成都理工大學 中英合作办学
CDUT Sino-British Collaborative Education

OXFORD BROOKES UNIVERSITY

UNDERGRADUATE PROJECT REPORT

Project Title:	**(Web Based) Comprehensive Personal Expenses Tracking System**
Surname:	**Wang**
First Name:	**Yingrui**
Student Number:	**202018020316**
Supervisor Name:	**Maged Refat**
Module Code:	**CHC 6096**
Module Name:	**Project**
Date Submitted:	**May 5, 2023**

Chengdu University of Technology Oxford Brookes College

Chengdu University of Technology

BSc (Single Honours) Degree Project

Programme Name: **Software Engineering**

Module No.: **CHC 6096**

Surname: ………………………………Wang…………………………………………………………

First Name: ……………………………Yingrui…………………………………………………………

Project Title: ………………Comprehensive Personal Expenses Tracking System………………

Student No.: …………………………202018020316…………………………………………………

Supervisor: ……………………………Maged Refat……………………………………………………

2ND Supervisor (if applicable): **Not Applicable**

Date submitted: **May 5, 2023**

A report submitted as part of the requirements for the degree of BSc (Hons) in Software Engineering

At

Chengdu University of Technology Oxford Brookes College

Declaration

Student Conduct Regulations:

Please ensure you are familiar with the regulations in relation to Academic Integrity. The University takes this issue very seriously and students have been expelled or had their degrees withheld for cheating in assessment. It is important that students having difficulties with their work should seek help from their tutors rather than be tempted to use unfair means to gain marks. Students should not risk losing their degree and undermining all the work they have done towards it. You are expected to have familiarised yourself with these regulations.

https://www.brookes.ac.uk/regulations/current/appeals-complaints-and-conduct/c1-1/

Guidance on the correct use of references can be found on www.brookes.ac.uk/services/library, and also in a handout in the Library.

The full regulations may be accessed online at https://www.brookes.ac.uk/students/sirt/student-conduct/

If you do not understand what any of these terms mean, you should ask your Project Supervisor to clarify them for you.

I declare that I have read and understood Regulations C1.1.4 of the Regulations governing Academic Misconduct, and that the work I submit is fully in accordance with them.

Signature ……………… Nelson …………………………..
Date ……………………2024.5.3…………………………………………

REGULATIONS GOVERNING THE DEPOSIT AND USE OF OXFORD BROOKES UNIVERSITY MODULAR PROGRAMME PROJECTS AND DISSERTATIONS

Copies of projects/dissertations, submitted in fulfilment of Modular Programme requirements and achieving marks of 60% or above, shall normally be kept by the Oxford Brookes University Library.

I agree that this dissertation may be available for reading and photocopying in accordance with the Regulations governing the use of the Oxford Brookes University Library.

Signature …………………… Nelson ……………………..
Date …………………………2024.5.3……………………………………..

Acknowledgment

First and foremost, I would like to express my sincere gratitude to the teachers who have helped me and taught me, as well as to my supervisor. My supervisor Maged Refat gave me a lot of help during my thesis and graduation project. When I was confused about the direction of my thesis and graduation project, he used his professional knowledge to guide me in the right direction and provided comprehensive and detailed guidance from the beginning of the research work to the completion of the thesis. Appreciate his patience and support.

Additionally, I want to thank my family and parents for their unwavering love and support. I've been able to advance in my studies consistently, learn more about myself, and have unending bravery and strength to pursue my aspirations thanks to your selfless love and encouragement.

In addition, I would also like to thank my classmates and friends for their support and help in my personal and academic life, which allowed me to experience the intimacy of friendship and the bonds of family affection, and gave me inspiration and comfort while I was studying.

In conclusion, I would like to express my heartfelt gratitude to all those who have supported me throughout this project. Their help and support provided me with important motivation and assistance in completing this graduation thesis.

Table of Contents

Abstract

Effective management of personal finances is essential for financial stability and prudent decision-making. However, the complexity of tracking expenses and income poses a significant challenge for many individuals. In response to this challenge, this paper proposes the development of a web-based Comprehensive Personal Expenses Tracking System.

The aim of this project is to create an intuitive, user-friendly platform that empowers users to efficiently track and manage their personal finances. Through extensive research and analysis of existing personal finance tracking websites, the system's objectives were delineated, including user-friendly interface design, login and registration functionality, expense recording, statistical chart display, expense classification, and user management features.

Implemented using a B/S architecture with Java programming language and Springboot framework, the system provides a secure and efficient platform for users to monitor their financial transactions remotely. Key features include expense categorization, customizable expense categories, and statistical analysis of income and expenditure over time.

The methodology employed a Waterfall Model, encompassing requirements gathering, design, implementation, testing, deployment, evaluation, and operation maintenance. The project adhered to a structured schedule and version management through Git.

Implementation results demonstrate successful registration, login, bill management, expense categorization, and statistical analysis functionalities. Future work includes comprehensive testing and presentation to the audience.

Keywords: Personal Finance Management, Expense Tracking System, Web-based Platform, User Interface Design, Financial Literacy, Data Protection, Ethical Considerations, Environmental Sustainability.

Abbreviations

B/S architecture: Browser/Server architecture

CPU: Central Processing Unit

GPU: Graphic Processing Unit

MVC: Model View Controller

GDPR: General Data Protection Regulation

CCPA: California Consumer Privacy Act

WCAG: Web Content Accessibility Guideline

UI: User Interface

CCPA: California Consumer Privacy Act

JDBC: Java Data Base Connectivity

TDD: Test-Driven Development

CI: Continuous Integration

Glossary

Personal Finance Management: Personal finance management refers to the process of planning, organizing, and controlling one's financial activities, including budgeting, saving, investing, and spending, to achieve financial goals and objectives.

Expense Tracking System: An expense tracking system is a software application or platform designed to help individuals or businesses monitor, record, and manage their expenses. It typically allows users to input and categorize expenses, generate reports, and analyze spending patterns.

Web-based Platform: A web-based platform is a software system or application that operates primarily over the internet or an intranet. Users access the platform through a web browser, eliminating the need for locally installed software. Web-based platforms often offer features such as scalability, accessibility from any device with internet connectivity, and centralized data storage.

User Interface Design: User interface design is the process of designing the visual layout and interactive elements of a software application or website. It focuses on creating an intuitive and aesthetically pleasing interface that enhances user experience and usability.

Financial Literacy: Financial literacy refers to the knowledge and understanding of financial concepts, such as budgeting, saving, investing, debt management, and financial planning. It enables individuals to make informed decisions about their finances and effectively manage their money.

Data Protection: Data protection involves implementing measures to safeguard sensitive and confidential information from unauthorized access, use, disclosure, alteration, or destruction. It includes policies, procedures, and technologies designed to protect data privacy and ensure compliance with relevant laws and regulations.

Ethical Considerations: Ethical considerations in the context of software development and personal finance management involve addressing moral principles and values related to privacy, transparency, fairness, and accountability. It includes assessing the potential impact of the system on users and stakeholders and ensuring ethical behavior throughout the development and deployment process.

Environmental Sustainability: Environmental sustainability refers to practices and initiatives aimed at minimizing the negative impact of human activities on the environment and natural resources. In the context of personal finance management, it may involve promoting eco-friendly spending habits, investing in sustainable products or companies, and reducing carbon footprint.

Chapter 1 Introduction

1.1 Background

As societies evolve, the effective management of personal finances becomes increasingly paramount. Individuals often grapple with the complexities of monitoring their expenses and incomes, exacerbated by the multitude of income sources, diverse expenditure patterns, and the imperative to maintain financial stability (Gupta et al., 2020). In response to these pervasive financial challenges and the growing need for individuals to promptly assess their financial status and exert control over their fiscal health, the development of a web-based Comprehensive Personal Expenses Tracking System emerges as a crucial endeavor.

Traditional methods of financial management, such as manual tracking or spreadsheet-based systems, are often cumbersome and prone to errors. Moreover, they may lack the necessary features to provide comprehensive insights into one's financial situation. In this context, the proposed system aims to streamline personal expenditure and income tracking processes, offering users a user-friendly platform to monitor their financial transactions efficiently.

Existing personal finance tracking tools offer some degree of functionality, but often fall short in providing a comprehensive solution that meets the diverse needs of users. A critical analysis of these tools reveals varying levels of feature completeness, ranging from basic functionalities such as bill management to more advanced capabilities like data analysis and synchronization with bank accounts (Gomathy, 2022; Patil et al., 2023). By synthesizing insights from existing platforms and addressing their limitations, the proposed Comprehensive Personal Expenses Tracking System seeks to offer a robust solution that empowers users to make informed financial decisions and achieve greater fiscal well-being.

1.2 Aim

The aim of this project is to design and develop a sophisticated yet user-friendly web-based Comprehensive Personal Expenses Tracking System. The aim is to create a platform that revolutionizes the way individuals manage their personal finances, offering an intuitive and versatile solution tailored to meet the diverse needs of users across various demographics. By web technologies and design principles, the system endeavors to provide users with a seamless and efficient means of tracking, analyzing, and optimizing their financial activities.

1.3 Objectives

The objectives are as follows:

(1) Complete a background check on personal expenditure and income tracking websites.
(2) Understand the technology required for the website and determine project requirements.
(3) Design an intuitive interface for users to access and use easily.
(4) Implement login and registration functions.
(5) Implement Bill recording function.
(6) Implement bill search function.
(7) Implement bill modification function.
(8) Implement bill deletion function
(9) Implement expense classification function and customize expense categories.
(10) Realize the statistical chart display function of user expenses.
(11) Display user's income and expenditure statistics over a period of time.
(12) Implement the function of adding home users.
(13) Implement user password change function.
(14) Implement user exit function.
(15) Carry out test.
(16) Present the work to the audience.

Conduct Comprehensive Background Research	a) Undertake an exhaustive review of existing personal expenditure and income tracking websites, analyzing their features, functionalities, user interfaces, and shortcomings. b) Document key insights regarding the effectiveness and limitations of

	current platforms to inform the development process effectively.
Understand the Technology and Determine Project Requirements	a) Identify the technological stack required for the web-based system, including programming languages, backend and frontend frameworks, databases, and development tools. b) Gather and document project requirements through user feedback, surveys, and market research, ensuring alignment with user needs and project objectives.
Design an Intuitive User Interface	a) Create wireframes and mockups to visualize the user interface design, focusing on ease of navigation, accessibility, and aesthetic appeal. b) Incorporate user-centered design principles to ensure a seamless and intuitive user experience across various devices and demographics.
Implement User Authentication and Registration Functions	a) Develop robust login and registration functionality, ensuring secure access to the system and protection of user data through encryption and authentication mechanisms. b) Validate user inputs and enforce password complexity requirements to enhance security and mitigate the risk of unauthorized access.
Implement Expense Tracking and Management Functions	Develop features for recording, categorizing, and managing expenses and incomes, allowing users to track their

	financial transactions with accuracy and granularity.
Realize Statistical Analysis and Visualization Features	a) Design and implement dynamic charts, graphs, and reports to provide users with visual representations of their expenditure patterns, income sources, and financial trends over time. b) Enable users to customize chart parameters and explore detailed insights into their financial data to facilitate informed decision-making and goal setting
Implement User Management and Customization Features	a) Develop functionality for managing user profiles, including adding new users, changing passwords, and securely exiting accounts. b) Allow users to customize expense categories and consumption types to align with their unique financial habits and preferences.
Conduct Comprehensive Testing and Quality Assurance	a) Conduct rigorous testing across all stages of development, including unit testing, integration testing, and user acceptance testing, to ensure the reliability, security, and usability of the system. b) Address any identified bugs, glitches, or performance issues promptly to deliver a stable and polished product to end users.

Present the Work to the Audience	a) Organize presentations or demonstrations to showcase the completed project to stakeholders, including project sponsors, potential users, and industry professionals. b) Solicit feedback and suggestions from the audience to validate the project's effectiveness and gather insights for future enhancements and iterations.
Project Documentation and Version Management	a) Maintain comprehensive documentation of project processes, methodologies, and outcomes, ensuring transparency and accountability throughout the development lifecycle. b) Utilize version control systems such as Git for efficient collaboration, code management, and version tracking, facilitating seamless project management and coordination among team members.

Table 1. Expand Objectives Table

1.4 Project Overview

1.4.1 Scope

The scope of the software development project is extensive and multifaceted, centered around the creation of a sophisticated web-based Comprehensive Personal Expenses Tracking System. This system aims to address the burgeoning need for efficient personal finance management solutions in today's dynamic economic landscape. With an emphasis on user-centric design and robust functionality, the software endeavors to revolutionize the way individuals manage their financial affairs.

At its core, the Comprehensive Personal Expenses Tracking System offers a comprehensive suite of features designed to streamline and optimize the process of monitoring daily expenses and incomes. Users can expect a user-friendly interface that facilitates seamless navigation and accessibility, ensuring an intuitive and engaging experience. Key functionalities include the ability to add, delete, modify, and view financial transactions, along with advanced capabilities such as categorizing expenses and managing default expense categories.

Furthermore, the system goes beyond basic expense tracking by providing users with valuable insights into their financial habits and trends. Through dynamic charts, graphs, and reports, individuals can gain a deeper understanding of their expenditure patterns, income sources, and financial trends over time. This analytical component empowers users to make informed decisions, identify areas for improvement, and set achievable financial goals.

From a technical standpoint, the software operates on a robust B/S architecture, leveraging industry-standard technologies to ensure reliability, scalability, and security. The use of the Java programming language, coupled with the Spring Boot framework and MySQL database, underscores the system's commitment to performance and efficiency. Moreover, adherence to MVC design principles ensures clear separation of concerns, facilitating easier maintenance and future enhancements.

1.4.2 Audience

The target audience for the Comprehensive Personal Expenses Tracking System is diverse and inclusive, encompassing a wide range of individuals seeking effective solutions for managing their personal finances. This audience segmentation includes:

Company Employees: Professionals from various industries who aim to maintain a clear overview of their personal finances alongside their professional commitments. This includes individuals looking to budget effectively, save for future goals, and track business-related expenses separately.

Students: Young adults navigating the complexities of financial independence, including managing student loans, budgeting for tuition and living expenses, and planning for post-graduation financial stability. The software provides a valuable tool for fostering financial literacy and responsibility among student populations.

Working Professionals: Individuals with diverse income streams, including salaried employees, freelancers, and gig economy workers. The software caters to the unique financial management needs of this audience segment, offering features such as income tracking, expense categorization, and budgeting tools to support their financial goals.

Entrepreneurs: Business owners and startup founders who require robust financial tracking solutions to manage both personal and business expenses. The software's ability to synchronize bank accounts, generate detailed expense reports, and facilitate tax preparation makes it an indispensable tool for entrepreneurial endeavors.

General Users: Anyone seeking to gain better control over their finances and make informed decisions about spending, saving, and investing. Whether individuals are planning for major life events, such as buying a home or starting a family, or simply aiming to build long-term financial security, the Comprehensive Personal Expenses Tracking System provides the necessary tools and insights to support their journey towards financial well-being.

By catering to the diverse needs and preferences of its target audience, the software aims to democratize access to effective personal finance management tools, empowering individuals of all backgrounds to take control of their financial futures.

Chapter 2 Background Review

The Comprehensive Personal Expenses Tracking System helps to manage our income and expenses daily or regularly, or to remind us at any time (Gomathy, 2022). The fee tracker can help users track their consumption habits and effectively manage their finances, which can help users determine the priority of spending and make wise decisions on resource allocation (Patil et al., 2023).

The comparison in Table 1 shows that various accounting websites have developed relatively complete functions in adding bills, viewing bills, modification of bill information, and bill reports, indicating that these functions are basic functions at the same level as login and registration. However, the import bill data and account information management functions do not cover all platforms, indicating that these functions are secondary. Most platforms do not implement the bill data analysis and synchronize bank card information function, which means that this function is a subsidiary function with low importance.

Website name / Website Features	Zoho Expense	Sui shouji	Wa cai	Ai jizhang
Add bill	√	√	√	√
View bill	√	√	√	√
Modification of bill information	√	√	√	√
Bill data analysis	√	N/A	N/A	N/A
Import bill data	√	N/A	N/A	√
Bill Report	√	N/A	√	√

Account information management	N/A	√	N/A	√
Synchronize bank card information	√	N/A	N/A	N/A

Table 2. Features Comparison of Different Websites

Chapter 3 Methodology

3.1 Approach

3.1.1 Development Model

With the Waterfall Model, the logical implementation is separated from the physical implementation with structured analysis and design methods. In this project, the software life cycle is divided into seven basic activities: requirements gathering, design, implementation, testing, deployment, evaluation, and operation maintenance. As shown in Figure 1.

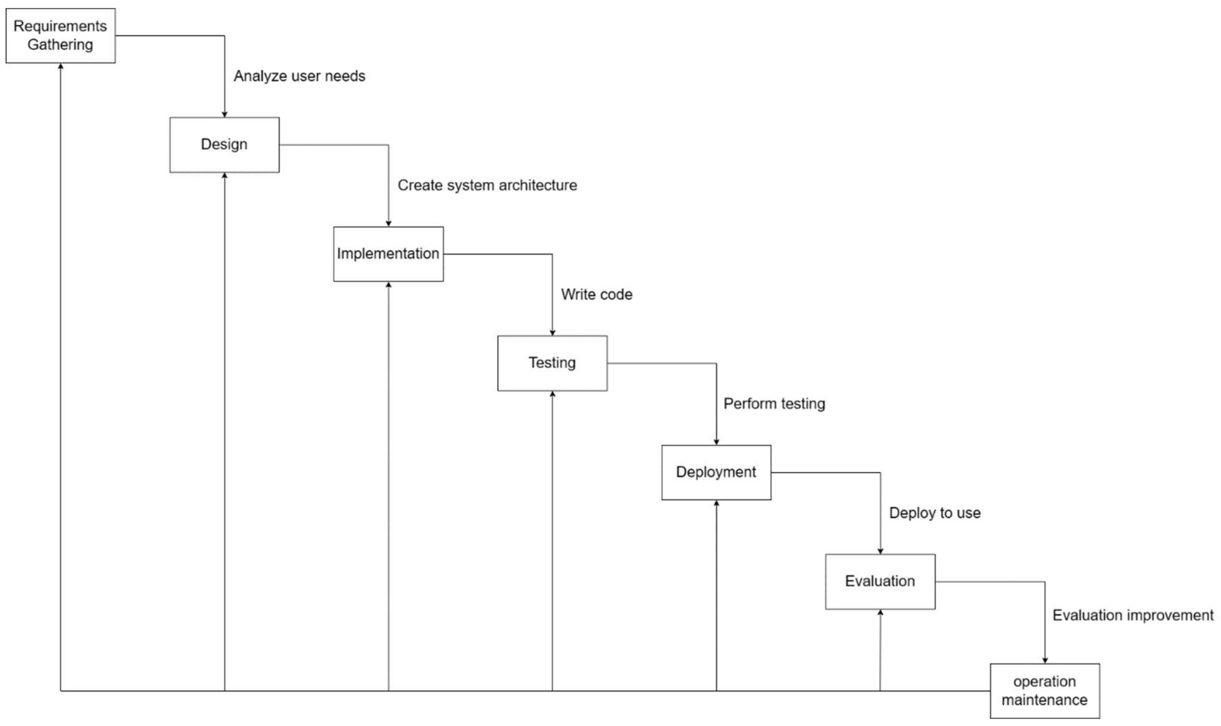


Figure 1. Software Development - Waterfall Model

Our approach will involve the following key steps:

1) Requirement Gathering: We will conduct interviews and surveys with potential users to collect their input and preferences.
2) Design: Create wireframes and mockups to visualize the user interface and overall system architecture.

3) Development: Implement features according to user stories and iterative sprints.
4) Testing: Conduct regular testing, including unit testing, integration testing, and user acceptance testing.
5) Deployment: Regularly deploy updates to a test environment for user feedback and then to a production environment.
6) Evaluation: Collect user feedback and measure system performance.
7) Operation maintenance: Make system changes and maintenance based on user feedback.

3.1.2 Database Design

This database is designed for a financial management system that tracks transactions, categorizes expenses, and manages users and households. It has four main tables: \`bill\`, \`consumption_type\`, \`house\`, and \`user\`.

\`bill\` Table: Stores financial transactions, including consumption and revenue. Each record has an amount, related user, associated house, and a type indicating whether it's consumption or revenue. It also has timestamps for creation and updates, enabling tracking of changes over time.

\`consumption_type\` Table: Contains different types of consumption, such as food, utilities, and entertainment. This table allows flexible categorization of expenses, facilitating easier analysis of spending patterns.

\`house\` Table: Represents households or housing units. It tracks information like the house's administrator, budget, and timestamps for creation and updates. This table allows you to manage multiple households within the system, each with unique budgets and administrators.

\`user\` Table: Stores user details, including names, passwords, and roles. It also identifies whether a user is an administrator and links each user to a specific house.

This structure supports role-based access and management, allowing control over who can perform certain actions.

This database setup is ideal for a financial management system or a budgeting application. It allows for comprehensive tracking of financial transactions, flexible categorization of expenses, management of multiple households, and role-based user access. With this design, you can track spending, manage budgets, and control user permissions efficiently, making it a robust solution for personal finance or household budgeting.

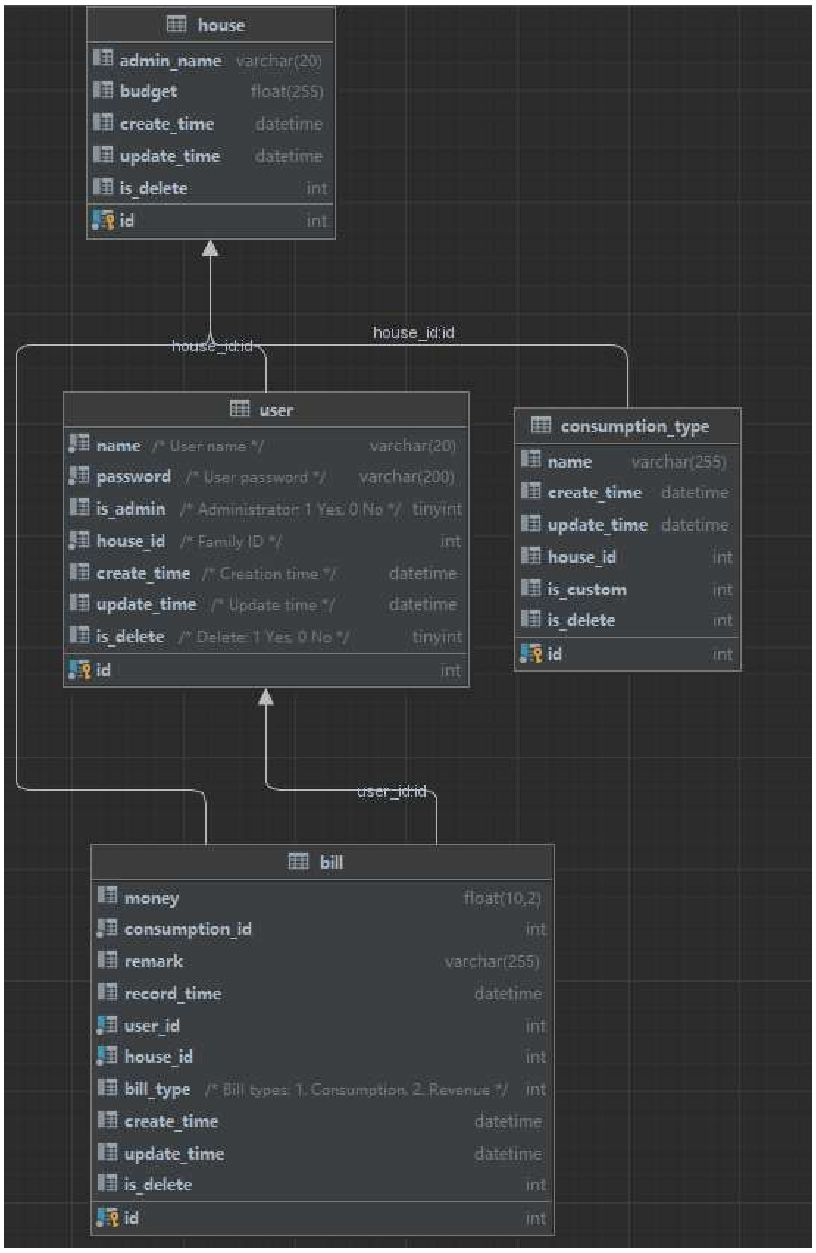


Figure 2. Database Diagram

3.1.3 Requirement Gathering Method

1) User survey: User surveys are an effective way to collect quantitative data from a broad audience. This method involves creating a structured questionnaire with targeted questions designed to understand users' preferences, financial tracking habits, and desired features. Surveys can be distributed through online platforms, social media, or email, ensuring a diverse range of responses. Consider incorporating

both multiple-choice and open-ended questions to gather a mix of quantitative and qualitative data.

- Design: Surveys were designed with clear and concise questions, covering a range of topics such as user habits, preferences, pain points, and expectations for a personal expense tracking system.
- Distribution: The surveys were distributed through multiple channels, including email, social media, and online forums, to reach a broad audience and collect a diverse range of responses.
- Analysis: After collecting the survey responses, the data was analyzed to identify trends, common requirements, and key insights. This analysis provided a foundational understanding of the user base and helped to prioritize features.

2) User Interviews: Interviews provide a more in-depth approach to requirement gathering. By conducting one-on-one or group interviews with representative users, the development team gains a deeper understanding of users' needs, challenges, and expectations. Interviews allow for real-time feedback, enabling developers to ask follow-up questions and explore specific topics in detail. This method is ideal for identifying pain points, discussing potential solutions, and building a relationship with end-users.

- Selection: Potential users from different backgrounds were selected to participate in interviews, ensuring a diverse representation of opinions and needs.
- Preparation: Interview questions were prepared to guide the conversation and focus on topics relevant to the development of the expense tracking system, such as specific features, usability concerns, and desired outcomes.
- Execution: Interviews were conducted in person or via video conferencing to gather in-depth insights from participants. This interactive format allowed for follow-up questions and detailed discussions.
- Documentation: Interview responses were documented, highlighting key requirements, user pain points, and suggestions for system improvements. This documentation played a vital role in refining project requirements and ensuring user-centric development.

3) Competitor Analysis: Analyzing existing personal expense tracking systems and similar software solutions can provide insights into common features, industry trends, and user expectations. This method involves reviewing competitor platforms to

identify their strengths, weaknesses, and unique selling points. By understanding what competitors offer, the development team can ensure that the proposed system provides a unique and competitive value proposition.

- Identification of Competitors: A list of competitors in the personal expense tracking space was compiled, including both established platforms and emerging tools. This provided a comprehensive view of the market landscape.
- Feature Comparison: A detailed comparison of key features was conducted, focusing on aspects such as user interface design, ease of use, functionality, customization options, and integration with other systems. This comparison allowed the project team to identify unique selling points and gaps in the market.
- User Reviews: Reviews and ratings from users of competing platforms were analyzed to gauge customer satisfaction, common complaints, and areas for improvement. This provided direct insights into user expectations and experiences with other systems.
- Market Positioning: An assessment of competitors' market positioning was performed to understand their target audience, pricing strategies, and marketing approaches. This information helped to guide the positioning of the new system to ensure a competitive edge.
- Documentation: Findings from the competitor analysis were documented and shared with the project team. This documentation included a summary of key takeaways, recommendations for differentiation, and suggestions for unique features to incorporate into the new system.

4) Use case analysis involves defining specific scenarios in which users interact with the system. This method helps identify the various functionalities and interactions required to support different user roles and goals. Use cases can be documented in a structured format, detailing the steps, actors, and expected outcomes for each scenario. This approach ensures that the system meets user needs and supports a wide range of use cases.

- Use Case Identification: The project team identified a range of use cases that represented common user interactions with the expense tracking system. These use cases covered various scenarios, such as recording expenses, creating budgets, tracking spending, and generating reports.

- Use Case Development: Each use case was detailed with specific steps, actors involved, and expected outcomes. This development process ensured that the use cases were comprehensive and accounted for different user roles and system functionalities.
- Validation and Testing: The use cases were validated by stakeholders and tested through simulations to ensure they accurately represented real-world interactions. This testing helped to identify any gaps or inconsistencies in the system's design.
- Documentation: The use cases were documented and used as a reference throughout the project's development lifecycle. This documentation served as a guide for developers and helped to ensure that the system was built to meet user expectations.

3.2 Technology

The technology stack used for this web-based Comprehensive Personal Expenses Tracking System includes:

Programming Language: Java, Html

Backend Framework: Spring, SpringBoot, Mybatis

Frontend Framework: Vue.js and ElementUI

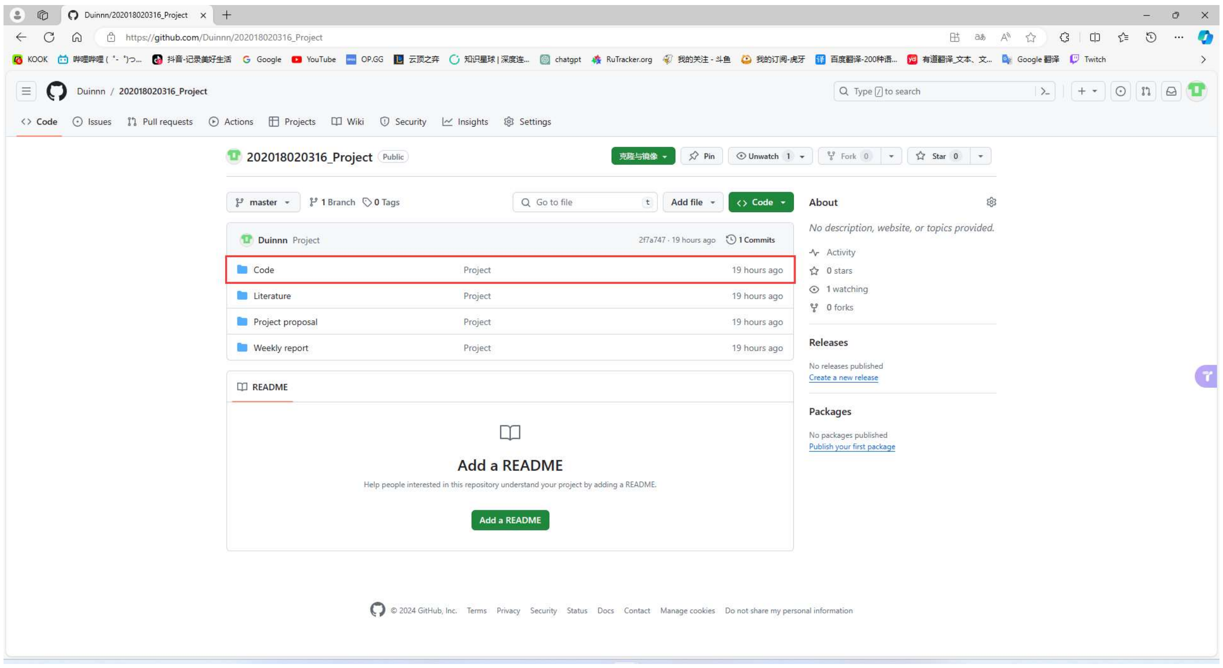
Database: MySQL 8.0

Development Tools: IntelliJ IDEA, Visual Studio Code

Server: Apache Tomcat 9.0

3.3 Project Version Management

First, I open Git Bash in the Code folder. Then, in the Git Bash command, I entered "git status" to check the status of the git repository, indicating that I have untracked files in the "" directory. Then enter "git add" to temporarily store all changes in the working directory, including untracked files. Then enter "git status" again to check the status display. The new file in the directory is ready for submission. Finally, enter "git push" to push the changes to a remote repository on GitHub.



https://github.com/Duinnn/202018020316_Project

Duinnn / 202018020316_Project

202018020316_Project Public

master · 1 Branch · 0 Tags

Duinnn Project — 2f7a747 · 19 hours ago — 1 Commits

Code	Project	19 hours ago
Literature	Project	19 hours ago
Project proposal	Project	19 hours ago
Weekly report	Project	19 hours ago

README

Add a README

Help people interested in this repository understand your project by adding a README.

About

No description, website, or topics provided.

Activity · 0 stars · 1 watching · 0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

© 2024 GitHub, Inc.

Figure 3. Project Version Management with Git

Link: https://github.com/Duinnn/202018020316_Project

Chapter 4 Implementation and Testing

4.1 Implementation

4.1.1 Register

In the login interface, users need to click the “Register” button to register. When registering, the password must consist of 8 letters and numbers, and cannot contain illegal characters.

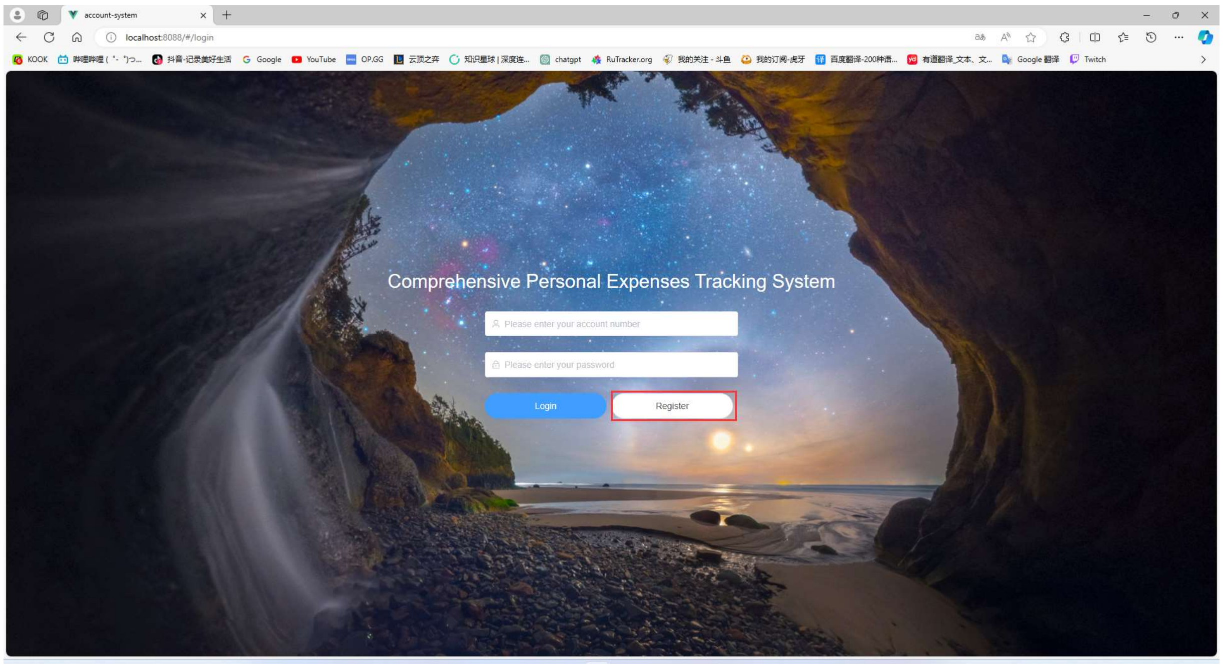

account-system
localhost:8088/#/login
Comprehensive Personal Expenses Tracking System
Please enter your account number
Please enter your password
Login
Register

Figure 4. Login Page (Highlight the “Register” button)

4.1.2 Login

When you register a user with the username user01, you can now enter your username and password and click the “Login” button to log in to the website.

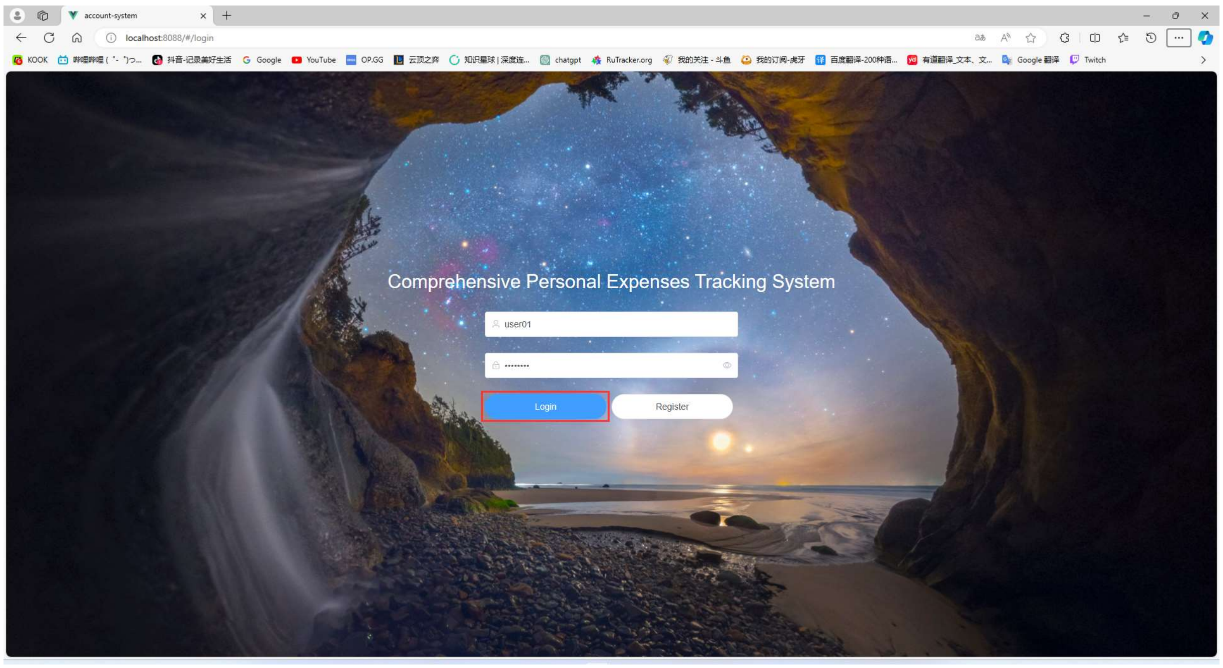


Figure 5. Login page (Highlight the “Login” button)

4.1.3 Add Bill

When logging into the website, you will first be redirected to the Bill Management page, where you can add bills, search for bills, and view bills.

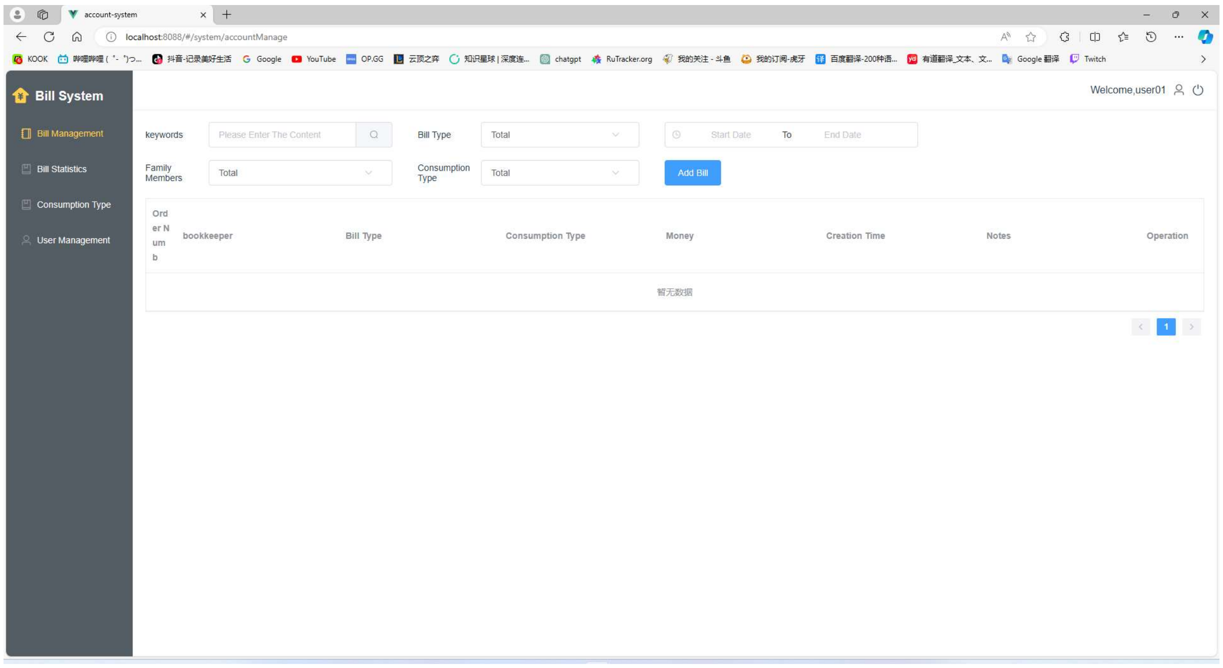


Figure 6. Bill Management page

When you need to add a Bill, you just need to click the “Add Bill” button and you will see the “Add Bill” form (Expense Bill Form and Income Bill Form).

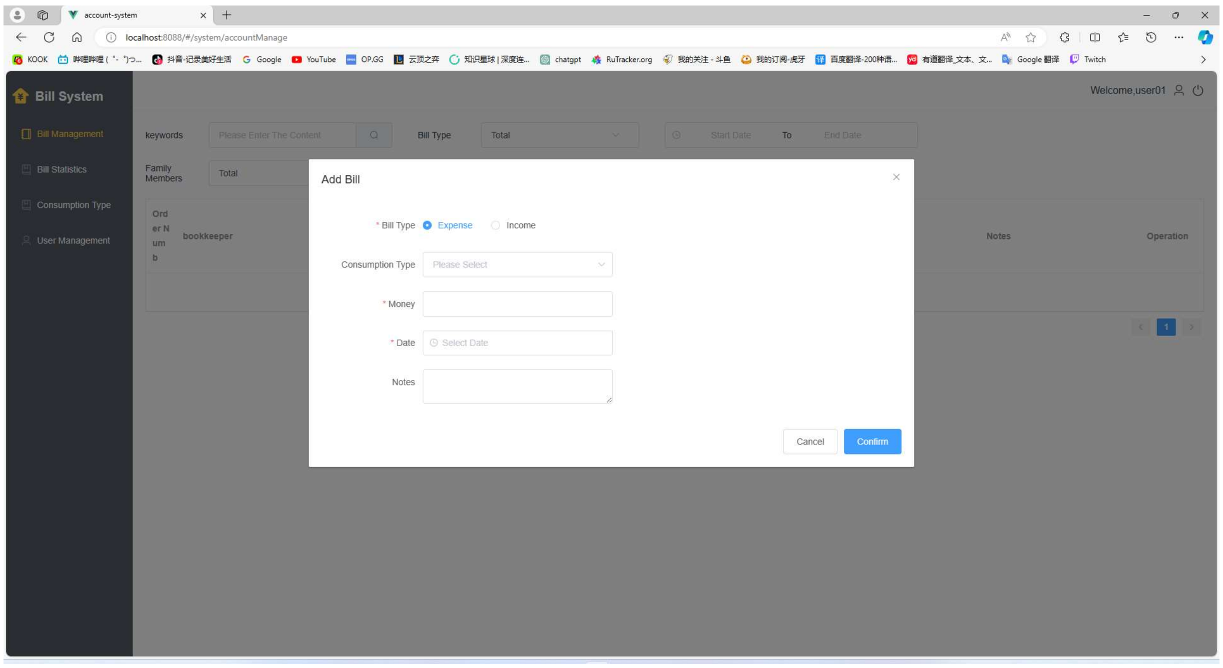


Figure 7. Expense Bill Form

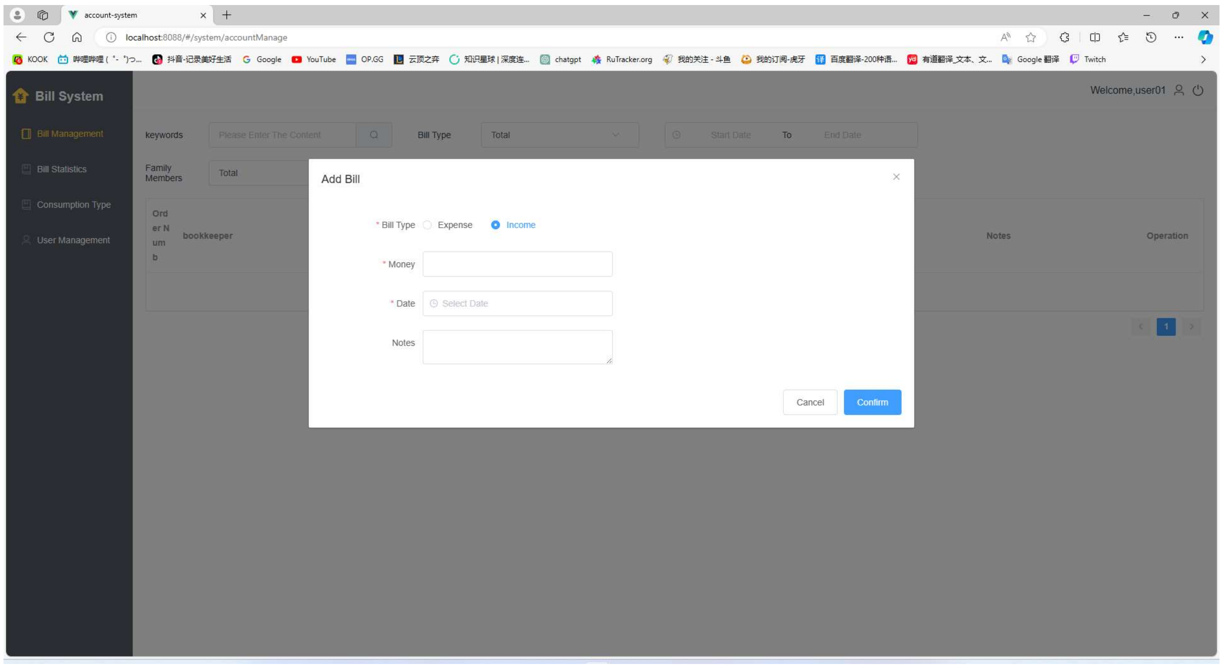
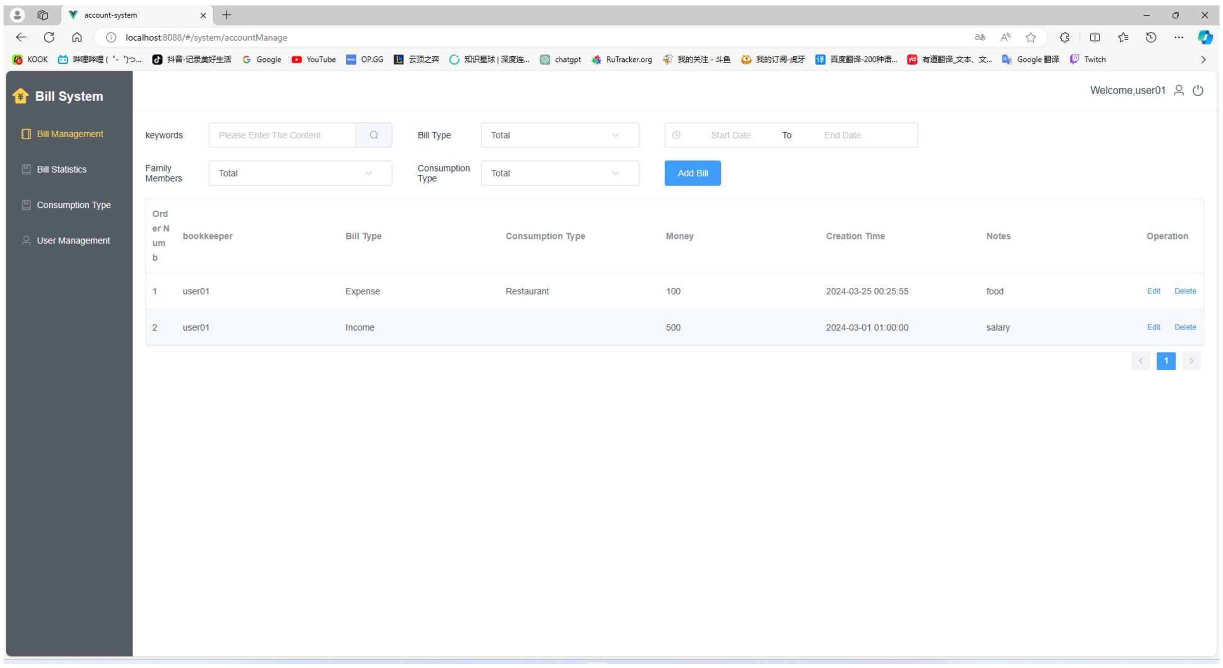


Figure 8. Income Bill Form

After entering the Bill form information, you can see the bill you just added on the Bill Management page.



keywords: Please Enter The Content | Bill Type: Total | Start Date To End Date

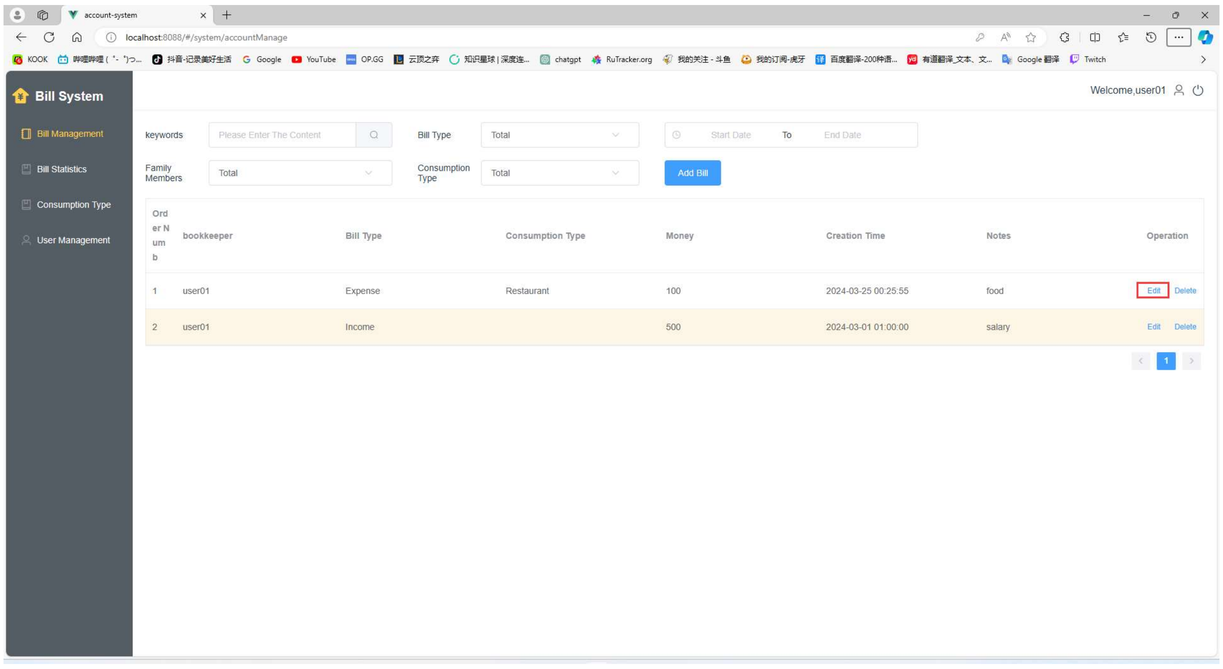
Family Members: Total | Consumption Type: Total | Add Bill

Order Numb	bookkeeper	Bill Type	Consumption Type	Money	Creation Time	Notes	Operation
1	user01	Expense	Restaurant	100	2024-03-25 00:25:55	food	Edit Delete
2	user01	Income		500	2024-03-01 01:00:00	salary	Edit Delete

Figure 9. Bill Management page (After adding bill)

4.1.4 Edit Bill

If the bill information you just added is incorrect, you can click the Edit button to display the “Edit Bill” form information.

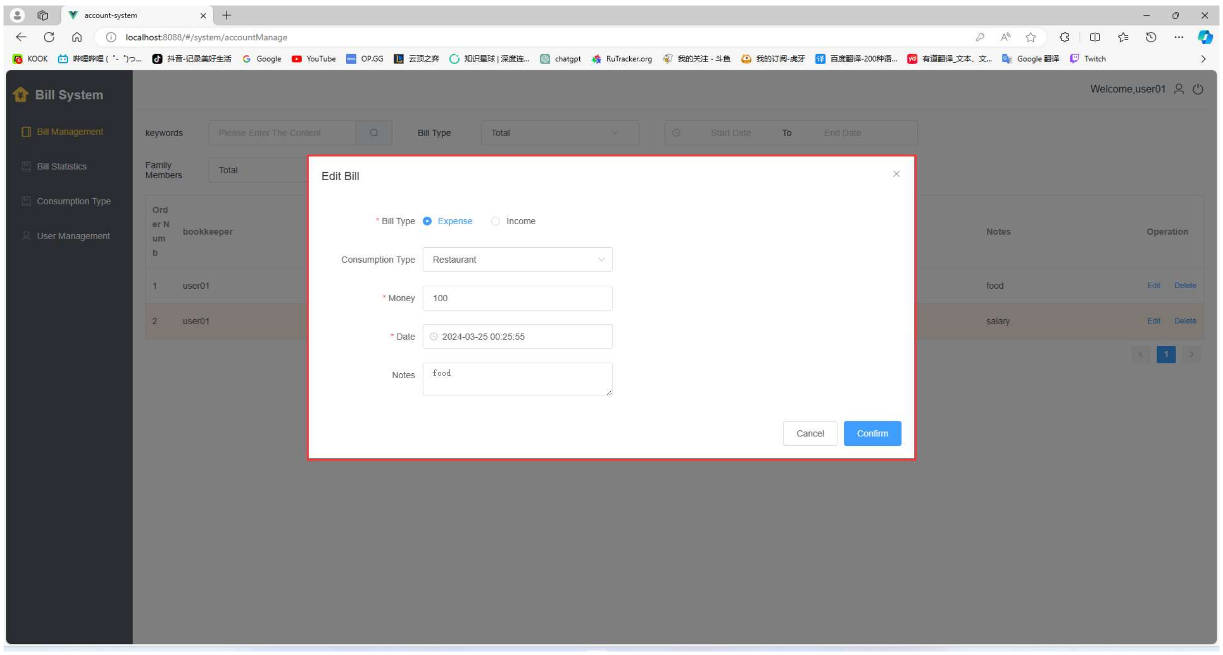


keywords: Please Enter The Content
Bill Type: Total
Start Date To End Date
Family Members: Total
Consumption Type: Total

Order Numb	bookkeeper	Bill Type	Consumption Type	Money	Creation Time	Notes	Operation
1	user01	Expense	Restaurant	100	2024-03-25 00:25:55	food	Edit Delete
2	user01	Income		500	2024-03-01 01:00:00	salary	Edit Delete

Figure 10. Bill Management page (Highlight the “Edit” button)

You can change your bill information in the Edit Bill Form and click the “Confirm” button to successfully change the bill information.



Edit Bill

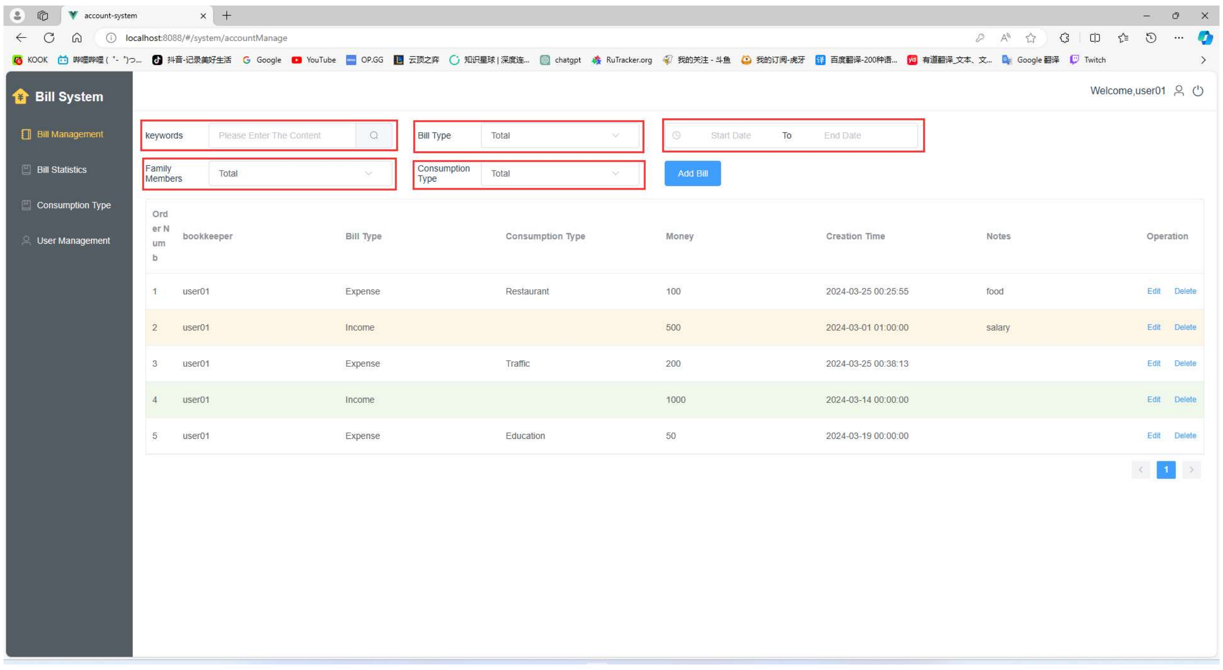
* Bill Type	[✓ Expense] Income
Consumption Type	Restaurant
* Money	100
* Date	2024-03-25 00:25:55
Notes	food

Order Numb	bookkeeper	Notes	Operation
1	user01	food	Edit Delete
2	user01	salary	Edit Delete

Figure 11. Edit Bill Form

4.1.5 Search Bill

When you want to query a specific bill, you can enter the query field in the Keywords input box, select whether you want to query income or expenses in the Bill Type selection box, select which family member's bill you want to query in the Family Members selection box, select the type of bill you want to query in the Consumption Type selection box, and finally select a specific time range in the Time selection box to only view bills within that range.



Order Numb	bookkeeper	Bill Type	Consumption Type	Money	Creation Time	Notes	Operation
1	user01	Expense	Restaurant	100	2024-03-25 00:25:55	food	Edit Delete
2	user01	Income		500	2024-03-01 01:00:00	salary	Edit Delete
3	user01	Expense	Traffic	200	2024-03-25 00:38:13		Edit Delete
4	user01	Income		1000	2024-03-14 00:00:00		Edit Delete
5	user01	Expense	Education	50	2024-03-19 00:00:00		Edit Delete

Figure 12. Bill Management Page (Highlight Keywords, Bill Type, Family Members, Consumption Type and Time)

4.1.6 View Bill Statistics

When you click the "Bill Statistics" button, you will be redirected to the statistics table page, where you can see your billing information presented in line and sector charts.

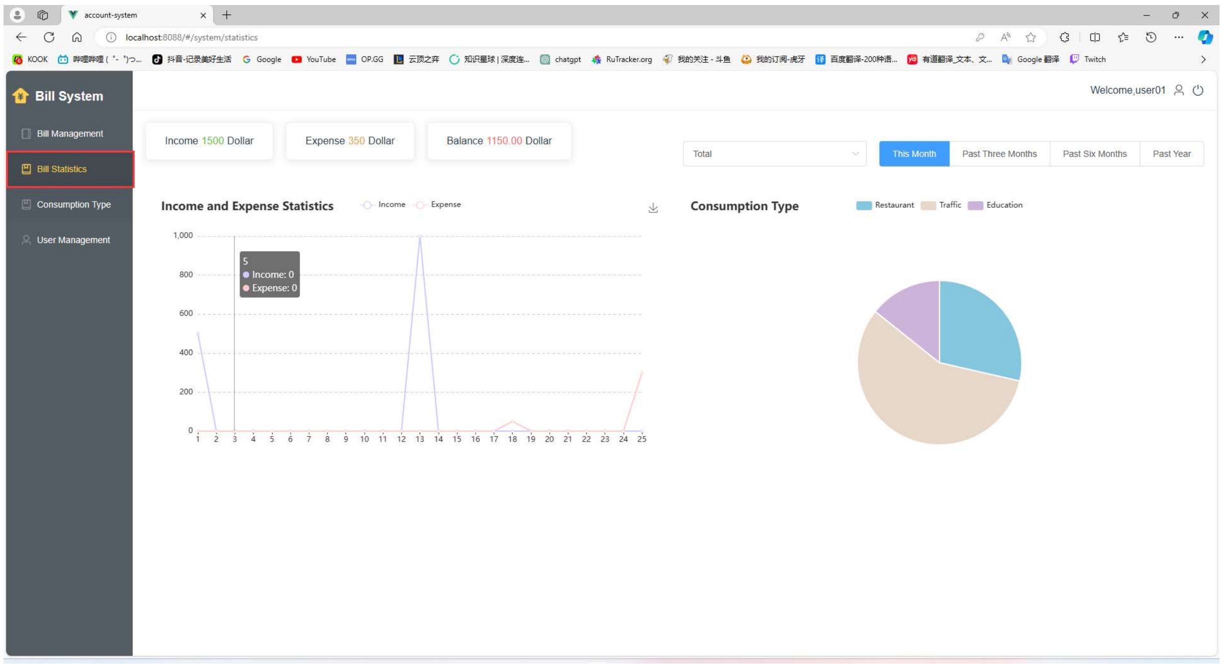


Figure 13. Bill Statistics Page (Highlight "Bill Statistics" button)

In the line chart, you can click the Income button and the Expense button to choose whether to display the income bill separately or the expense bill separately. And you can choose both of them to display.

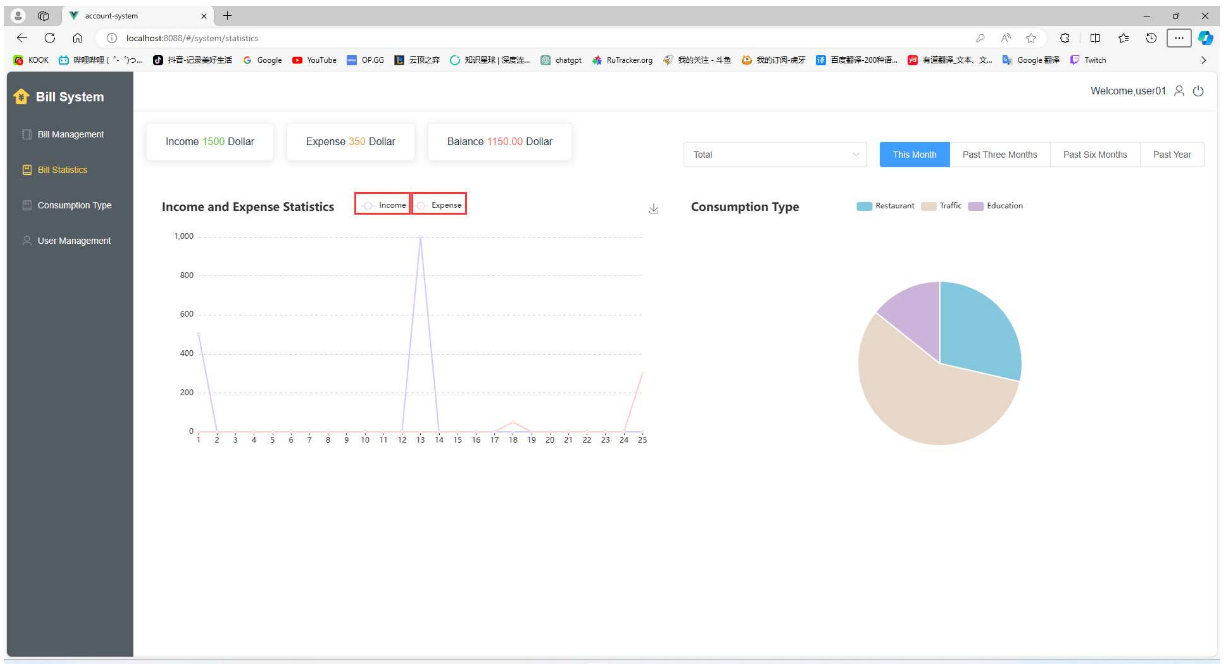


Figure 14. Bill Statistics Page (Highlight "Income" and "Expense" button)

You can click on This month, Pash Three months, Past Six months, and Past Year to select the time range you want to view, which is valid for both line and sector charts.

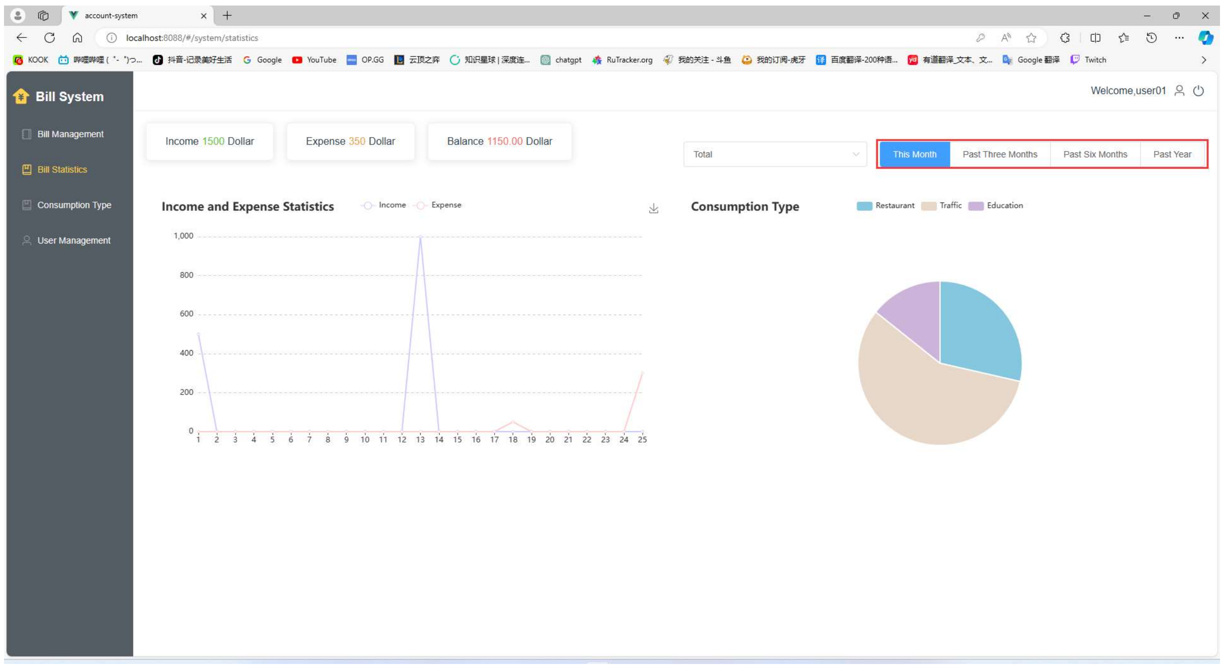
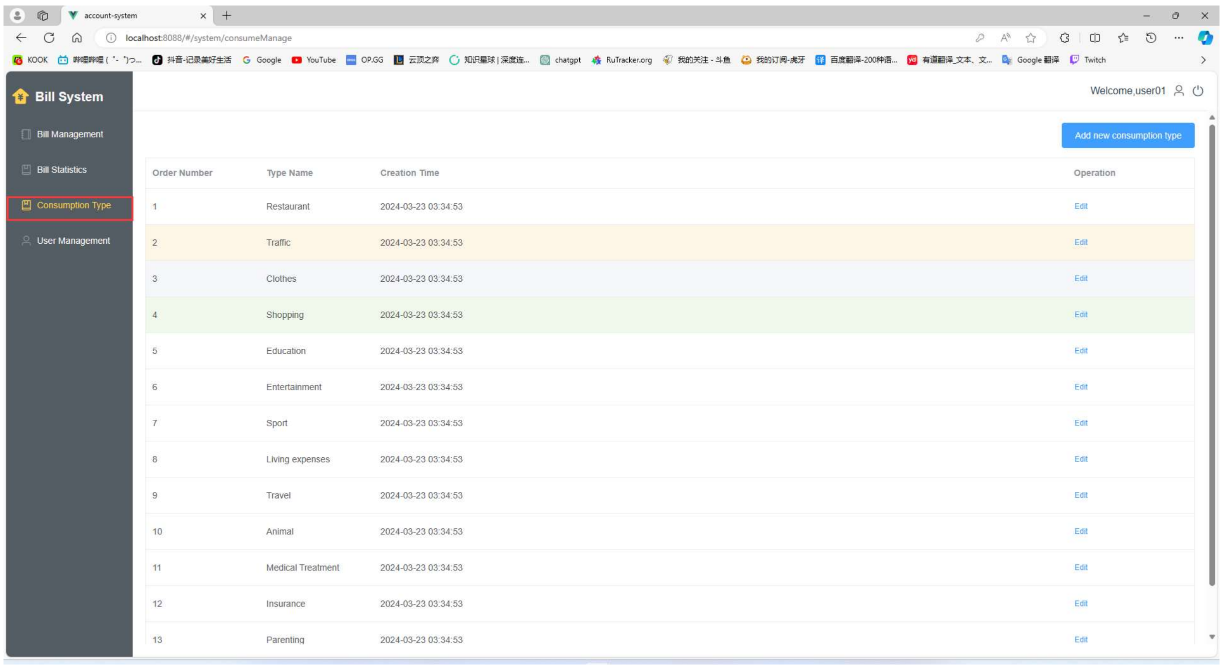


Figure 15. Bill Statistics Page (Highlight time button)

4.1.7 View Consumption Type and Add Consumption Type

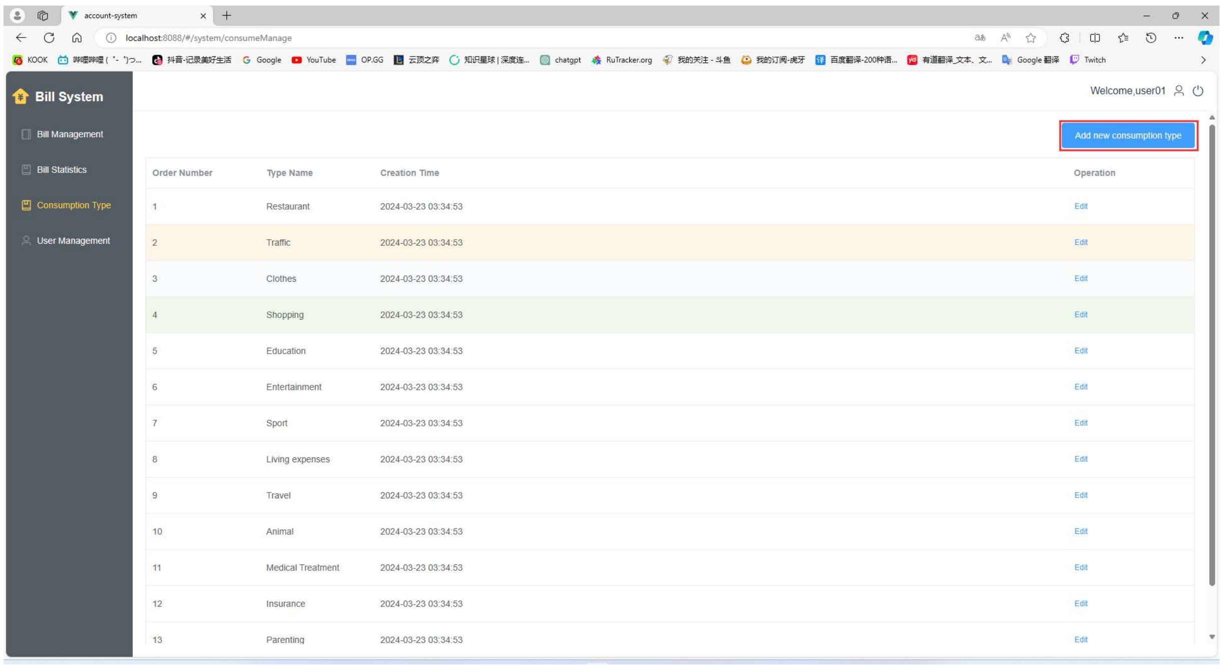
You can click the "Consumption Type" button to jump to the Consumption Type page, where you can see the default consumption type, but the default consumption type cannot be modified.



Order Number	Type Name	Creation Time	Operation
1	Restaurant	2024-03-23 03:34:53	Edit
2	Traffic	2024-03-23 03:34:53	Edit
3	Clothes	2024-03-23 03:34:53	Edit
4	Shopping	2024-03-23 03:34:53	Edit
5	Education	2024-03-23 03:34:53	Edit
6	Entertainment	2024-03-23 03:34:53	Edit
7	Sport	2024-03-23 03:34:53	Edit
8	Living expenses	2024-03-23 03:34:53	Edit
9	Travel	2024-03-23 03:34:53	Edit
10	Animal	2024-03-23 03:34:53	Edit
11	Medical Treatment	2024-03-23 03:34:53	Edit
12	Insurance	2024-03-23 03:34:53	Edit
13	Parenting	2024-03-23 03:34:53	Edit

Figure 16. Consumption Type Page (Highlight “Consumption Type” button)

You can click the "Add new consumption type" button to jump to the "Add new consumption type" Form.


localhost:8088/#/system/consumeManage

Bill System

Welcome,user01

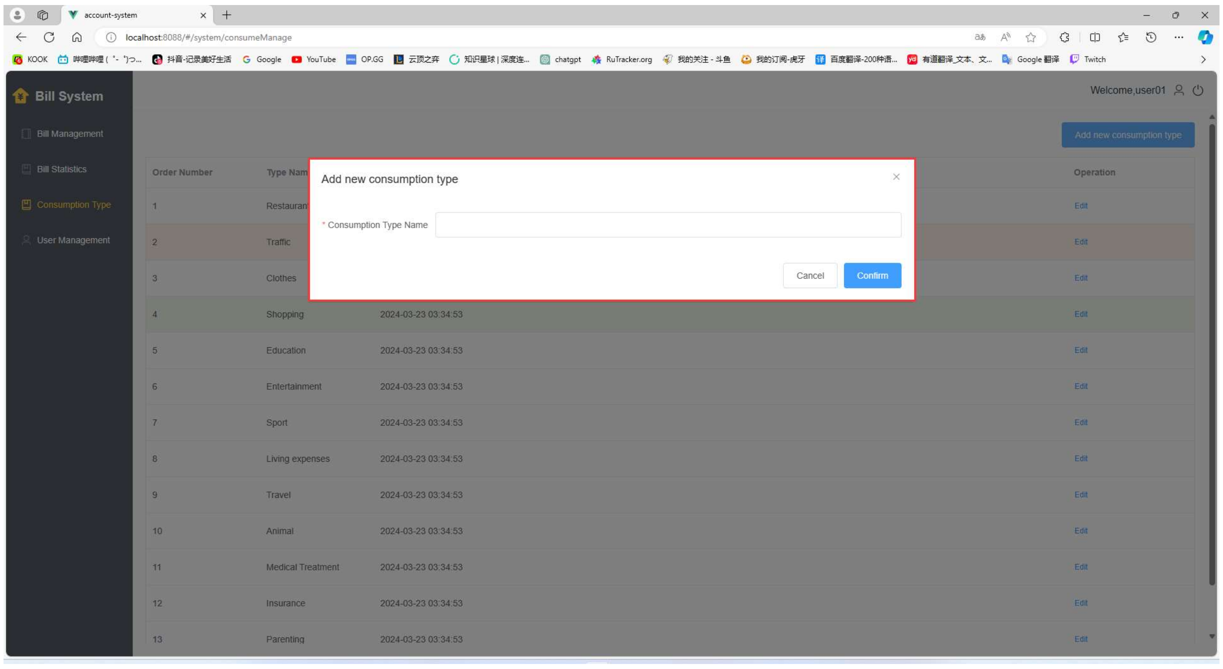
- Bill Management
- Bill Statistics
- Consumption Type
- User Management

Add new consumption type

Order Number	Type Name	Creation Time	Operation
1	Restaurant	2024-03-23 03:34:53	Edit
2	Traffic	2024-03-23 03:34:53	Edit
3	Clothes	2024-03-23 03:34:53	Edit
4	Shopping	2024-03-23 03:34:53	Edit
5	Education	2024-03-23 03:34:53	Edit
6	Entertainment	2024-03-23 03:34:53	Edit
7	Sport	2024-03-23 03:34:53	Edit
8	Living expenses	2024-03-23 03:34:53	Edit
9	Travel	2024-03-23 03:34:53	Edit
10	Animal	2024-03-23 03:34:53	Edit
11	Medical Treatment	2024-03-23 03:34:53	Edit
12	Insurance	2024-03-23 03:34:53	Edit
13	Parenting	2024-03-23 03:34:53	Edit

Figure 17. Consumption Type Page (Highlight "Add new consumption type" button)

In the Add new consultation type form, you can enter the name of the consultation type you want to add, and then click the Confirm button to successfully add it.


localhost:8088/#/system/consumeManage

Add new consumption type

* Consumption Type Name

Cancel Confirm

Figure 18. Add new consumption type Form

4.1.8 View User Management and Add User

You can click the “User Management” button to jump to the User Management Page.

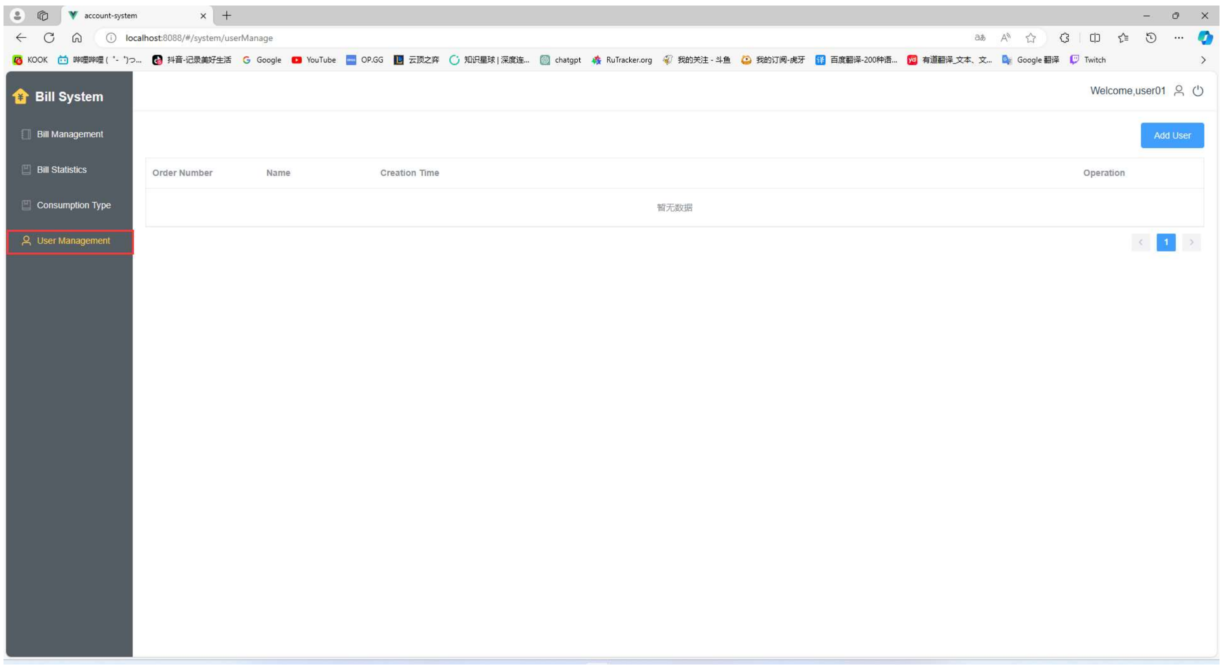


Figure 19. User Management Page (Highlight "User Management" button)

On the User Management Page, you can click the "Add User" button to display the Add family member form.

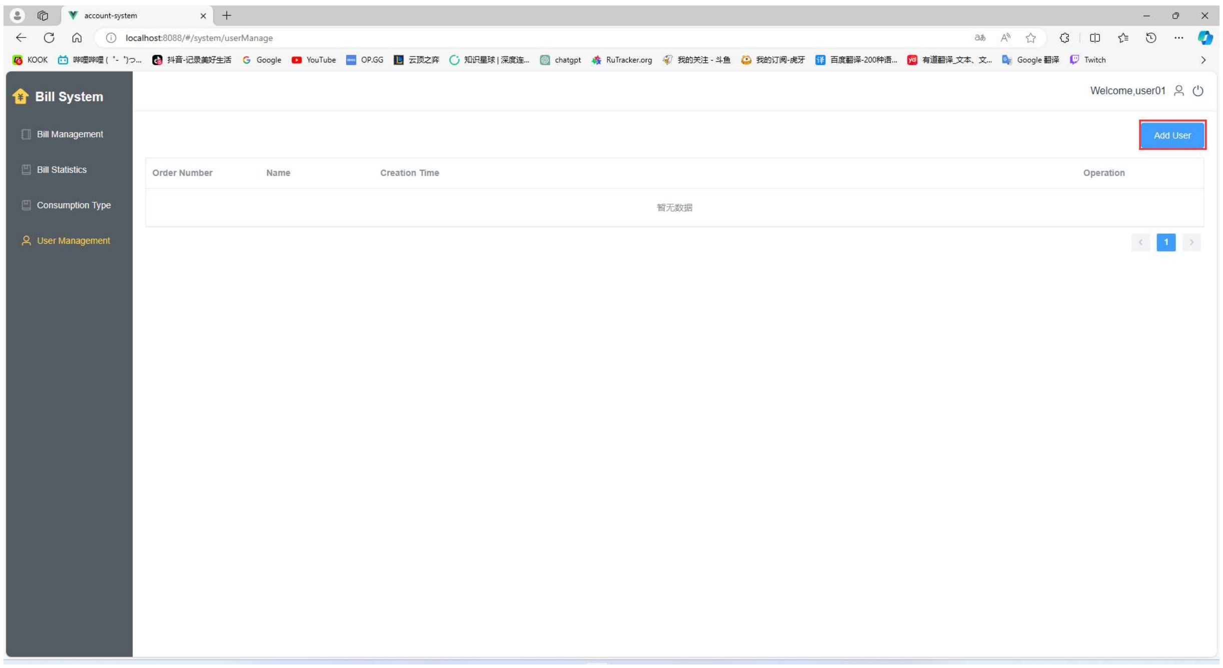


Figure 20. User Management Page (Highlight "Add User" button)

In the Add Family Member Form, you can enter the Account Number and Password of the family member you want to add, and then click the “Confirm” button to successfully add the family member.

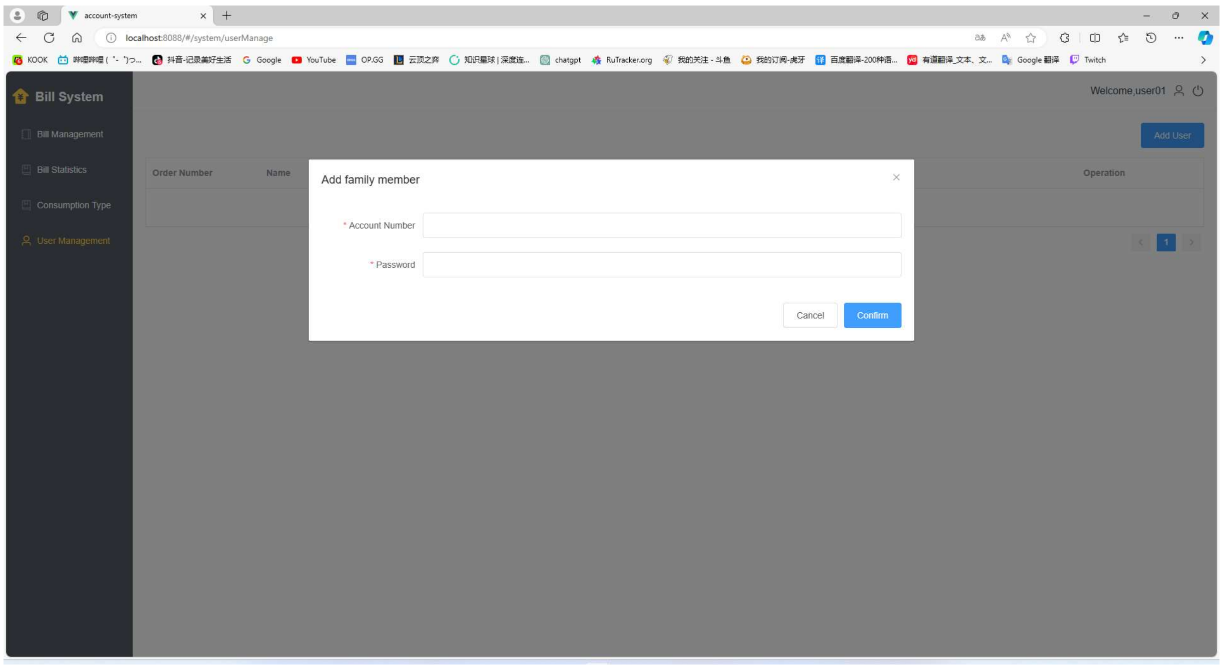


Figure 21. Add Family Member Form

4.1.9 Change the Current User Password

When you move the mouse over the portrait icon, a “Change Password” button will appear. Then, you can click the” Change Password” button to jump to the “Change Password” Form.

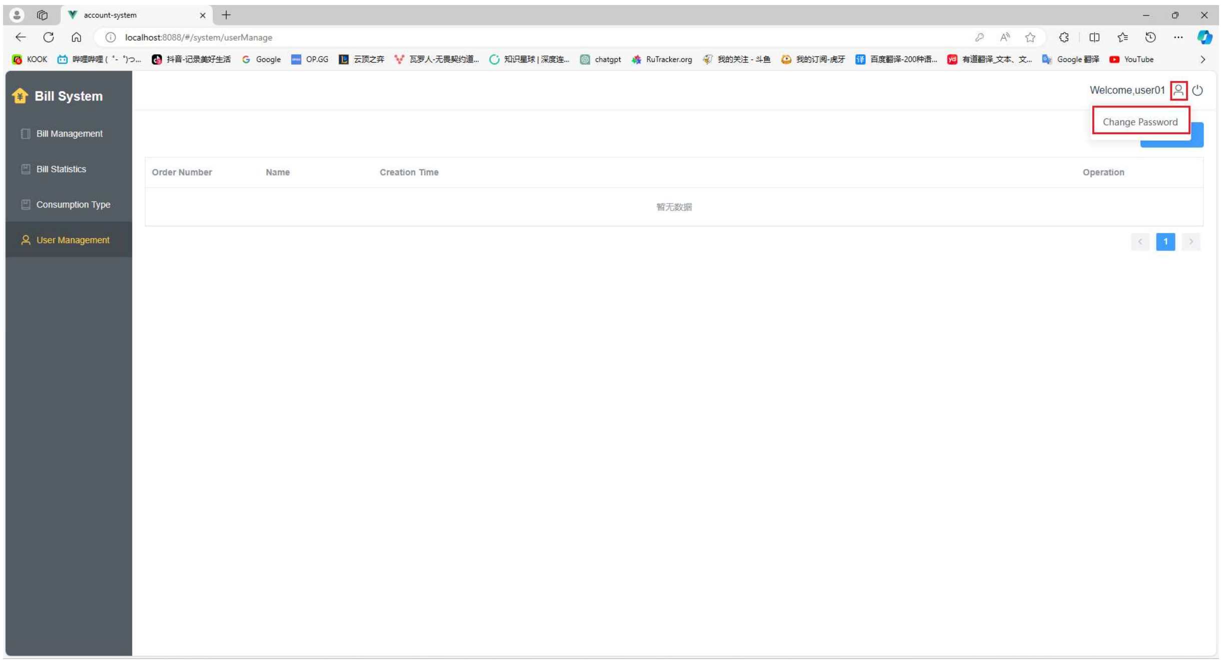


Figure 22. Change password (Highlight "Change Password" button)

In the Change Password Form, you can enter the Origin Password, New Password, Confirm Password to change your password, and then click the “Confirm” button to successfully change the password.

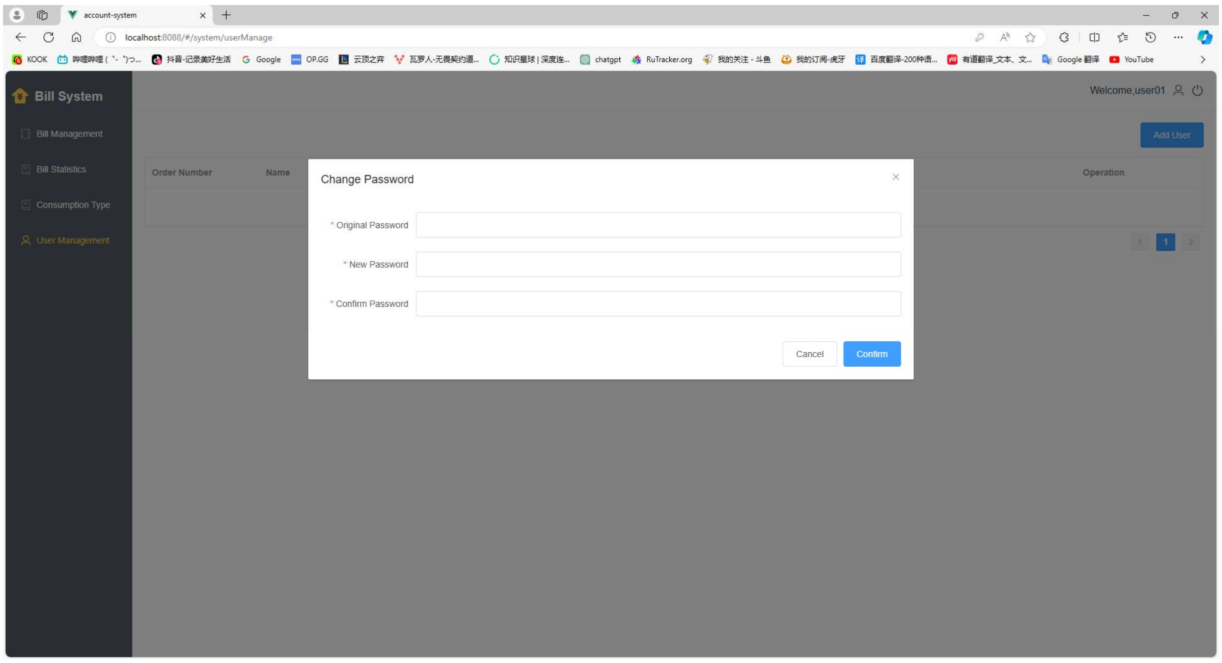


Figure 23. Change Password Form

4.1.10 Exit Account

When you move the mouse over the “Exit” button, an Exit text prompt will appear. If you click the button, you can successfully exit the current account.

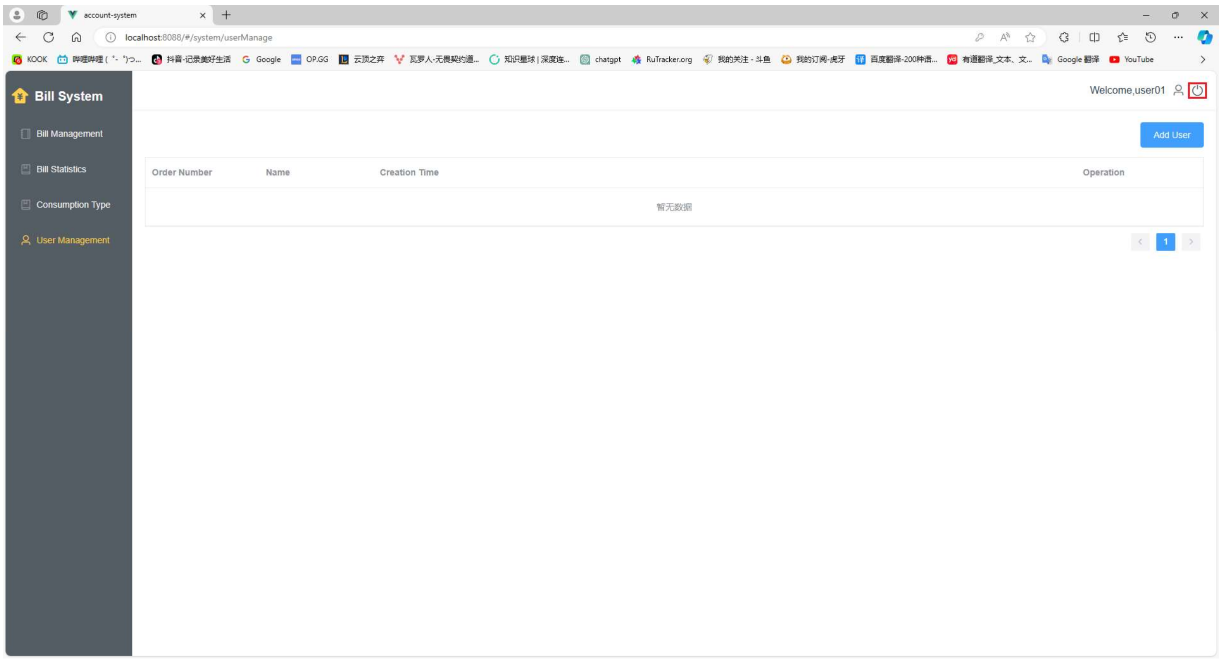


Figure 24. User Management Page (Highlight "Exit" button)

After clicking the “Exit” button, you will return to the Login interface and successfully log out of your account.

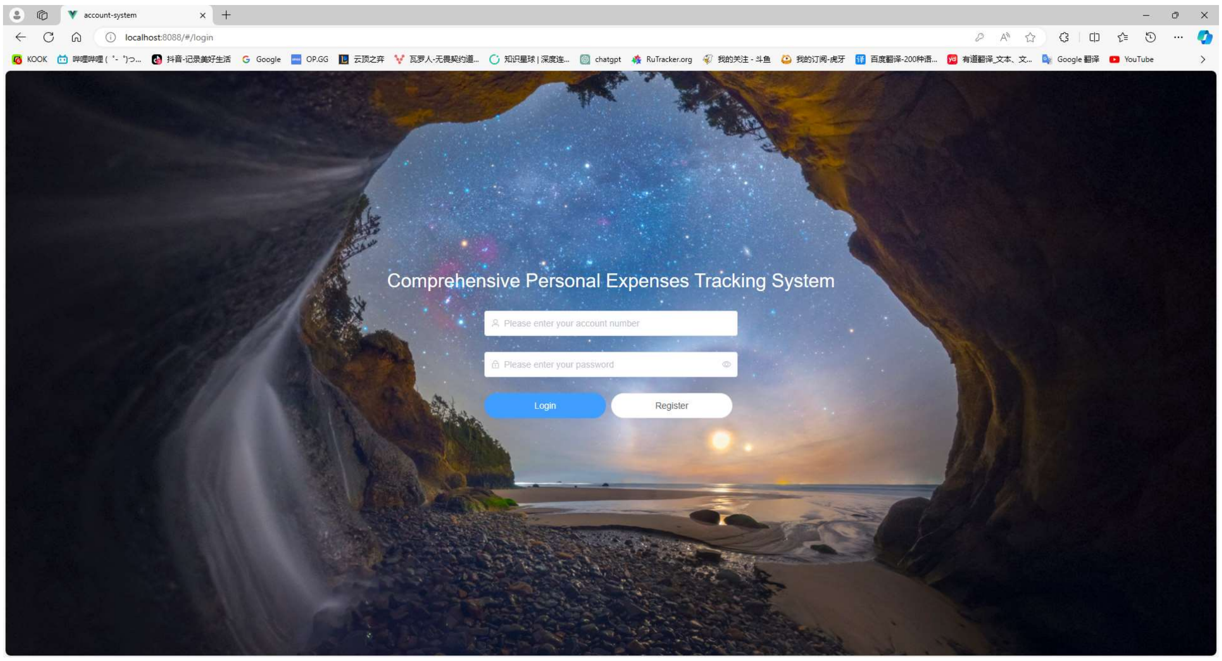

account-system
localhost:8088/#/login
Comprehensive Personal Expenses Tracking System
Please enter your account number
Please enter your password
Login
Register

Figure 25. Login Page (After you click the “Exit” button)

4.2 Testing

4.2.1 Functional Testing

JUnit is a widely used framework for functional testing in Java applications. Its primary purpose is to facilitate the creation and execution of automated test cases to verify that individual units of code, such as methods or classes, function as expected. By allowing developers to write comprehensive test suites, JUnit helps ensure that the software behaves correctly under various conditions and complies with its functional requirements.

The benefits of using JUnit for functional testing are substantial. It promotes a test-driven development (TDD) approach, where tests are written before the code, fostering better design and higher code quality. JUnit's annotation-based structure simplifies the process of setting up, executing, and tearing down test cases, making it easier for developers to focus on testing logic (García and Dueñas, 2011). Its integration with build tools like Maven and Gradle ensures seamless execution within continuous integration (CI) pipelines, enabling early detection of bugs and reducing regression risks (Cleary and Inglis, 2007). Additionally, JUnit's comprehensive reporting features help teams quickly identify and address test failures. By incorporating JUnit into the development process, teams can improve code reliability, accelerate development cycles, and maintain robust test coverage, ultimately leading to higher-quality software.

Scope of Testing	Test Purpose	Test Type
service	The purpose of testing services is to verify the business logic implemented within them behaves as expected. By writing tests for service methods, you can ensure that they perform the intended operations and produce the correct results according to the specified requirements. This helps maintain the correctness and reliability of the application's core functionality.	(Functional Testing) Unit Test
mapper	The purpose of testing mappers is to validate their ability to correctly map data between application objects and database tables. By writing integration tests for mappers, you can ensure that they accurately translate object-oriented data structures to relational database representations and vice versa. This helps maintain data integrity and	(Functional Testing) Unit Test

	consistency within the application.	
utils	The purpose of testing utility methods is to verify their correctness and reliability, especially if they contain critical or complex logic. By writing unit tests for utility methods, you can ensure that they perform as expected and produce the desired outcomes, thereby enhancing the overall robustness of the application.	(Functional Testing) Unit Test

Table 3. Types of classes that require testing

Test Class: BillMapperTest

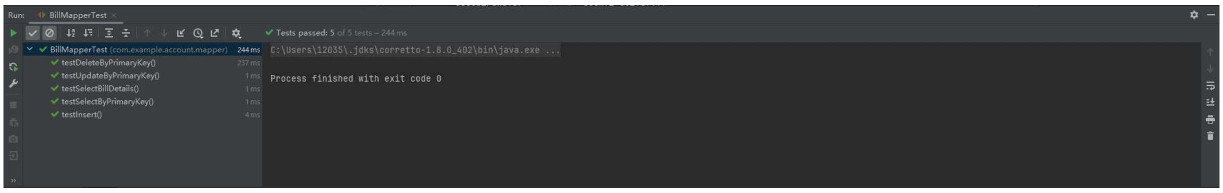
Testing Scope: BillMapper

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. ‘testSelectByPrimaryKey’: Retrieve Bill by primary key 2. ‘testInsert’: Insert a Bill 3. ‘testSelectBillDetails’: Retrieve Bill details 4. ‘testUpdateByPrimaryKey’: Update a Bill by primary key 5. ‘testDeleteByPrimaryKey’: Delete a Bill by primary key	1. Return the expected Bill 2. Successful insertion (1 returned) 3. Retrieve list of Bill details 4. Successful update (1 returned)	1. Expected Bill was returned 2. Insertion returned 1 3. List of Bill details returned 4. Update returned 1

	5. Successful deletion (1 returned)	5. Deletion returned 1

Table 4. BillMapperTest Result



```
Run:  BillMapperTest
Tests passed: 5 of 5 tests – 244 ms
BillMapperTest (com.example.account.mapper)  244 ms
    testDeleteByPrimaryKey()                 237 ms
    testUpdateByPrimaryKey()                   1 ms
    testSelectBillDetails()                    1 ms
    testSelectByPrimaryKey()                   1 ms
    testInsert()                               4 ms

C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...

Process finished with exit code 0
```

Figure 26. BillMapperTest Result

Test Class: ConsumptionTypeMapperTest

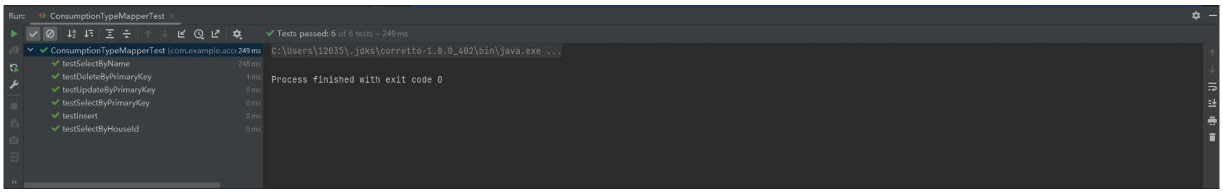
Testing Scope: ConsumptionTypeMapper

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. ‘testInsert’: Insert a new Consumption Type 2. ‘testSelectByPrimaryKey’: Retrieve by primary key 3. ‘testSelectByHouseId’: Retrieve Consumption Types by house ID 4. ‘testUpdateByPrimaryKey’: Update a Consumption Type 5. ‘testDeleteByPrimaryKey’: Delete by primary key 6. ‘testSelectByName’: Retrieve Consumption Type by name	1. Successful insertion (1 returned) 2. Return expected Consumption Type 3. List of expected Consumption Types by house ID 4. Successful update (1 returned) 5. Successful deletion (1 returned)	1. Insertion returned 1 (success) 2. Expected Consumption Type returned 3. Expected Consumption Types list returned 4. Update returned 1 (success) 5. Deletion returned 1 (success) 6. Expected Consumption Type returned by name

	6. Return expected Consumption Type by name	

Table 5. ConsumptionTypeMapperTest Result



```
Run:  ConsumptionTypeMapperTest
✓ Tests passed: 6 of 6 tests – 249 ms
✓ ConsumptionTypeMapperTest (com.example.acc  249 ms
    ✓ testSelectByName          248 ms
    ✓ testDeleteByPrimaryKey      1 ms
    ✓ testUpdateByPrimaryKey      0 ms
    ✓ testSelectByPrimaryKey      0 ms
    ✓ testInsert                  0 ms
    ✓ testSelectByHouseId         0 ms

C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...

Process finished with exit code 0
```

Figure 27. ConsumptionTypeMapperTest Result

Test Class: HouseMapperTest

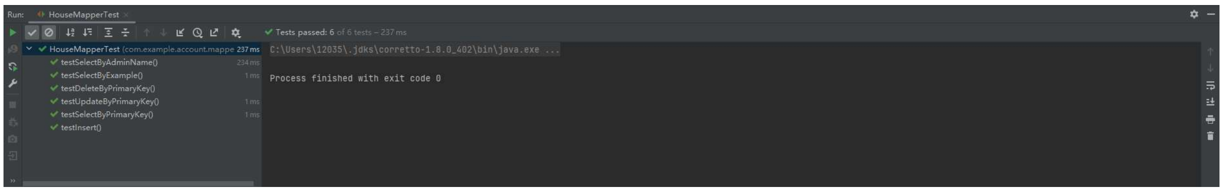
Testing Scope: HouseMapper

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. ‘testSelectByPrimaryKey’: Retrieve House by primary key 2. ‘testInsert’: Insert a new House 3. ‘testSelectByExample’: Retrieve Houses by example 4. ‘testUpdateByPrimaryKey’: Update House by primary key 5. ‘testDeleteByPrimaryKey’: Delete House by primary key 6. ‘testSelectByAdminName’: Retrieve House by admin name	1. Successful retrieval of House (admin name is 'John') 2. Successful insertion (1 returned) 3. Two Houses retrieved by example 4. Successful update (1 returned) 5. Successful deletion (1 returned) 6. Successful retrieval of House by admin name	1. House retrieved with admin name 'John' 2. Insertion returned 1 (success) 3. Two Houses returned 4. Update returned 1 (success) 5. Deletion returned 1 (success) 6. House retrieved with

		admin name 'John'

Table 6. HouseMapperTest Result



```
Run:  HouseMapperTest
Tests passed: 6 of 6 tests – 237 ms
HouseMapperTest (com.example.account.mappe 237 ms
  testSelectByAdminName()     234 ms
  testSelectByExample()         1 ms
  testDeleteByPrimaryKey()
  testUpdateByPrimaryKey()      1 ms
  testSelectByPrimaryKey()      1 ms
  testInsert()

C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...

Process finished with exit code 0
```

Figure 28. HouseMapperTest Result

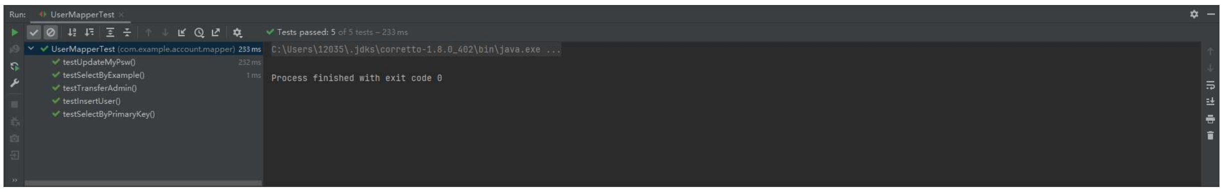
Test Class: UserMapperTest

Testing Scope: UserMapper

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. 'testInsertUser': Insert a new User 2. 'testSelectByExample': Retrieve Users by example 3. 'testSelectByPrimaryKey': Retrieve User by primary key 4. 'testUpdateMyPsw': Update User's password 5. 'testTransferAdmin': Transfer admin rights	1. Successful insertion (1 returned) 2. Two Users retrieved 3. Successful retrieval by primary key 4. Successful password update (1 returned) 5. Successful admin transfer (1 returned)	1. Insertion returned 1 (success) 2. Two Users returned 3. User retrieved successfully 4. Password update returned 1 (success) 5. Admin transfer returned 1 (success)

Table 7. UserMapperTest Result



```
Tests passed: 5 of 5 tests – 233 ms
UserMapperTest (com.example.account.mapper) 233 ms
    testUpdateMyPsw()        232 ms
    testSelectByExample()      1 ms
    testTransferAdmin()
    testInsertUser()
    testSelectByPrimaryKey()

C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...

Process finished with exit code 0
```

Figure 29. UserMapperTest Result

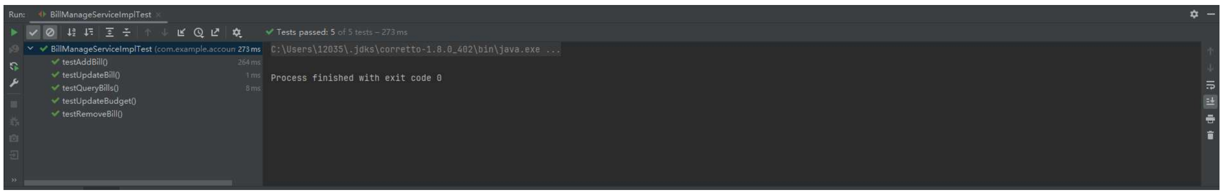
Test Class: BillManageServiceImplTest

Testing Scope: BillManageServiceImpl

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. 'testAddBill': Add a new Bill 2. 'testQueryBills': Query Bills with given parameters 3. 'testUpdateBill': Update an existing Bill 4. 'testRemoveBill': Remove a Bill by primary key 5. 'testUpdateBudget': Update the budget for a House	1. Result status is "200" with description "success" 2. Result status is "200", data is a 'PageResult' with total 1 and non-empty list 3. Result status is "200", description "success" 4. Result status is "200", description "success" 5. Result status is "200", description "success"	1. Add Bill returned expected result with status "200" 2. Query Bills returned expected result with status "200", 'PageResult' with total 1 3. Update Bill returned expected result with status "200" 4. Remove Bill returned expected result with status "200" 5. Update Budget returned expected result with status "200"

Table 8. BillManageServiceImplTest Result



```
Run: BillManageServiceImplTest
Tests passed: 5 of 5 tests – 273 ms
BillManageServiceImplTest (com.example.accoun 273 ms
  testAddBill()        264 ms
  testUpdateBill()       1 ms
  testQueryBills()       8 ms
  testUpdateBudget()
  testRemoveBill()

C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...

Process finished with exit code 0
```

Figure 30. BillManageServiceImplTest Result

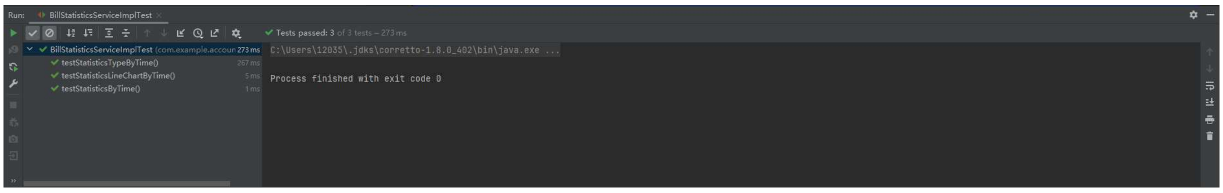
Test Class: BillStatisticsServiceImplTest

Testing Scope: BillStatisticsServiceImpl

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. 'testStatisticsByTime': Get budget and expenditure statistics for a given time type 2. 'testStatisticsTypeByTime': Get expenditure statistics by bill type 3. 'testStatisticsLineChartByTime': Get income and expenditure statistics over time	1. Result status "200", data with 'TotalBudget' and 'TotalExpend' 2. Result status "200", data containing list of bill types with totals 3. Result status "200", data with non-empty lists for dates, income, and expenditure	1. Status "200", with expected data containing budget and expenditure 2. Status "200", with expected list of bill types and totals 3. Status "200", with expected lists for dates, income, and expenditure

Table 9. BillStatisticsServiceImplTest Result



```
Run: BillStatisticsServiceImplTest
Tests passed: 3 of 3 tests – 273 ms
BillStatisticsServiceImplTest (com.example.accoun 273 ms
  testStatisticsTypeByTime()        267 ms
  testStatisticsLineChartByTime()     5 ms
  testStatisticsByTime()              1 ms

C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...

Process finished with exit code 0
```

Figure 31. BillStatisticsServiceImplTest Result

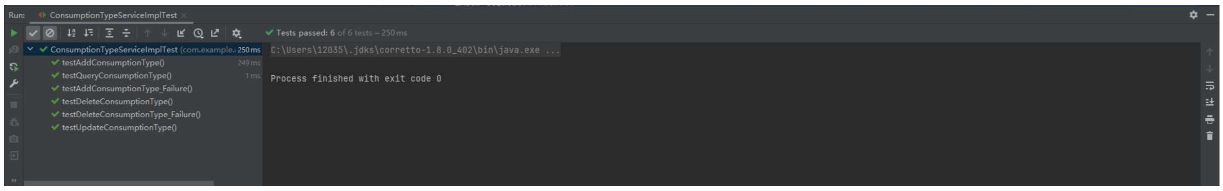
Test Class: ConsumptionTypeServiceImplTest

Testing Scope: ConsumptionTypeServiceImpl

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. 'testQueryConsumptionType': Query Consumption Types by house ID 2. 'testAddConsumptionType': Add a new Consumption Type 3. 'testAddConsumptionType_Failure': Attempt to add a duplicate Consumption Type 4. 'testUpdateConsumptionType': Update an existing Consumption Type 5. 'testDeleteConsumptionType': Delete a custom Consumption Type 6. 'testDeleteConsumptionType_Failure': Attempt to delete a non-custom Consumption Type	1. Result with SUCCESS status and non-null data 2. 'SUCCESS' status, with verification of 'insert' call 3. 'ADD_CONSUMTYPE_FAIL' status due to duplicate name 4. 'SUCCESS' status, with verification of 'update' call 5. 'SUCCESS' status with successful deletion 6. 'DELETE_CONSUMTYPE_FAIL' status due to non-custom type	1. Query Consumption Type returned expected result with status 'SUCCESS' 2. Add Consumption Type returned expected result with status 'SUCCESS' 3. Duplicate Consumption Type returned expected result with status 'ADD_CONSUMTYPE_FAIL' 4. Update Consumption Type returned expected result with status 'SUCCESS' 5. Delete Consumption Type returned expected result with status 'SUCCESS' 6. Non-custom type returned expected result with status 'DELETE_CONSUMTYPE_FAIL'

Table 10. ConsumptionTypeServiceImplTest Result



```
✓ Tests passed: 6 of 6 tests – 250 ms
✓ ConsumptionTypeServiceImplTest (com.example.  250 ms
    ✓ testAddConsumptionType()                  249 ms
    ✓ testQueryConsumptionType()                  1 ms
    ✓ testAddConsumptionType_Failure()
    ✓ testDeleteConsumptionType()
    ✓ testDeleteConsumptionType_Failure()
    ✓ testUpdateConsumptionType()

C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...

Process finished with exit code 0
```

Figure 32. ConsumptionTypeServiceImplTest Result

Test Class: UserServiceImplTest

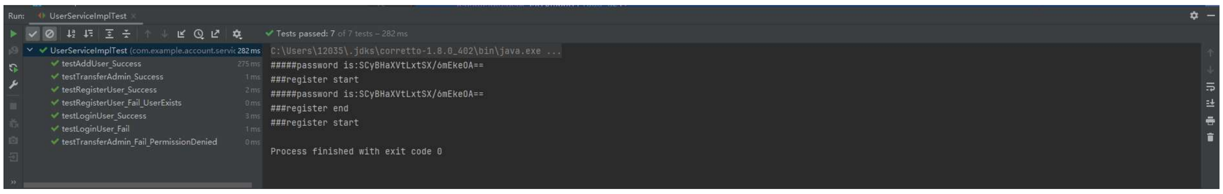
Testing Scope: UserServiceImpl

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. ‘testRegisterUser_Success’: Register a new user successfully 2. ‘testRegisterUser_Fail_UserExists’: Fail registration due to existing user 3. ‘testAddUser_Success’: Add a new user successfully 4. ‘testLoginUser_Success’: Successfully log in a user 5. ‘testLoginUser_Fail’: Fail login due to incorrect password 6. ‘testTransferAdmin_Success’: Successfully transfer admin rights 7. ‘testTransferAdmin_Fail_PermissionDenied’: Fail admin transfer due to permission issues	1. ‘SUCCESS’ code and user is registered 2. ‘REGISTER_FAIL’ code due to existing user 3. ‘SUCCESS’ code and user added 4. ‘SUCCESS’ code for successful login 5. ‘LOGIN_FAIL’ code for incorrect password 6. ‘SUCCESS’ code for admin transfer 7. ‘PERMISSION_DENIED’ code due to lack of permissions	1. User registration returned SUCCESS code 2. Registration failed due to existing user, returned ‘REGISTER_FAIL code’ 3. User addition returned ‘SUCCESS’ code 4. User login successful with ‘SUCCESS’ code 5. User login failed with ‘LOGIN_FAIL’ code 6. Admin transfer successful,

		returned 'SUCCESS' code 7. Admin transfer failed with 'PERMISSION_DENIED' code

Table 11. UserServiceImplTest Result



```
Run:  UserServiceImplTest
✓ Tests passed: 7 of 7 tests – 282 ms
✓ UserServiceImplTest (com.example.account.servic 282 ms
  ✓ testAddUser_Success                    275 ms
  ✓ testTransferAdmin_Success                1 ms
  ✓ testRegisterUser_Success                 2 ms
  ✓ testRegisterUser_Fail_UserExists         0 ms
  ✓ testLoginUser_Success                    3 ms
  ✓ testLoginUser_Fail                       1 ms
  ✓ testTransferAdmin_Fail_PermissionDenied  0 ms

C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...
#####password is:SCyBHaXVtLxtSX/6mEkeOA==
###register start
#####password is:SCyBHaXVtLxtSX/6mEkeOA==
###register end
###register start

Process finished with exit code 0
```

Figure 33. UserServiceImplTest Result

Test Class: DateUtilTest

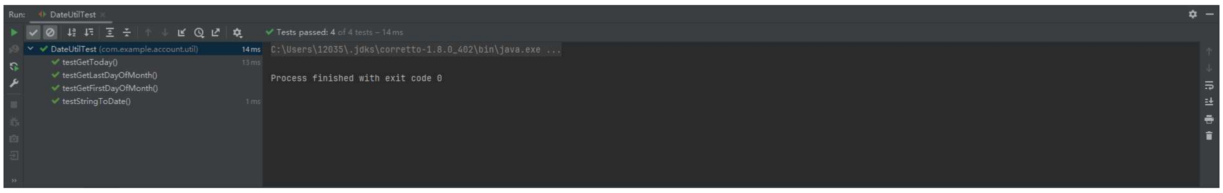
Testing Scope: DateUtil

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. 'testGetToday': Get the current date at midnight 2. 'testGetFirstDayOfMonth': Get the first day of the month 3. 'testGetLastDayOfMonth': Get the last day of the month 4. 'testStringToDate': Convert a string to a date and handle invalid parsing	1. Returned date should be at midnight, with no hour/minute/second/millisecond 2. Returned date should be the first day of the month 3. Returned date should be the last day of the month 4. Valid date should parse correctly, invalid date should throw 'ParseException'	1. The returned date was at midnight with expected values 2. Returned the correct first day of the month 3. Returned the correct last day of the month 4. Correctly parsed a valid date and threw

		‘ParseException’ for invalid date

Table 12. DateUtilTest Result



```
Run:  DateUtilTest
Tests passed: 4 of 4 tests – 14 ms
DateUtilTest (com.example.account.util)   14 ms
  testGetToday()                          13 ms
  testGetLastDayOfMonth()
  testGetFirstDayOfMonth()
  testStringToDate()                       1 ms

C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...

Process finished with exit code 0
```

Figure 34. DateUtilTest Result

Test Class: MD5UtilsTest

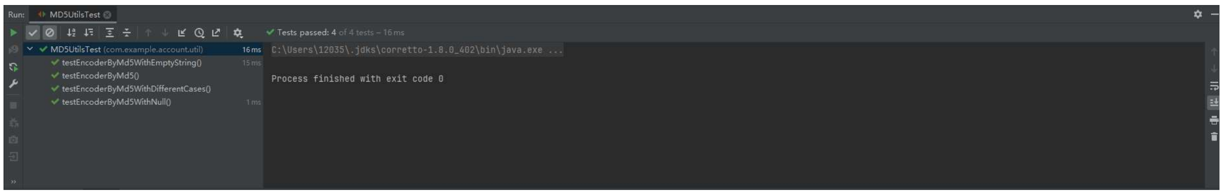
Testing Scope: MD5Utils

Test Method: Junit

Test Cases	Expected Test Results	Actual Test Results
1. ‘testEncoderByMd5’: Encode a known string to MD5 and compare with expected Base64 output 2. ‘testEncoderByMd5WithEmptyString’: Encode an empty string to MD5 and Base64 3. ‘testEncoderByMd5WithNull’: Verify that encoding ‘null’ throws ‘IllegalArgumentException’ 4. ‘testEncoderByMd5WithDifferentCases’: Compare the MD5 hash of similar but case-different strings	1. The encoded MD5+Base64 for "hello world" matches expected output 2. The encoded MD5+Base64 for an empty string matches expected output 3. ‘IllegalArgumentException’ is thrown when encoding ‘null’ 4. Case-sensitive strings yield distinct	1. The encoded output matches expected MD5+Base64 of "hello world" 2. The encoded output matches expected MD5+Base64 of an empty string 3. Encoding ‘null’ ‘raised’ ‘IllegalArgumentException’ 4. Case-sensitive strings produce unique

	MD5+Base64 hashes	MD5+Base64 hashes

Table 13. MD5UtilsTest Result



```
✓ Tests passed: 4 of 4 tests – 16 ms
✓ MD5UtilsTest (com.example.account.util)   16 ms
    ✓ testEncoderByMd5WithEmptyString()      15 ms
    ✓ testEncoderByMd5()
    ✓ testEncoderByMd5WithDifferentCases()
    ✓ testEncoderByMd5WithNull()              1 ms
```

```
C:\Users\12035\.jdks\corretto-1.8.0_402\bin\java.exe ...

Process finished with exit code 0
```

Figure 35. MD5UtilsTest Result

4.2.2 Non-functional Testing

JMeter is a popular open-source tool used for load testing, with the primary purpose of evaluating the performance and scalability of web applications and services. By simulating concurrent user interactions, JMeter allows teams to assess how a system behaves under different loads, ranging from light traffic to extreme stress scenarios (Palomäki, 2009). This capability is crucial for identifying performance bottlenecks, detecting system weaknesses, and ensuring that applications can handle expected user volumes without degradation in response times or reliability.

The benefits of using JMeter for load testing are numerous. It offers flexibility in designing complex test scenarios, supports a wide range of protocols (including HTTP and JDBC), and allows for distributed testing, enabling testers to simulate large-scale loads across multiple machines. JMeter's graphical interface and extensive plugin support make it accessible to users with varying levels of expertise, while its detailed reporting capabilities provide valuable insights into system performance metrics. By using JMeter, development teams can proactively address potential performance issues, ensuring that applications are robust, responsive, and capable of delivering a seamless user experience under varying conditions.

Test Method: JMeter

Test Configuration:

a) Multiple thread groups were used to simulate concurrent user activity during the test.
b) A total of 8100 samples were generated during the testing period.

Scope: The test focused on different user interactions with the system, simulating typical operations such as adding, modifying, and querying data. The objective was to ensure the system's reliability and performance under concurrent user activity.

Overall APDEX Score: 1.000

This indicates that the system's performance met or exceeded user expectations within the defined thresholds.

Request Summaries:

a) Total Requests: 8100
b) Pass Rate: 100% (all requests were successful without errors)
c) Average Response Time: 2.91 ms
d) Max Response Time: 40.66 ms (the maximum response time recorded during the test)

Overall, the test results indicate a stable and efficient system. The APDEX score of 1.000 and a 100% success rate suggest that the system meets user expectations. The response times were within acceptable ranges, and no significant errors were detected during the test. These results suggest that the system is ready for production use or further testing with increased load scenarios.

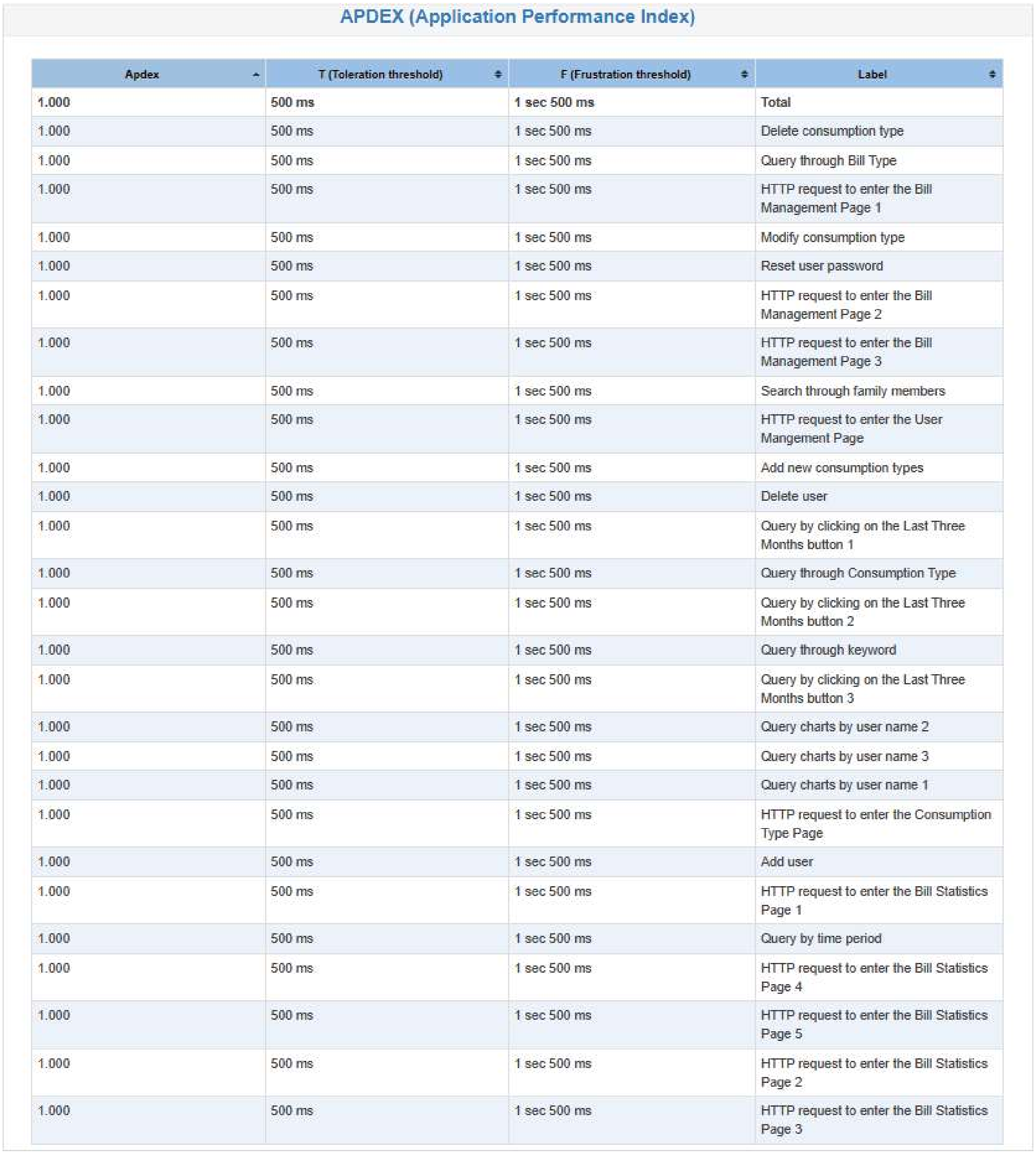
Test Case	HTTP Request Type	Path	Body Data
Enter the Bill Management Page	Get	a) /bill/query?houseId=3&userId=9&consumptionId=&billType=&onlySelf=false&pageNum=1&pageSize=10&remark= b) /user/query?houseId=3 c) /consumptionType/query?houseId=3	Null

Query through keyword	Get	/bill/query?houseId=4&userId=10&consumptionId=&billType=&onlySelf=false&pageNum=1&pageSize=10&remark=200	Null
Query through Bill Type	Get	/bill/query?houseId=4&userId=10&consumptionId=&billType=2&onlySelf=false&pageNum=1&pageSize=10&remark=	Null
Query by time period	Get	/bill/query?houseId=4&userId=10&consumptionId=&billType=2&onlySelf=false&beginDate=2024-04-12+00:00:00&endDate=2024-04-30+00:00:00&pageNum=1&pageSize=10&remark=	Null
Search through family members	Get	/bill/query?houseId=4&userId=10&consumptionId=&billType=2&onlySelf=true&beginDate=2024-04-12+00:00:00&endDate=2024-04-30+00:00:00&pageNum=1&pageSize=10&remark=	Null
Query through Consumption Type	Get	/bill/query?houseId=4&userId=10&consumptionId=2&billType=&onlySelf=false&pageNum=1&pageSize=10&remark=	Null
Enter the Bill Statistics Page	Get	a) /bill/statisticsByTime?houseId=4&userId=10&timeType=1&onlySelf=false b) /user/query?houseId=3 c) /bill/statisticsLineChart?houseId=3&userId=9&timeType=1&onlySelf=false d) /bill/statisticsType?houseId=3&userId=9&timeType=1&onlySelf=false	Null
Query by clicking on the Last Three Months button	Get	a) /bill/statisticsByTime?houseId=4&userId=10&timeType=2&onlySelf=false b) /bill/statisticsLineChart?houseId=4&userId=10&timeType=2&onlySelf=false c) /bill/statisticsType?houseId=4&userId=10&timeType=2&onlySelf=false	Null

Query charts by user name	Get	a) /bill/statisticsByTime?houseId=4&userId=10&timeType=1&onlySelf=true b) /bill/statisticsLineChart?houseId=4&userId=10&timeType=1&onlySelf=true c) /bill/statisticsType?houseId=4&userId=10&timeType=1&onlySelf=true	Null
Enter the Consumption Type Page	Get	/consumptionType/query?houseId=3	Null
Add new consumption types	Post	/consumptionType/add	{"name":"qww","houseId":"3"}
Modify consumption type	Put	/consumptionType/updateName	{"houseId":"3","id":28,"newName":"qwww"}
Delete consumption type	Put	/consumptionType/deleteType	{"id":28}
Enter the User Management Page	Get	/user/query?houseId=3&pageNum=1&pageSize=10	Null
Add user	Post	/user/add	{"houseId":"3","name":"user06","password":"123456

			7q","isAdmin":0}
Delete user	Put	/user/remove	{"adminUserId":"9","removeUserId":12}
Reset user password	Put	/user/resetPwd	{"id":14}

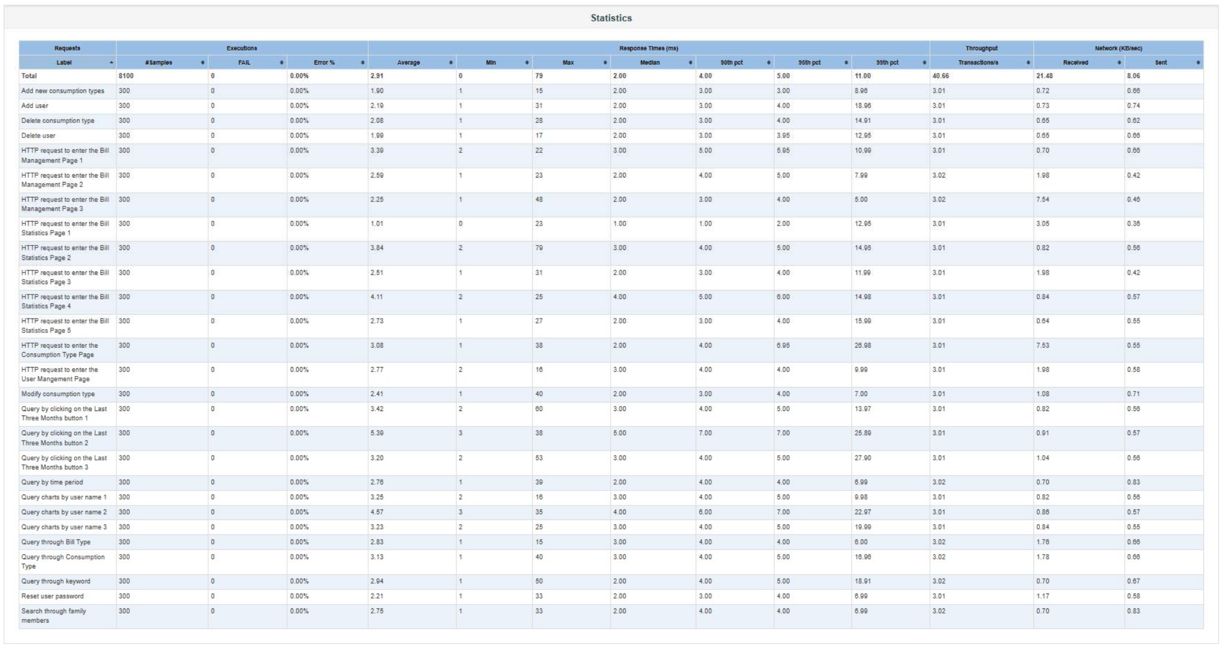
Table 14. Information on different testing plans



APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	**500 ms**	**1 sec 500 ms**	**Total**
1.000	500 ms	1 sec 500 ms	Delete consumption type
1.000	500 ms	1 sec 500 ms	Query through Bill Type
1.000	500 ms	1 sec 500 ms	HTTP request to enter the Bill Management Page 1
1.000	500 ms	1 sec 500 ms	Modify consumption type
1.000	500 ms	1 sec 500 ms	Reset user password
1.000	500 ms	1 sec 500 ms	HTTP request to enter the Bill Management Page 2
1.000	500 ms	1 sec 500 ms	HTTP request to enter the Bill Management Page 3
1.000	500 ms	1 sec 500 ms	Search through family members
1.000	500 ms	1 sec 500 ms	HTTP request to enter the User Mangement Page
1.000	500 ms	1 sec 500 ms	Add new consumption types
1.000	500 ms	1 sec 500 ms	Delete user
1.000	500 ms	1 sec 500 ms	Query by clicking on the Last Three Months button 1
1.000	500 ms	1 sec 500 ms	Query through Consumption Type
1.000	500 ms	1 sec 500 ms	Query by clicking on the Last Three Months button 2
1.000	500 ms	1 sec 500 ms	Query through keyword
1.000	500 ms	1 sec 500 ms	Query by clicking on the Last Three Months button 3
1.000	500 ms	1 sec 500 ms	Query charts by user name 2
1.000	500 ms	1 sec 500 ms	Query charts by user name 3
1.000	500 ms	1 sec 500 ms	Query charts by user name 1
1.000	500 ms	1 sec 500 ms	HTTP request to enter the Consumption Type Page
1.000	500 ms	1 sec 500 ms	Add user
1.000	500 ms	1 sec 500 ms	HTTP request to enter the Bill Statistics Page 1
1.000	500 ms	1 sec 500 ms	Query by time period
1.000	500 ms	1 sec 500 ms	HTTP request to enter the Bill Statistics Page 4
1.000	500 ms	1 sec 500 ms	HTTP request to enter the Bill Statistics Page 5
1.000	500 ms	1 sec 500 ms	HTTP request to enter the Bill Statistics Page 2
1.000	500 ms	1 sec 500 ms	HTTP request to enter the Bill Statistics Page 3

Figure 36. Test APDEX results


Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**8100**	**0**	**0.00%**	**2.91**	**0**	**79**	**2.00**	**4.00**	**5.00**	**11.00**	**40.66**	**21.48**	**8.06**
Add new consumption types	300	0	0.00%	1.90	1	15	2.00	3.00	3.00	8.96	3.01	0.72	0.66
Add user	300	0	0.00%	2.19	1	31	2.00	3.00	4.00	18.96	3.01	0.73	0.74
Delete consumption type	300	0	0.00%	2.08	1	28	2.00	3.00	4.00	14.91	3.01	0.65	0.62
Delete user	300	0	0.00%	1.99	1	17	2.00	3.00	3.95	12.95	3.01	0.65	0.66
HTTP request to enter the Bill Management Page 1	300	0	0.00%	3.39	2	22	3.00	5.00	5.95	10.99	3.01	0.70	0.66
HTTP request to enter the Bill Management Page 2	300	0	0.00%	2.59	1	23	2.00	4.00	5.00	7.99	3.02	1.98	0.42
HTTP request to enter the Bill Management Page 3	300	0	0.00%	2.25	1	48	2.00	3.00	4.00	5.00	3.02	7.54	0.46
HTTP request to enter the Bill Statistics Page 1	300	0	0.00%	1.01	0	23	1.00	1.00	2.00	12.95	3.01	3.05	0.36
HTTP request to enter the Bill Statistics Page 2	300	0	0.00%	3.84	2	79	3.00	4.00	5.00	14.95	3.01	0.82	0.56
HTTP request to enter the Bill Statistics Page 3	300	0	0.00%	2.51	1	31	2.00	3.00	4.00	11.99	3.01	1.98	0.42
HTTP request to enter the Bill Statistics Page 4	300	0	0.00%	4.11	2	25	4.00	5.00	6.00	14.98	3.01	0.84	0.57
HTTP request to enter the Bill Statistics Page 5	300	0	0.00%	2.73	1	27	2.00	3.00	4.00	15.99	3.01	0.64	0.55
HTTP request to enter the Consumption Type Page	300	0	0.00%	3.08	1	38	2.00	4.00	6.95	26.98	3.01	7.53	0.55
HTTP request to enter the User Mangement Page	300	0	0.00%	2.77	2	16	3.00	4.00	4.00	9.99	3.01	1.98	0.58
Modify consumption type	300	0	0.00%	2.41	1	40	2.00	3.00	4.00	7.00	3.01	1.08	0.71
Query by clicking on the Last Three Months button 1	300	0	0.00%	3.42	2	60	3.00	4.00	5.00	13.97	3.01	0.82	0.56
Query by clicking on the Last Three Months button 2	300	0	0.00%	5.39	3	38	5.00	7.00	7.00	25.89	3.01	0.91	0.57
Query by clicking on the Last Three Months button 3	300	0	0.00%	3.20	2	53	3.00	4.00	5.00	27.90	3.01	1.04	0.56
Query by time period	300	0	0.00%	2.76	1	39	2.00	4.00	4.00	6.99	3.02	0.70	0.83
Query charts by user name 1	300	0	0.00%	3.25	2	16	3.00	4.00	5.00	9.98	3.01	0.82	0.56
Query charts by user name 2	300	0	0.00%	4.57	3	35	4.00	6.00	7.00	22.97	3.01	0.86	0.57
Query charts by user name 3	300	0	0.00%	3.23	2	25	3.00	4.00	5.00	19.99	3.01	0.84	0.55
Query through Bill Type	300	0	0.00%	2.83	1	15	3.00	4.00	4.00	6.00	3.02	1.76	0.66
Query through Consumption Type	300	0	0.00%	3.13	1	40	3.00	4.00	5.00	16.96	3.02	1.78	0.66
Query through keyword	300	0	0.00%	2.94	1	50	2.00	4.00	5.00	18.91	3.02	0.70	0.67
Reset user password	300	0	0.00%	2.21	1	33	2.00	3.00	4.00	6.99	3.01	1.17	0.58
Search through family members	300	0	0.00%	2.75	1	33	2.00	4.00	4.00	6.99	3.02	0.70	0.83

Figure 37. Detailed test data for each test plan

Chapter 5 Professional Issues

5.1 Project Management

5.1.1 Activities

Objective	Task	Completion Status
Complete a background check on personal expenditure and income tracking websites.	1) Research and review existing personal finance tracking websites. 2) Document key features and shortcomings.	All done
Understand the technology required for the website and determine project requirements.	1) Identify the technology stack for front-end and back-end development. 2) Gather and document project requirements based on user feedback and research.	All done

Design an intuitive interface for users to access and use easily.	Create wireframes and mockups for the user interface.	All done
Implement login and registration functions.	1) Develop user registration functionality. 2) Create a secure login system.	All done
Implement cost recording function.	Develop the functionality for users to record their expenses and income.	All done
Implement the statistical chart display function of user expenses	Design a line chart to represent the changes in user income and expenses over time, and then design a sector chart to represent the proportion of user income and expenses.	All done
Implement expense classification function and customize expense categories	Create a system for categorizing expenses and allow users to customize categories.	All done
Implement user's income and expenditure statistics over a period of time.	Design four buttons to represent the past month, the past three months, the past six months and the past year respectively. Implement the corresponding display of user income and expenses during this time	All done

	period when clicking the button.	
Carry out test.	Conduct various testing types, including unit testing, integration testing, and user acceptance testing.	All done
Present the work to the audience.	Organize presentations to showcase the project to the target audience.	All done

Table 15. The Objective and Task of Activities.

5.1.2 Schedule

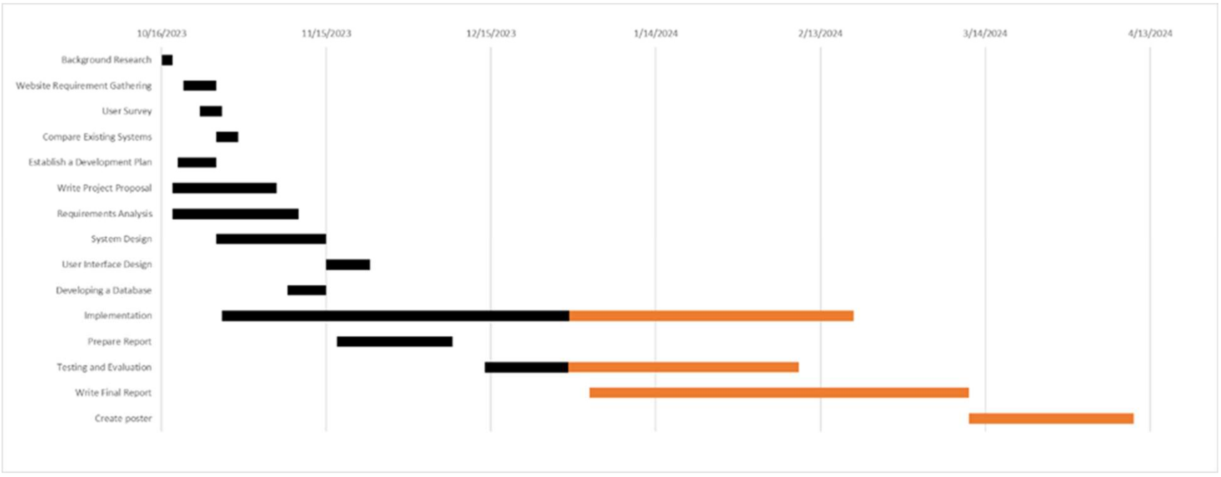


Figure 38. Project Gantt Chart.

The black part represents the completed part, while the yellow part represents the unfinished part.

Project Stages: The Gantt chart shows tasks from early stages of research and planning to later stages of design, implementation, testing, and documentation.

Timeline and Task Durations:

The project starts on October 16, 2023, and extends until April 10, 2024.

Task durations range from 2 days to 115 days, indicating significant variability in task lengths.

Critical Tasks:

1) Initial Research and Planning:

Background Research (2 days) and Website Requirement Gathering (6 days) are some of the earliest tasks, focusing on understanding project requirements.

2) Design and Development:

Tasks like System Design (20 days), Developing a Database (7 days), and User Interface Design (8 days) illustrate the core technical work required for the project.

3) Implementation and Testing:

The Implementation phase is the longest task (115 days), indicating the critical path for project completion.

Testing and Evaluation (57 days) occurs in parallel with other tasks like Prepare Report and Write Final Report, suggesting thorough validation.

4) Reporting and Documentation:

Write Project Proposal (19 days), Prepare Report (21 days), and Write Final Report (69 days) are key documentation milestones throughout the project timeline.

Create poster (30 days) represents a visual representation task at the project's conclusion.

Overlapping Tasks and Dependencies:

1) Several tasks overlap, indicating concurrent work and dependencies. For example:

Requirements Analysis (23 days) runs from October 18 to November 10, intersecting with other tasks.

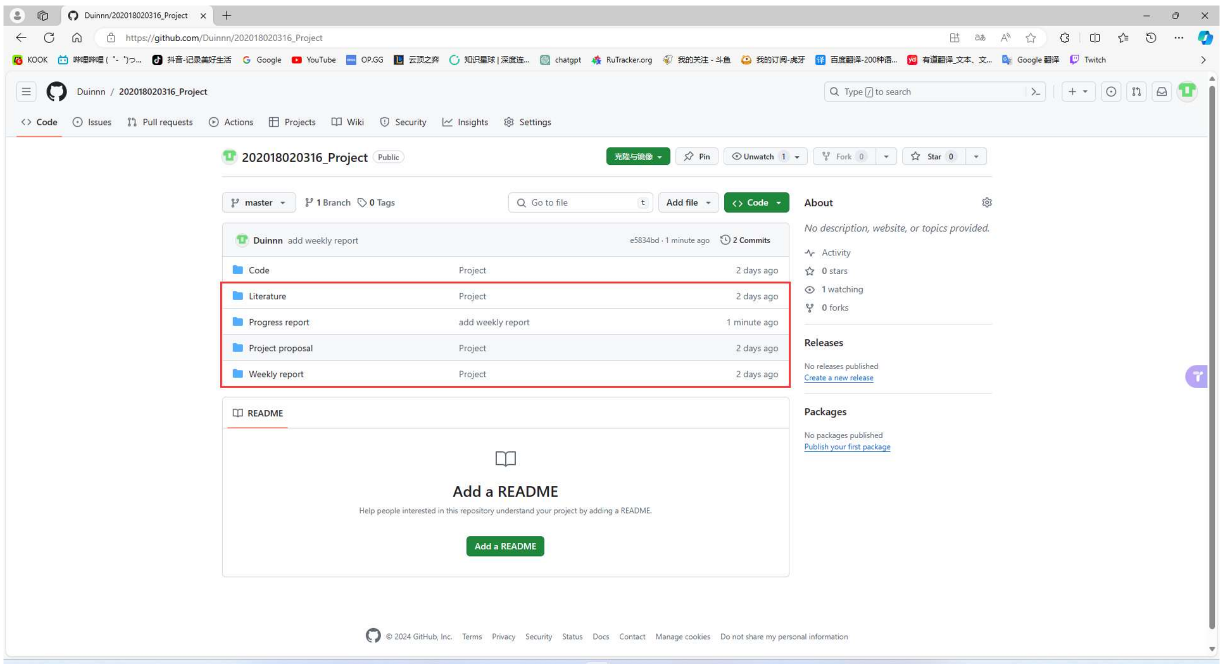
System Design overlaps with Developing a Database, emphasizing the need for coordination among teams.

2) Overlaps between Testing and Evaluation and Implementation indicate that testing occurs during ongoing development, reflecting iterative testing practices.

This Gantt chart outlines a detailed project timeline with clear task sequences, durations, and dependencies, providing a comprehensive overview of your project's workflow from research to final documentation and presentation.

5.1.3 Project Data Management

First, I open Git Bash in the document folder. Then, in the Git Bash command, I entered "git status" to check the status of the git repository, indicating that I have untracked files in the "" directory. Then enter "git add" to temporarily store all changes in the working directory, including untracked files. Then enter "git status" again to check the status display. The new file in the directory is ready for submission. Finally, enter "git push" to push the changes to a remote repository on GitHub.



Duinnn add weekly report e5834bd · 1 minute ago 2 Commits

Code	Project	2 days ago
Literature	Project	2 days ago
Progress report	add weekly report	1 minute ago
Project proposal	Project	2 days ago
Weekly report	Project	2 days ago

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

README

Add a README

Help people interested in this repository understand your project by adding a README.

Add a README

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Figure 39. Project Data Management with Git

5.1.4 Project Deliverables

Submitted documents:

Project proposal, Project code, Literature, Weekly report, Progress report, Project software, Final report

5.2 Risk Analysis

Cause ID	Potential Causes	Severity (1-5)	Likelihood (1-5)	Risk (1-20)	Mitigation ID	Mitigation
C1	Security breaches (unauthorized access, data leakage)	5	3	15	M 1	Implement robust security protocols, conduct security testing regularly
C2	Data loss (corruption, accidental deletion)	4	2	8	M 2	Implement regular backups, use version control systems
C3	Overdependence on third-party services (APIs, libraries)	3	3	9	M 3	Ensure proper licensing, have backup services or local solutions

C4	Regulatory compliance failure (data protection laws)	5	2	10	M 4	Consult legal experts, ensure the system meets all relevant legal requirements
C5	Technical debt accumulation (due to rushed releases, etc.)	3	4	12	M 5	Adhere to coding standards, allow time for refactoring, use code reviews

Table 16. Risk analysis

5.3 Professional Issues

5.3.1 Legal Issues

- Data Protection Laws: Ensure compliance with data protection laws such as GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act), which govern the collection, processing, and storage of personal data. Implement robust security measures, encryption, and access controls to safeguard users' financial information.
- Intellectual Property Rights: Respect intellectual property rights by obtaining proper licenses for third-party software components, ensuring that code and design elements are original or properly licensed, and avoiding infringement of copyrights or trademarks.
- Regulatory Compliance: Research and adhere to regulatory requirements specific to financial software, such as anti-money laundering regulations or consumer protection

laws. Compliance with regulations ensures legal operation and protects both users and the development team from legal liabilities.

5.3.2 Social Issues

1) Accessibility: Design the system to be accessible to users with disabilities, following Web Content Accessibility Guidelines (WCAG). Provide alternative text for images, keyboard navigation options, and ensure compatibility with screen readers to accommodate users with various disabilities.
2) Digital Divide: Address disparities in access to technology by designing a system that is compatible with different devices and internet connections. Consider providing offline functionality or mobile app versions for users with limited internet access.
3) Financial Literacy: Integrate educational resources, tutorials, or interactive tools within the system to enhance users' financial literacy. Provide explanations of financial terms, tips for budgeting and saving, and real-time feedback on spending habits to empower users to make informed financial decisions.

5.3.3 Ethical Issues

1) Transparency and Consent: Clearly communicate to users how their data will be used and obtain their consent for data processing activities. Provide options for users to control their privacy settings and opt out of data sharing if desired.
2) Fair Treatment: Avoid biases in algorithmic decision-making or data analysis that may disproportionately affect certain demographic groups. Implement fairness measures to ensure equitable treatment of all users regardless of race, gender, or socioeconomic status.
3) Confidentiality: Protect the confidentiality of users' financial information by implementing stringent access controls, encryption techniques, and regular security audits. Respect users' privacy rights and refrain from sharing their data with third parties without explicit consent.

5.3.4 Environment Issues

1) Energy Efficiency: Choose hosting providers and infrastructure technologies that prioritize energy efficiency and renewable energy sources. Optimize code and system architecture to minimize resource consumption and reduce carbon emissions associated with server operations.

2) Sustainability: Consider the environmental impact of hardware procurement, software development, and system maintenance activities. Use eco-friendly materials and sustainable practices throughout the project lifecycle, such as recycling electronic waste and reducing paper usage in documentation and communication.

Chapter 6 Conclusion

Managing personal finances effectively is a common challenge in today's increasingly complex financial landscape. To address this issue, this paper presents the development of a web-based Comprehensive Personal Expenses Tracking System. The system aims to simplify the tracking of personal income and expenses, enabling users to promptly assess their financial status and exert control over their financial well-being.

The paper delineates the objectives and methodology employed in developing the Comprehensive Personal Expenses Tracking System. Through a structured approach, the project successfully achieves several milestones. It begins with a comprehensive background review of existing personal finance tracking websites, laying the groundwork for system development. The Waterfall Model is adopted for software development, encompassing activities such as requirements gathering, design, implementation, testing, deployment, evaluation, and operation maintenance. A robust technology stack comprising Java, HTML, Spring, SpringBoot, Mybatis, Vue.js, ElementUI, MySQL 8.0, IntelliJ IDEA, Visual Studio Code, and Apache Tomcat 9.0 is selected for system development.

Implementation and results showcase the successful integration of various features, including user registration and login functionalities, bill management (addition, editing, searching), statistical chart display, expense classification, user management, password management, and account exit. Extensive user interface designs and forms are developed to ensure user-friendly interactions.

Professional issues such as project management, project data management, risk analysis, legal, social, ethical, and environmental considerations are addressed. Potential risks are identified, and mitigation strategies are proposed, emphasizing legal compliance, social inclusivity, ethical conduct, and environmental sustainability throughout the development process.

Future work entails conducting various testing types (unit testing, integration testing, user acceptance testing) and presenting the work to the target audience. Additionally, future iterations could focus on enhancing system functionalities, refining user experience, implementing additional security measures, and integrating educational resources to improve users' financial literacy.

References

Gomathy, C K. (2022) 'EXPENDITURE MANAGEMENT SYSTEM', *ENGINEERING, COMPUTING & ARCHITECTURE*, pp.182-184. Available at: https://www.researchgate.net/publication/360620084_EXPENDITURE_MANAGEMENT_SYSTEM, (Accessed: October 27 2023).

Gupta, Hrithik. et al. (2020) *Expense Tracker: A Smart Approach to Track Everyday Expense*, Available at: https://easychair.org/publications/preprint/73S7, (Accessed: October 27 2023).

Patil, Prof. Pallavi. et al. (2023) 'Personal Expenses Tracker', *International Journal of Research Publication and Reviews*, pp. 4357-4359. Available at: https://ijrpr.com/uploads/V4ISSUE3/IJRPR10840.pdf, (Accessed: October 27 2023).

Palomäki, J. (2009) *Web Application Performance Testing.* Available at: https://core.ac.uk/works/57940966, (Accessed: October 27 2023).

García, B. and Dueñas, J.C. (2011) 'Automated Functional Testing based on the Navigation of Web Applications', *EPTCS*, 61, pp. 49-65. Available at: https://arxiv.org/abs/1108.2357 (Accessed: 03 May 2024).

Cleary, J.G., and Inglis, S.J. (2007) *Jumble Java Byte Code to Measure the Effectiveness of Unit Tests*. Available at: https://core.ac.uk/works/15105786 (Accessed: 03 May 2024).

Appendices