**Software Design and Implementation**

Personal Finance Management App

Version <1.0>

Group Name: B7

Chu Chun Yin 1155194319 1155194319@link.cuhk.edu.hk

Ko Kam San 1155194979 1155194979@link.cuhk.edu.hk

Lai Tsz Hin 1155194965 1155194965@link.cuhk.edu.hk

Chan Yiu Cheung 1155193060 1155193060@link.cuhk.edu.hk

Instructor: Dr. Lam Tak Kei

Course: CSCI3100

Department: Computer Science and Engineering

Printing Date: 11/3/2025

Contents

[1. Introduction 3](#_Toc192625624)

[1.1 Document Purpose 3](#_Toc192625625)

[1.2 Requirement Overview 3](#_Toc192625626)

[1.3 Target Audience and Document Overview 3](#_Toc192625627)

[1.4 Definitions, Acronyms and Abbreviations 4](#_Toc192625628)

[1.5 Document Conventions 4](#_Toc192625629)

[2. System Architecture 5](#_Toc192625630)

[3. Data Model 6](#_Toc192625631)

[3.1 Data Flow 6](#_Toc192625632)

[3.2 Data Schema 6](#_Toc192625633)

[4. Interface Design 7](#_Toc192625634)

[4.1 Communication Protocol 7](#_Toc192625635)

[4.2 Security 7](#_Toc192625636)

[4.3 Client-Server API 7](#_Toc192625637)

[5. Component Design 8](#_Toc192625638)

[5.1 Login System 8](#_Toc192625639)

[5.2 Registration and Authorization System 9](#_Toc192625640)

[5.3 Expense Tracking System 10](#_Toc192625641)

[5.4 Budget Management System 11](#_Toc192625642)

[5.5 Report Generator System 11](#_Toc192625643)

[6. User Interface Design 12](#_Toc192625644)

[6.1 Header and Navigation Bar 12](#_Toc192625645)

[6.2 Dashboard 12](#_Toc192625646)

[6.3 Expenses, Budgets, Reports, and Settings 13](#_Toc192625647)

# 1. Introduction

## 1.1 Document Purpose

This document provides a comprehensive design and implementation for the Personal Finance Management Web App. Its purpose is to contribute to the technical guide for developers. Developers and product reviewers can get a clear breakdown of System Architecture, data model, interface design, component design, and user interface design of this application so this document can help them to develop this application from scratch.

## 1.2 Requirement Overview

The main purpose of this Personal Finance Management Web App is helping users to track their finance usage and getting better insights of personal finance.

Main Features:

* Track, record, and categorize user’s daily expense
* Set up budget goal with a real-time alert to prevent the spending from reaching the limit.
* Having better visualization on personal finance report
* Getting notification while the budget limit is approached

These Features can benefit users from more straightforward to manage their finance. Users can gain more awareness of finance management.

|  |  |
| --- | --- |
| **Technique Constraints** | **Background** |
| Hardware Limitations | The app must be responsive and performant on both desktop and mobile devices. |
| Technologies and Tools | UML modeling language for system design and documentation. Front-end development using React.js and Back-end development using Node.js and Express. Database management using PostgreSQL |
| Security Considerations | Implement strong encryption for data at rest and in transit and ensure compliance with GDPR and other relevant data protection regulations |
| Design Conventions and Programming Standards | All API follow REST style design principles. We adhere to coding standards and best practices for JavaScript and SQL. |

## 1.3 Target Audience and Document Overview

The intended Audience of this document:

* Developers: Receiving technical details for design, architecture, and implementation of the application to implement those technology
* Product reviewers: Testing the application’ features to check the quality and scalability

This design and implementation document starts with an introduction to provide the purpose and target audience. Then, the second part is about the data flow and data scheme in the application. The fourth part is about the interface design. The fifth part is about the component design which demonstrates the system details. The sixth part is about the user interface design which the application how to arrange the user interface.

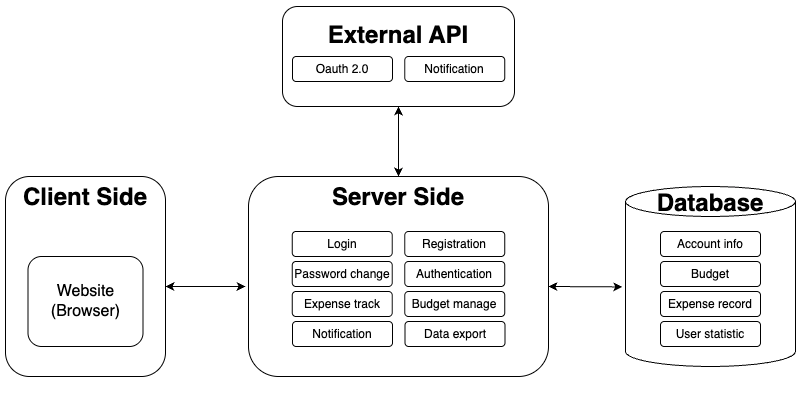
## 1.4 Definitions, Acronyms and Abbreviations

* **API:** Application Programming Interface
* DDOS**:** Denial-of-Service Attack
* HTTP**:** HyperText Transfer Protocol
* JWT**:** JSON Web Token
* REST**:** Representational State Transfer
* SQL**:** Structured Query Language
* SSL**:** Secure Sockets Layer

## 1.5 Document Conventions

* **Formatting:** This document is written in Times New Roman, size 11 for body text. Section headers are bolded and numbered according to IEEE standards
* **Numbering:** All parts in this document are numbered sequentially (e.g., 1.1, 2.1)
* **Terminology:** Terms are defined in Section 1.4 for consistency

# 2. System Architecture



This diagram shows the overall architecture of the software application. The whole application contains three major components, client side, server side and database. For the client side, the website using React.js will display the data received from the server side on the browser. It also handles the user input and request to the server. For the server side built on Node.js, it processes the requests from the client side and executes different functions based on the users’ input. Users can register and login to their accounts. Users can track their expenses and manage their budgets. The system also notifies the user and exports a financial report. Besides, the server connects to the external API such as Google API for Oauth 2.0 authentication and Mailjet API for emailing users. When executing all these functions, the server side stores and request user data including user account information, expense record, users’ budgets and user statistics.

# 3. Data Model

## 3.1 Data Flow

The user enters account information (email, password) and financial data (expense, record) through the client interface. The client sends data to the server. After that, the server validates, processes and stores the data in the database. The server also fetches data from the database for reports and calculations. The processed result will be sent to the client for display.

## 3.2 Data Schema

There are three tables in the database, the user account table, the expense record table and the budget table.

**User Account Table**

|  |  |
| --- | --- |
| Name | Description |
| user\_id | Integer, primary key |
| email | Varchar, unique |
| password | Varchar |
| auth\_token | Text, for social login and JWT |

**Expense Record Table**

|  |  |
| --- | --- |
| Name | Description |
| expense\_id | Integer, primary key |
| user\_id | Integer, foreign key, refer to another table |
| amount | Decimal |
| category | Varchar, type of expense, e.g. “food", "transportation” |
| data | Data |

**Budget Table**

|  |  |
| --- | --- |
| Name | Description |
| budget\_id | Integer, primary key |
| user\_id | Integer, foreign key, refer to another table |
| budget\_limit | Decimal, positive |
| category | Varchar, type of budget, e.g. “food", "transportation” |
| period | Varchar, e.g. “weekly”, “month” |
| alert\_threshold | Decimal, e.g. 80% of budget |

# 4. Interface Design

## 4.1 Communication Protocol

Since this is a web application, HTTP/1.1 will be used as the primary communication protocol between the server component and the client component. For the internal communication between the database component and the server component, the SQLite module of the Node.js standard library will be used with connection pooling enabled, and possibly via an ORM library.

Additionally, since OAuth 2.0 is designed specifically to work with HTTP/1.1, communication between the external API (Google’s authentication server) and our server component will also be using the same protocol.

## 4.2 Security

The communication will be on HTTP over SSL (HTTPS), where the packets transferred between components will be encrypted and prevented from eavesdropping.

JSON Web Token (JWT) and cookies on client-side will be employed for the authentication mechanism in the client-server API, such that only authenticated users can fetch certain data through the API.

Inputs from the API will be validated and sanitized before processing and storing into the database to prevent attacks such as SQL injection. In the database, bcrypt encryption is used for sensitive data e.g., user passwords.

In addition, the API should enforce rate-limiting on server-side to block brute-force and DDOS attacks.

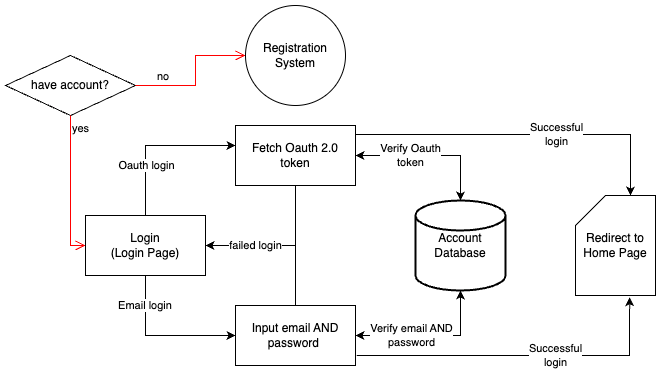
## 4.3 Client-Server API

The API data will be using the JSON format for simplicity, flexibility and maximum compatibility in both the client and the server. The API will follow the REST style to be more intuitive for developers. Below are the major API endpoints and their respective input (parameters) and output (response) examples.

* Expense Tracking: POST /expenses
  + { amount: 50.0, category: "Groceries", date: "2023-10-05", currency: "JPY" }
  + { status: 201, id: 123 }
* Budget Goals: POST /budgets
  + { category: "Entertainment", limit: 200.0, timeframe: "monthly" }
  + { status: 201, id: 123 }
* Statistics Report: GET /reporting
  + { report: "monthly", from: "2023-10-01", to: "2023-10-02" }
  + { categories: ["Food", "Transport"], amounts: [300, 150] }

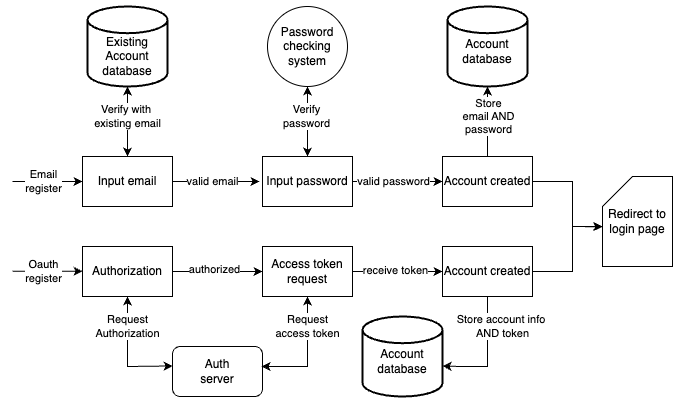
# 5. Component Design

## 5.1 Login System



Users can login with email or social media accounts (through Oauth 2.0). If users login to their accounts successfully, the system will redirect them to the home page. Otherwise, the system will show a notification of the wrong username and password and return to login page.

## 5.2 Registration and Authorization System

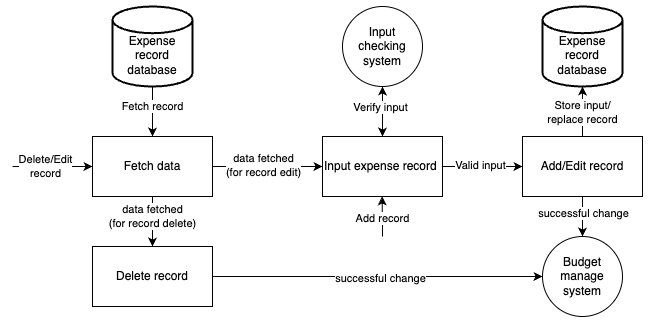


The registration system creates user accounts in two ways, email registration and social media account registration.

For email registration, users input their email, and the system will verify with the existing account database to ensure this email is not registered. Otherwise, the system will show a notification and require the user to input another email. After that, users have to input a password for their account. The password checking system will verify its security (length: 6 – 12, including at least one uppercase letter, lowercase letter and special character) With valid email and password, the system will create an account and store all user information in account database.

For Oauth registration, the system requests authorization for the Auth server. After getting the authorization the system will request the access token. After that the system stores all account information and the token into the account database. If the request fails, the system will send a notification for the user for unsuccessful registration and return to login page. The users have to register again.

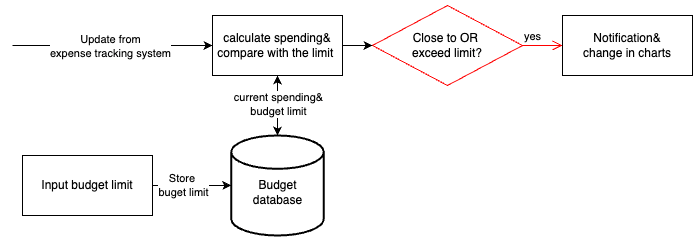
## 5.3 Expense Tracking System



The expense tracking system is to add, edit, delete and list the expense. For deleting records, the system will fetch the record with expense\_id and remove it from the expense record database. For editing data, the system fetches the record and requires the user to input the data. The input checking system will validate the data. After validation, the system will store the new input and replace the old one. For adding records, the system also validates the data and stores the new record. When the expense tracking system and expense record database have a successful change (delete, add, edit), it will trigger the budget management system to update the budget progress.

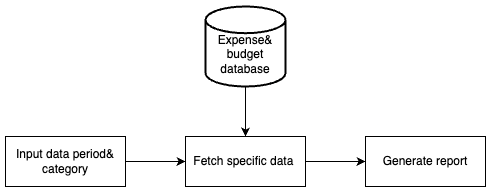
Users can select a specific period or category of the expense data. The system will select specific expense records from the database. After that, it will send these filtered data to the client and display it on the website.

## 5.4 Budget Management System



The budget management is to set the budget goal and notify the users if the expense is close to or exceeds the budget limit. Users can input the budget limits and period for different categories. After receiving updates from the expense tracking system, the system will fetch the current spending with the same types, such as categories and time periods, and calculate the spending for budget comparisons. After that, the system stores the new spending in the budget database for the next calculation. The method only requires the total spending and any updated expense records instead of all expense data. If the new spending is close to or exceeds the budget limit (exceed alert\_threshold and budget\_limit), the system will notify the user and change the chart.

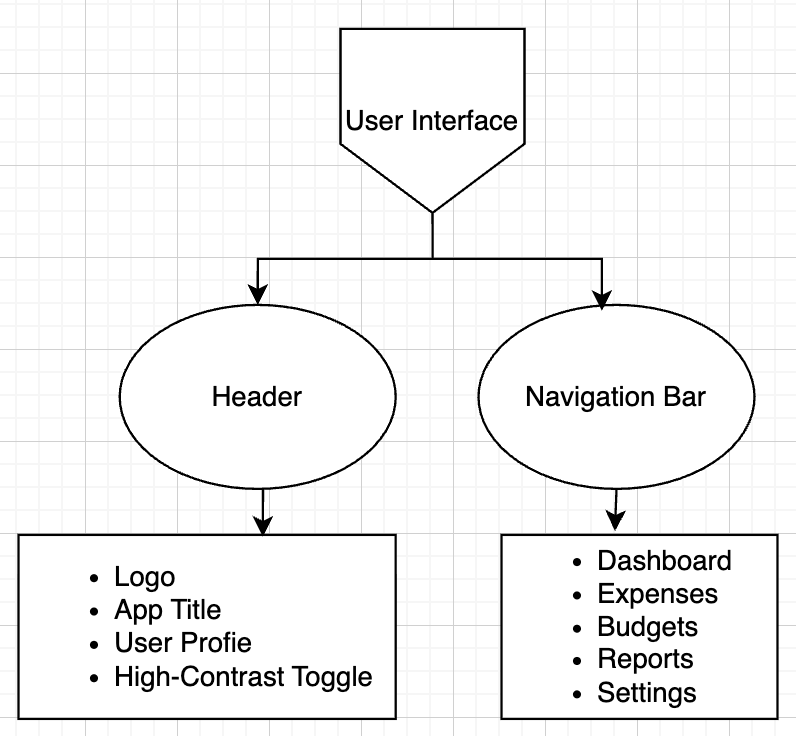
## 5.5 Report Generator System



The report generator system generates a report that allows users to specify data ranges and categories. The system will pull raw expense records and budget limits from the database. After that, the system will convert these data into a report.

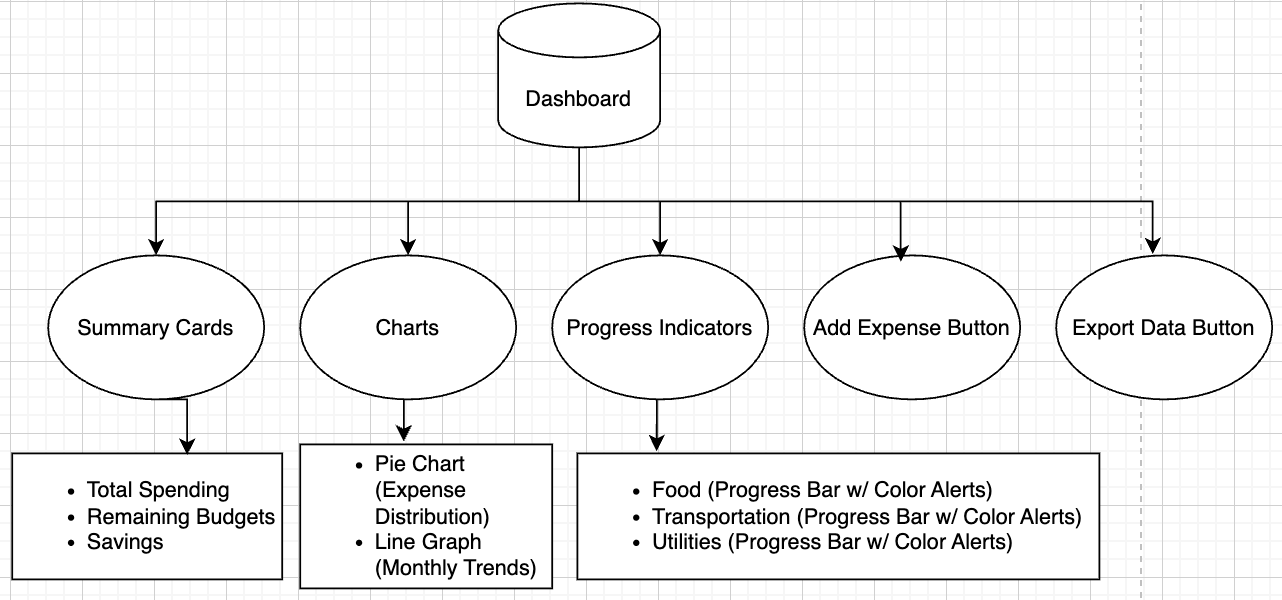
# 6. User Interface Design

## 6.1 Header and Navigation Bar



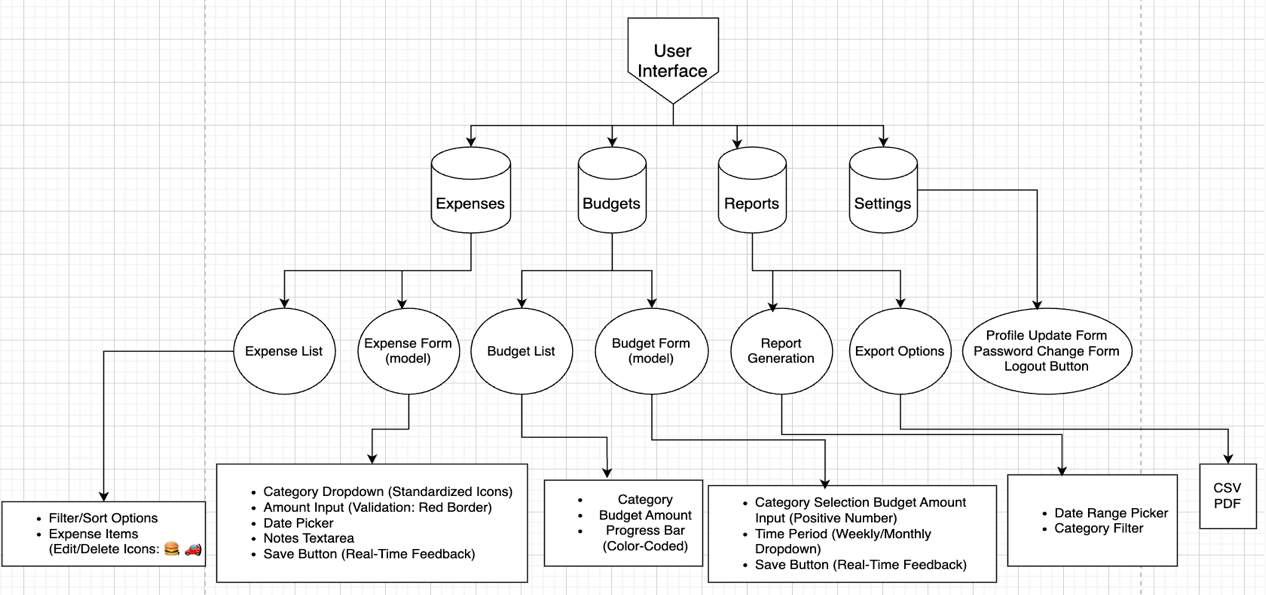
The structural components of the user interface are Header and Navigation Bar. For Header, it contains logo and application title which represent the app’s branding, and the name of the app. Header also contains a user profile to provide personal account information. The High-Contrast Toggle provides an accessibility feature for visually impaired users. For Navigation Bar enables users to navigate between multiple sections, such as expenses, budgets, reports, and dashboard most importantly. Setting also allows the user to customize options.

## 6.2 Dashboard



Dashboard serves as a major section of the entire app, which shows the overview of financial data and actionable insights based on the users’ input information. Summary Cards show total spending, remaining budgets, and savings. Certain charts or graphs such as pie charts and line graphs could visualize the data. Multiple Progress Indicators track the expenses of different categories on food, transportation, utilities and so on, with progress bar and color alert. Besides, the Add Expense Button enables users to add a new expense category easily. Export Data Button allows exportation of certain graphs or reports.

## 6.3 Expenses, Budgets, Reports, and Settings



Additionally, the other four UI designs are also essential for the app, including expenses, budgets, report and setting. Those contain specific functionalities.

The Expenses section contains expense lists which can filter the expense option and design the expense items by emojis. Also, the expense forms (model) allow users to add new expenses based on fields like category dropdown, amount input, date picker, notes text area, and save button. The Budgets section allows users to set budget amount for their categories with progress bar through budget list. Also, the budget forms (model) return the budget amount user input with certain time interval (weekly, monthly), and save. The Reports section generates reports with charts or graphs to users using data range picker and category filter, then those reports could be exported for eternal usage using CSV or PDF formats. Finally, the Setting section allows users to update their personal information, such as username and password. Also, it provides login and logout functions for the app.