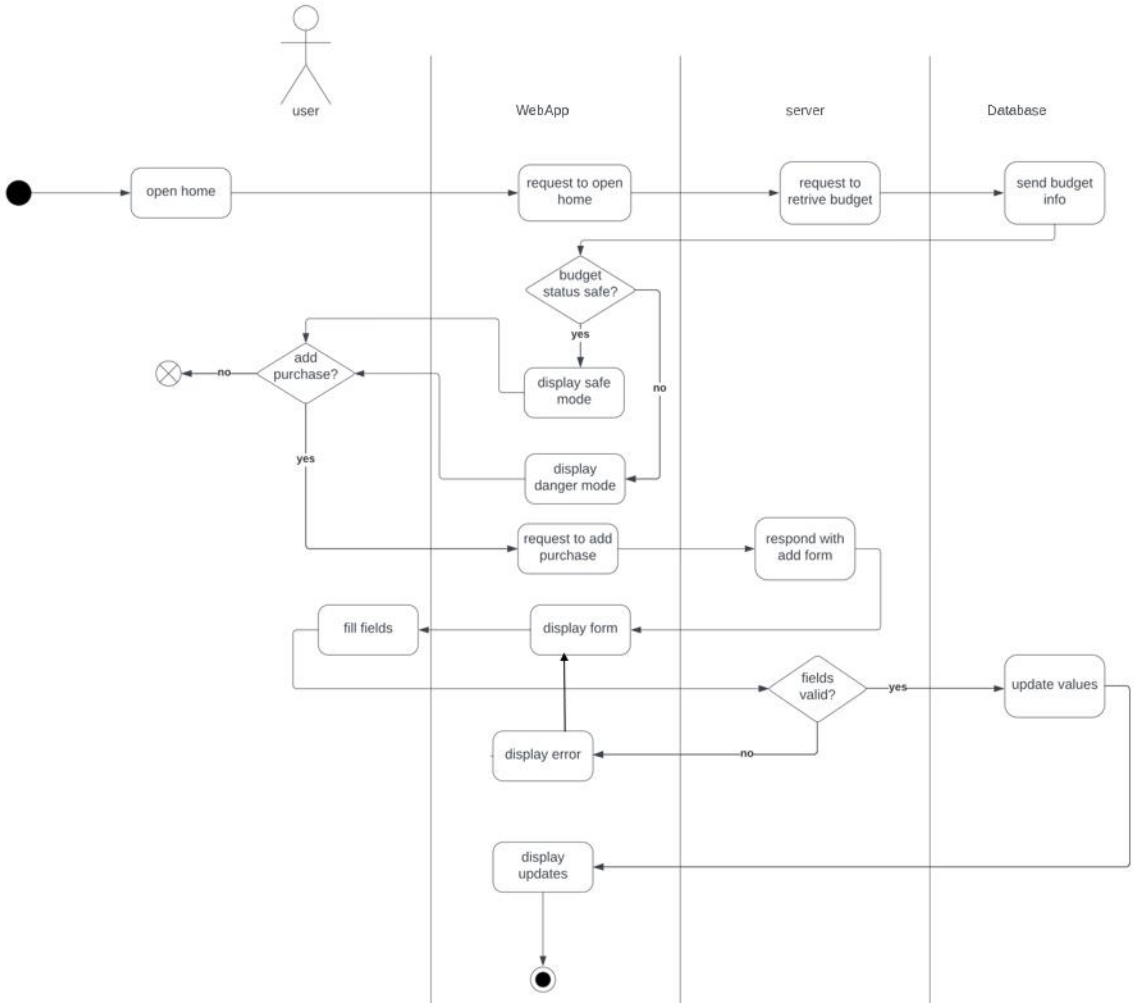
6.6 settings Sequence Diagram:



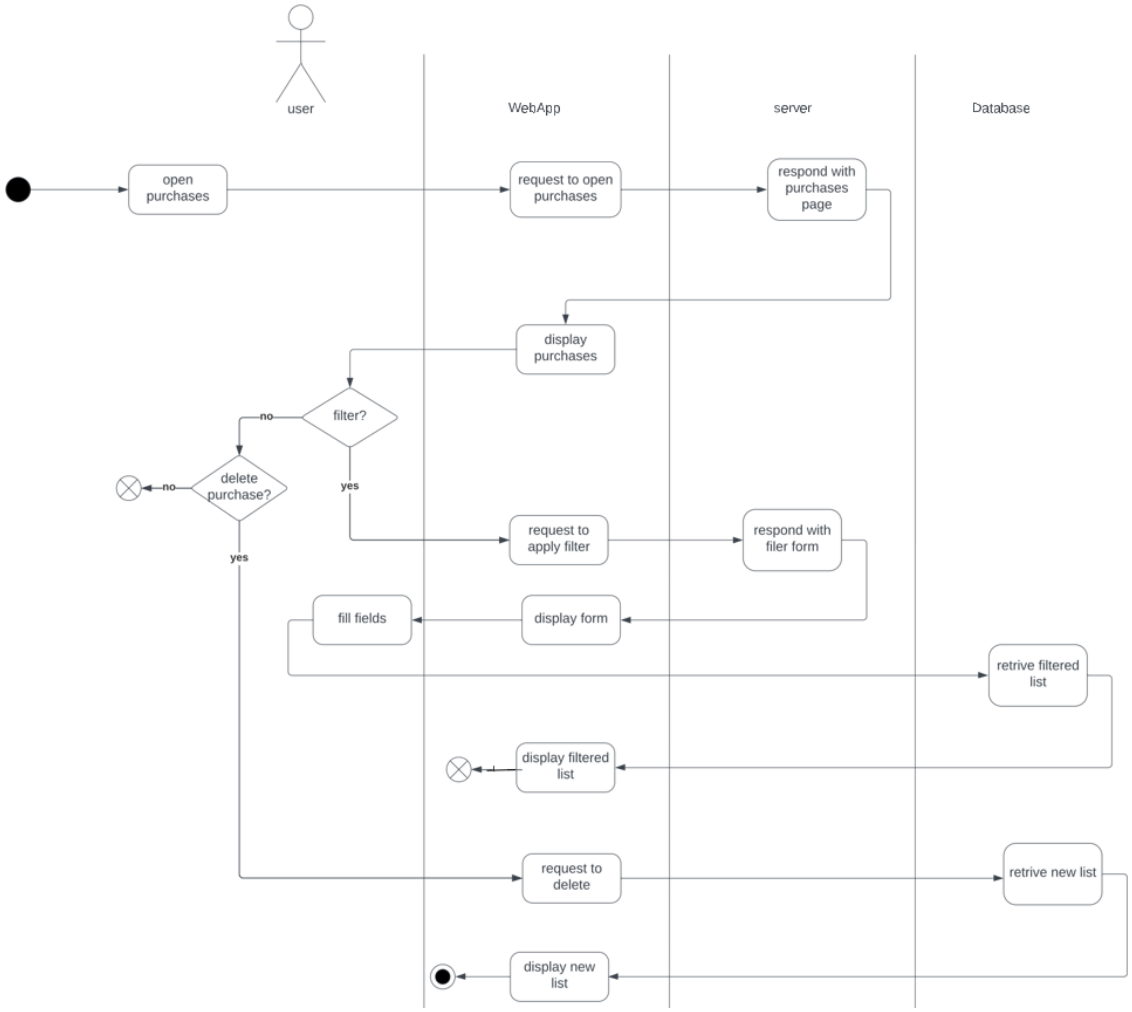
6.7 registration and login activity Diagram:



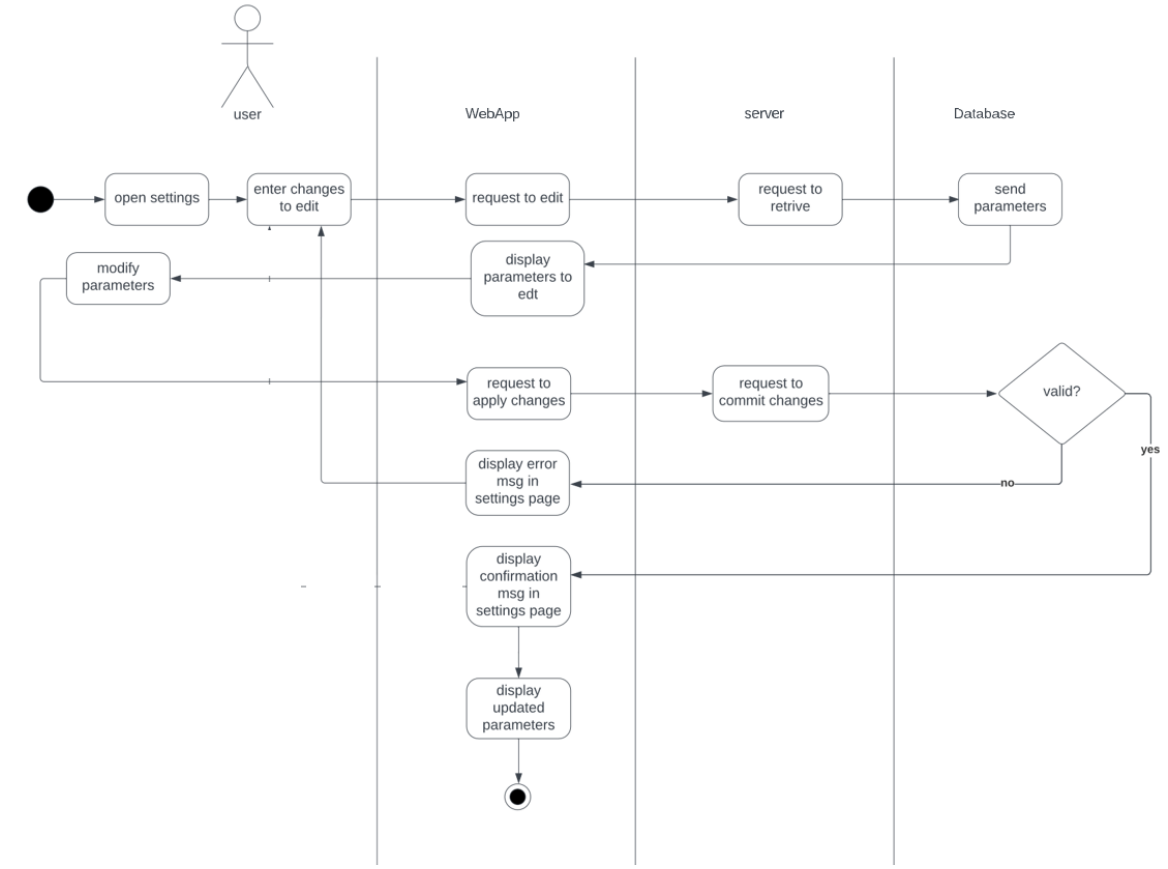
6.8 home and add purchase activity Diagram:



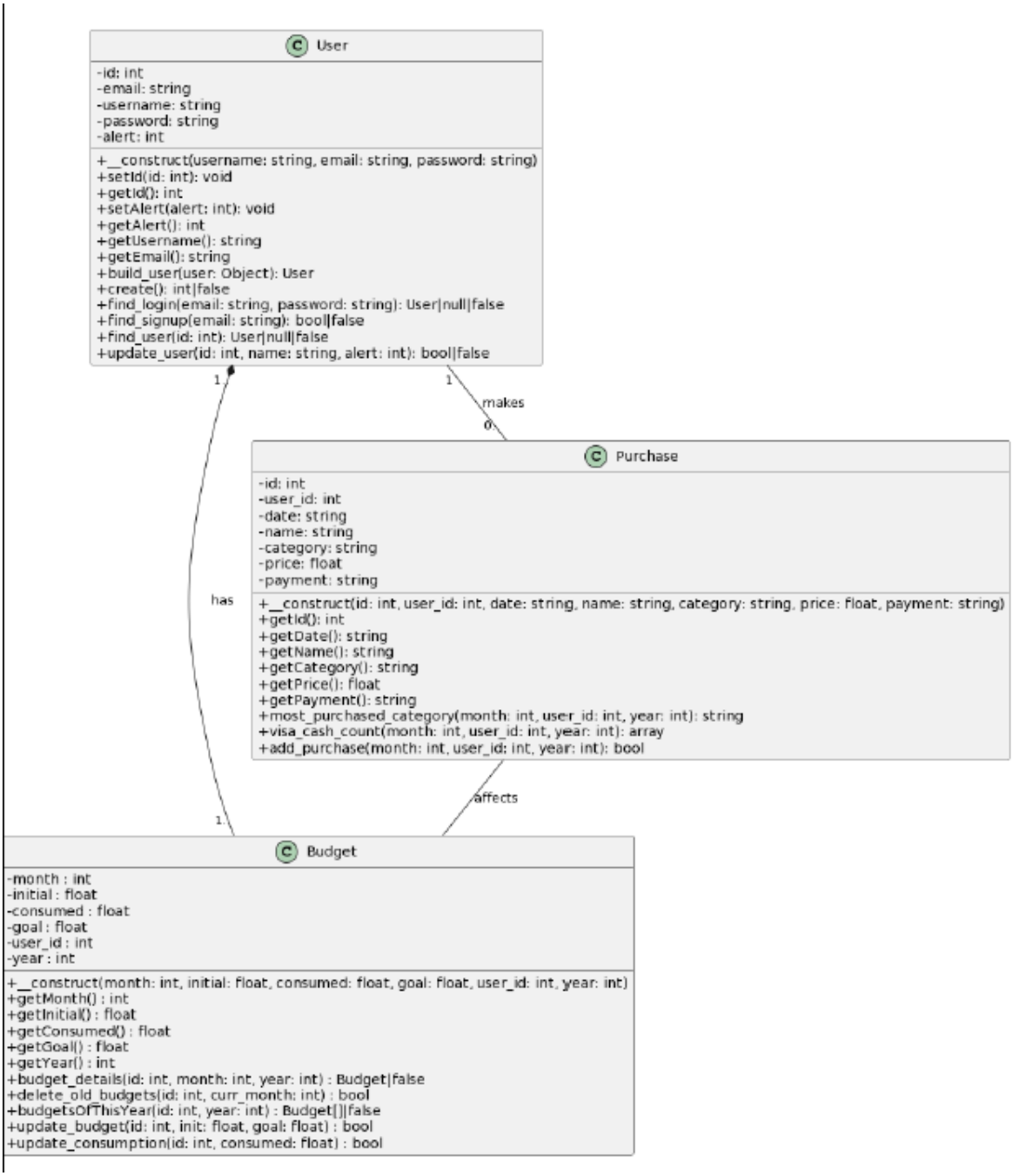
6.9 filter and delete purchases purchase activity Diagram:



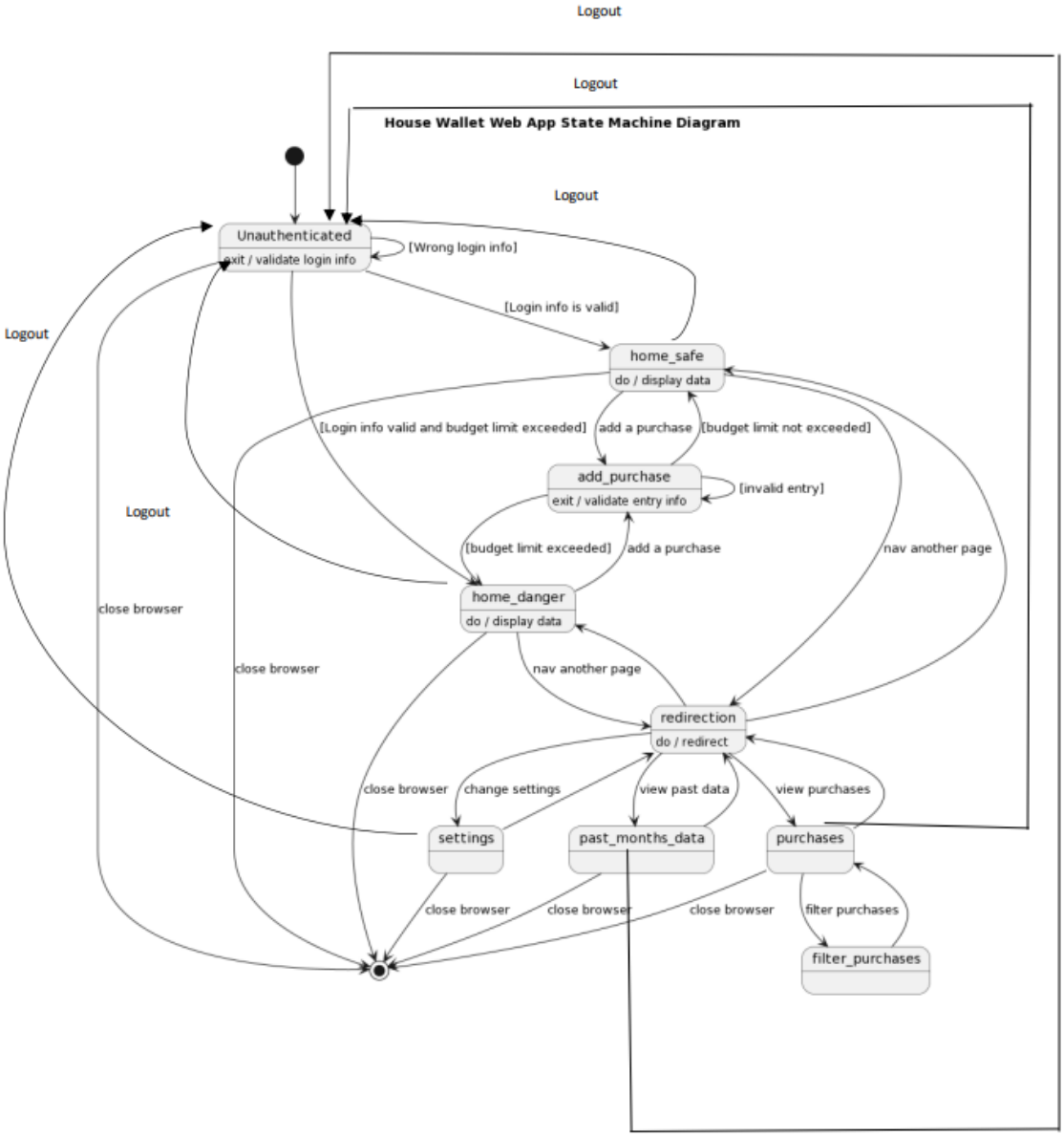
6.10 settings activity Diagram:



6.11 Class Diagram:



6.12 State Machine Diagram:



7. System Evolution

The House Wallet web application will be continually improved and updated to ensure it meets the changing needs of users. The following is a list of planned updates, upgrades, and maintenance for the system:

7.1 Planned Updates • Adding new features such as investment tracking and integration with external accounts. • Enhancing user experience by providing more options for customizing the interface and improving user support. • Improving performance and scalability as the number of users and data volume increases.

7.2 Upgrades • Upgrading the database management system to handle larger volumes of data. • Upgrading the user interface to a more modern design.

7.3 Maintenance • Routine maintenance will be carried out to ensure that the application runs smoothly and to fix any bugs and security issues. • Regular backups of the database will be taken to prevent data loss. • Periodic security audits will be conducted to ensure that the application remains secure.

.

8. Appendices

8.1 Use Cases Use cases describe the different scenarios in which users will interact with the system, including the steps involved and the expected outcomes.

8.2 User Stories User stories are short, simple descriptions of a feature or functionality from the user's perspective. They help to define the requirements of the system and ensure that they meet the needs of the user.

8.3 Data Flow Diagrams Data flow diagrams are visual representations of the flow of data through the system. They show how data is input, processed, and outputted by the different components of the system.