POSTS AND TELECOMMUNICATIONS
INSTITUTE OF TECHNOLOGY
Faculty of Information Technology 1

FINAL REPORT
FOUNDATION INTERNSHIP

ChippyCash

Smart Chatbot for Easier Expense Management

Instructor : **Kim Ngoc Bach**
Student Name : **Dinh Manh Hung**
Student ID : **B22DCCN359**
Class : **E22TTNT**

Hanoi – 2025

Table of contents

I. Overview

1. Motivation

In the context of rapid advancements in digital technology, the need for effective personal financial management has become increasingly urgent for individuals. Many people struggle to track their income and expenses, plan budgets, or make sound financial decisions due to a lack of supportive tools. Traditional methods such as keeping handwritten records or using Excel spreadsheets, though common, have significant limitations in terms of flexibility, analytical capabilities, and convenience in today's fast-paced environment.

Recognizing this reality, the development of a modern personal finance management application integrated with advanced technologies like Artificial Intelligence (AI) has become an essential trend to meet the growing demands of users. Such an application is not only a tool for data entry but also serves as a smart "financial assistant," helping users analyze, forecast, and make more scientific and proactive financial decisions.

The "ChippyCash" project was created to address these challenges, offering a comprehensive, intelligent, and user-friendly solution for personal finance management. By leveraging modern technological achievements, "ChippyCash" aims to optimize user experience, enhance financial management efficiency, and promote healthy financial habits within the community.

2. Reason for Choosing the Topic

Originating from the real-world need both personal and shared by many others for a smart, user-friendly, and flexibly interactive expense management tool, the topic **"ChippyCash – Intelligent Chatbot for Personal Expense Management"** was chosen. The goal of the project is to address common challenges users face by providing an intuitive financial management platform, integrated with an AI-powered chatbot capable of quickly recording expenses (even from receipt images) and offering helpful suggestions.

In addition, developing a user-friendly website to serve as the primary interface for daily interactions with the chatbot is a key component of the project. The combination of an intelligent chatbot and an accessible web platform is expected to enhance the personal finance management experience for users.

3. Research Subjects

The project focuses on developing and deploying a personal finance management chatbot platform, with the following key technological components:

Web development frameworks and languages: PHP, HTML, CSS, JavaScript, Bootstrap

Database: MySQL

Chatbot development techniques and frameworks: Python, FastAPI, Large Language Models (LLMs) from OpenAI, LlamaIndex

Optical Character Recognition (OCR) technology for processing receipt images

Methods for collecting, processing, and visualizing personal financial data

User Interface (UI) and User Experience (UX) design principles for financial applications

4. Objectives and Scope of the Research

The objective of this research is to explore and apply large language models (LLMs) and existing technologies to develop a personal finance management chatbot. Specifically, the study aims to investigate methods and technologies to build a chatbot capable of:

Recording income and expense transactions through natural language conversations

Analyzing and automatically extracting data from receipt images

Providing overviews of financial status and spending statistics

Offering personalized financial advice based on user data (e.g., the 50/30/20 rule)

The scope of this study focuses on the following key areas:

Explore website development technologies (PHP, HTML, CSS, JavaScript, Bootstrap).

Learn about large language models and effective usage methods (OpenAI API, LlamaIndex).

Build a chatbot that supports recording transactions, analyzing invoices, and providing financial advice.

Develop a website with a user-friendly interface to interact with chatbots and manage finances.

Through this research, the project aims to contribute to the advancement of personal chatbot technology, addressing society's growing need for smart and convenient financial management tools. By providing a 24/7 virtual assistant, chatbots can make it easier for users to track their spending and make more informed financial decisions.

II. Foundation Knowledge

1. Web Development Platforms

This section explores the key technologies and programming languages used in the development of the ChippyCash website, highlighting their benefits and common applications.

1.1 PHP (Hypertext Preprocessor)

PHP is a server-side scripting programming language that is widely used in web development, which stands out for its flexibility, open source, and strong support community. PHP enables building dynamic, highly interactive web applications, and has become the foundation for many well-known e-commerce websites and content management systems. In the ChippyCash project, PHP plays a central role in processing the backend-side business logic, including user authentication and management, performing operations with databases such as adding, editing, deleting, querying financial transactions, as well as developing APIs for the front-end interface. The use of PHP helps ensure stable performance and flexible integration with a wide range of databases, especially MySQL. PHP's scalability and easy maintenance also support long-term project development and allow for seamless interoperability with modern frontend technologies

1.2 HTML, CSS, JavaScript

HTML (HyperText Markup Language), CSS (Cascading Style Sheets) and JavaScript are the three core technologies in web user interface development, which play a fundamental role in building and operating modern web applications.

HTML provides content structure, organizes elements on a website such as headers, tables, forms, and links, creating a premise for the browser to display data in a logical way.

CSS is responsible for formatting, presenting, optimizing the aesthetics and layout of web interfaces, ensuring consistency and visual appeal across a variety of devices.

JavaScript brings dynamism and interoperability to web applications, handles user events, performs client-side data validation, and updates website content without reloading the entire page [2].

The tight combination of HTML, CSS, and JavaScript helps create a smooth user experience, meets dynamic interaction requirements, and provides a foundation for integration with modern frameworks and libraries in web development.

1.3 Bootstrap

Bootstrap is a well-known open-source front-end framework, developed by Twitter, that provides a system of interface components and standardized layout grids, helping programmers build web interfaces quickly and efficiently. from desktop to mobile. In addition, Bootstrap also integrates many UI components such as buttons, forms, tables, and navigation bars, helping to speed up the development process and ensure consistency for the application interface [3]. In the ChippyCash project, Bootstrap was used to design an intuitive, easy-to-use user interface, increase user experience, and minimize the time spent building the interface manually.

1.4 MySQL

MySQL is a popular open-source relational database management system (RDBMS) that is popular for its processing speed, reliability, and flexible scalability. MySQL supports the SQL standard, which allows complex data queries to be performed, ensuring the integrity and consistency of stored information. In ChippyCash, MySQL serves as a central data warehouse, storing all information related to users, financial transactions, revenue and expenditure categories, as well as the history of interaction with the system. Choosing MySQL not only helps the project optimize data performance and security, but also facilitates the expansion or integration with modern backend technologies such as PHP, Python, or big data analytics BI systems in the future [4].

2. Web Development Platforms

2.1 Large Language Models

Large Language Models (LLMs), such as OpenAI's GPT series, are the foundation of modern chatbot technologies. These models are pre-trained on large amounts of text data, allowing them to generate human-like responses based on context and input. The large size of these models allows them to grasp the nuances of

language, making them proficient in understanding and generating coherent, contextually appropriate responses.

ChippyCash leverages this capability of LLMs (via the OpenAI API) to understand spending entry requirements, analyze chat content, and provide financial advice.

2.2 LlamaIndex Framework

LlamaIndex (formerly GPT Index) is a framework designed to support the development of chatbot applications that require efficient indexing and retrieval of information. It integrates with LLMs to perform context-based document retrieval and generate responses. LlamaIndex simplifies the implementation of enhanced retrieval systems (RAGs) by providing developers with tools to build and manage indexes of large collections of documents (e.g., user transaction history, financial rules), improve the efficiency and accuracy of chatbot responses.

In ChippyCash, LlamaIndex helps the chatbot access and use users' personal financial data in a safe and efficient way to provide appropriate analysis and advice.

2.3 FastAPI

FastAPI is a modern, high-speed web framework (asynchronous support) for building APIs in Python, based on standard Python-style suggestions. It is designed to be high-performance and easy to use, with OpenAPI documentation automation and support for asynchronous programming, making it ideal for real-time data processing or demanding applications.

In ChippyCash, FastAPI is used to build a backend API for chatbots, ensuring fast response times and good scalability when handling requests from users.

2.4 Invoice Image Processing (OCR)

Optical Character Recognition (OCR) technology enables the conversion of images containing text (e.g., snapshots of invoices) into text data that can be processed by a computer.

In ChippyCash, OCR is integrated (or plans to be integrated) so that users can take a photo of the invoice, and the chatbot will automatically extract important information such as the amount, store name, date to record the transaction, saving time and minimizing manual data entry errors. Files such as capcha_ocr.py and

mb_capcha_ocr in the project show that there is research and development related to OCR.

2.5 Function Calling

Function calling is a capability that allows Large Language Models to connect to external tools and APIs more reliably. Instead of just generating text, the LLM can determine when an external action is needed (e.g., fetching data from a database, calling a specific backend service) and output a JSON object containing the name of a function to call and the arguments it requires. This enables developers to define a set of functions that the LLM can "request" to be executed. For the ChippyCash project, function calling would be crucial for:

Translating natural language user requests (e.g., "record my lunch expense of 50k," "what's my current balance?") into specific, actionable calls to backend services.

Reliably extracting parameters like amounts, categories, or dates from user input to populate the arguments for these function calls (e.g., a record_transaction function).

The system would then execute the specified function with the provided arguments and could return the result to the LLM, which can then use this information to formulate a more informed and accurate response to the user.

III. Project Management Report

1. Requirement Analysis

a) Website development requirements

1. Login

Goal: Allow users to securely sign in to your app with a username and password.

Requirements:

Users must provide both a username and a password.

The system authenticates credentials against saved data.

Log in successfully, redirect to the dashboard.

Fails, displays an error, and allows retrying.

2. Login

Goal: Allow users to securely sign out and redirect to the home

page (or login page).

Requirements:

Offer a conspicuous sign-out option.

After logging out, redirect the user.

The user's work session must be safely terminated

3. **Sign up**

Objective: Allow new users to create an account by providing the necessary information.

Requirements:

The user provides information for the required fields (username, password, email).

Validate data (e.g., email formatting, password strength) and make sure the username is unique.

Registration is successful, redirecting to the login page or dashboard.

4. **Manage Revenue/Expenditure Transactions**

Objective: Allow users to add, view, edit, delete financial transactions.

Requirements:

Data entry form for new transactions: amount, description, type (revenue/expenditure), category, transaction date.

Displays a list of deals with filters (by date, type, category).

The function of editing and deleting transactions.

The system has a save-transactions.php file to save the transaction.

5. **Spend Portfolio Management**

Goal: Allows users to create, view, edit, and delete spend/income categories.

Requirements:

Interface to add new categories (name, type).

Displays a list of available categories.

Catalog modification and deletion function.

API get-categories.php to get catalog information.

6. **View Financial Statistics and Charts**

Objective: Provide users with an overview of their financial situation through reports and charts.

Requirements:

Displays a pie/column chart of your spending structure by category.

The line graph shows the revenue/expenditure trend over time.

Statistics on total revenue, total expenditure, and balance in different periods of time (weeks, months, years).

Statistics excel export function

7. **Personal Information Page and Settings**

Goal: Allow users to view and update personal information, change passwords.

Requirements:

Display user information (name, email).

Password change function.

profile.php page to manage personal information.

API settings.php for installations.

8. **Chat Interface with Chatbot**

Objective: Provide an interface for users to interact (chat) with ChippyCash.

Requirements:

Text input box to send the message.

Show conversation history in an organized way.

(Advanced) Supports uploading invoice images.

chatbot.php API for handling chatbot interactions.

9. **Banking Connection and Transaction Synchronization (MBBank)**

Objective: Allow users to connect MBBank accounts to automatically sync transactions.

Requirements:

Interface for users to enter MBBank login information securely.

Handle authentication with MBBank, including captcha solving

(if any).

Sync transaction history from MBBank to the ChippyCash system.

Displays synced transactions.

b) Development Requirements for Admins

1. Admin Login:

Goal: Allow administrators to log in to a separate admin area with higher privileges.

Request:

Use an account with the 'admin' role.

Private sign-in interface or decentralization after general sign-in (e.g., admin.php).

2. User Management:

Objective: Allow admins to manage all user accounts in the system.

Request:

View a list of all users (ID, username, email, creation date, status).

Edit user information

Delete user accounts (with a confirmation mechanism).

3. View System Statistics:

Objective: To provide admins with an overview of the operation of the entire system.

Request:

Displays the total number of registered users.

Displays the total number of transactions that have been recorded in the system.

API admin-stats.php to get statistical data for admin.

4. Database Backup and Restore:

Objective: Ensure data security for the system.

Request:

Database backup creation function.

5. System Configuration

Goal: Allow admins to adjust some general settings of the app.

Request:

Notification settings

Maintenance Mode

c) Chatbot Development Requirements

1. Recording and Financial Consulting via Chat:

The system must be able to conduct natural, friendly conversations about personal financial matters.

Support users to record revenues/expenses through descriptions in natural language (e.g., "spend 50k for breakfast", "receive salary 10 million this month").

Provide information about your financial situation (e.g., "how much did I spend on food this month?", "my current income and expenditure ratio?").

Offer savings advice, budget management based on spending data, and popular financial models (e.g., the 50/30/20 rule).

2. Invoice Image Analysis:

The system needs to be able to identify and extract information from the invoice image provided by the user (amount, date, main content).

Automatically suggest or record transactions based on extracted information

3. Diverse interactive roles

Provide different "roles" for the chatbot (e.g., Smart Assistant, Hot-tempered Mama, Homie) to increase attractiveness and match user preferences.

4. Storing and Retrieving Chat History:

Save the entire conversation history between the user and the chatbot.

The load_chat.py file on the backend handles the loading of chat history.

2. System Analysis and Design

a) Overall Design

- **Use cases:**
    - **User**

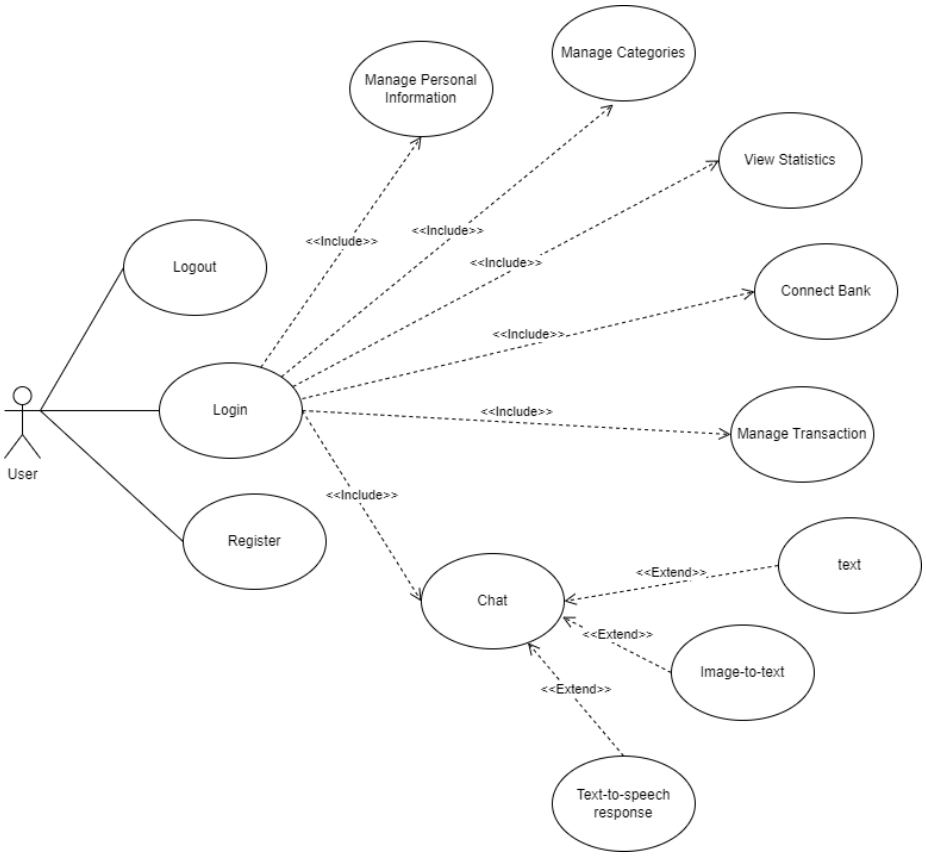


Figure 1. User Use case

Register: This use case allows users to register new accounts in the system.

Login: This use case allows users to log into their accounts to access all system functionalities.

Logout: This use case allows users to securely log out from their accounts.

Manage Personal Information: This use case allows users to

view and update their personal information, such as name, email, or contact details.

Manage Categories: This use case allows users to create, modify, or delete different transaction categories for better management.

View Statistics: This use case allows users to view summarized statistics and reports related to their transactions.

Connect Bank: This use case allows users to connect their bank accounts with the system for automatic transaction syncing.

Manage Transaction: This use case allows users to add, edit, or delete their financial transactions.

Chat: This use case allows users to interact with the chatbot after logging in successfully.

Chat with text: This use case allows users to communicate with the chatbot using text messages.

Chat with image-to-text: This use case allows users to upload images, which are then converted to text for chatbot interaction.

Text-to-speech response: This use case allows users to receive the chatbot's response in speech/audio form.

- **Admin**

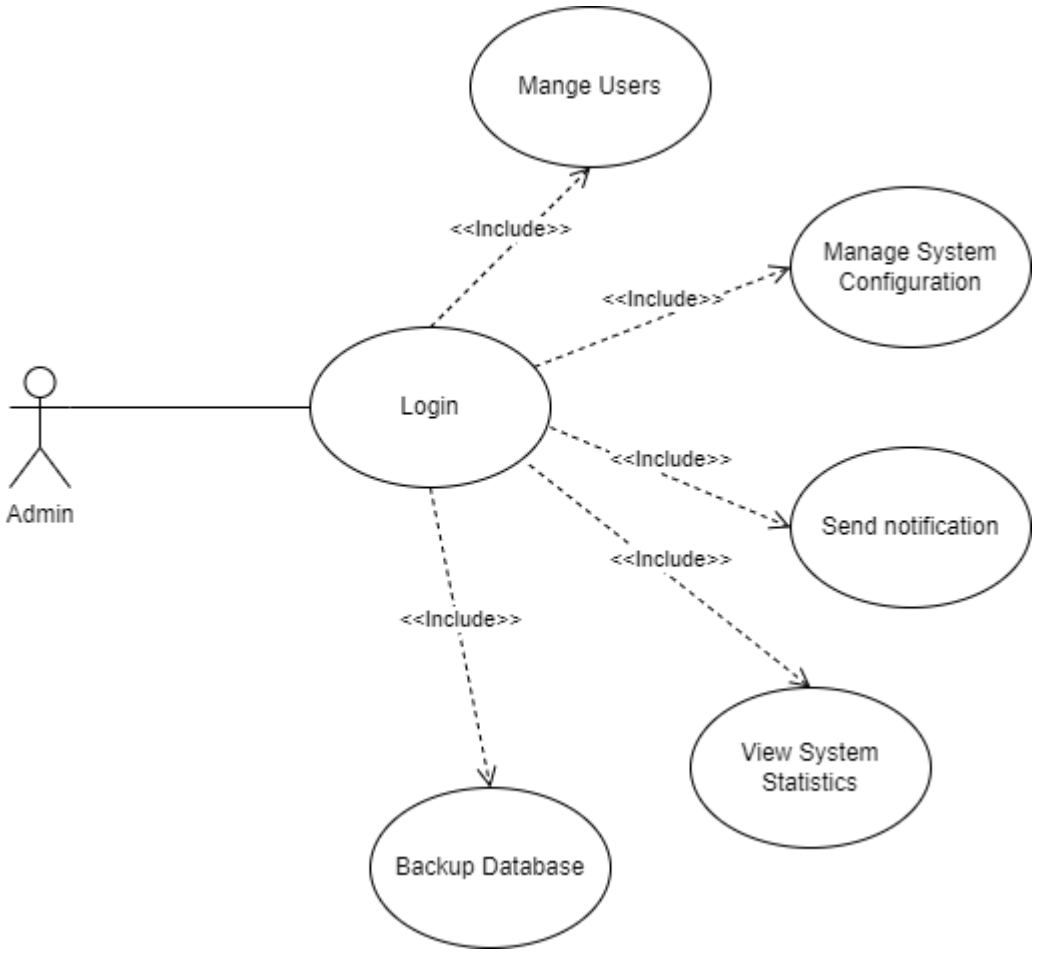


Figure 2. Admin Use case

Login: This use case allows the administrator to log into the system to access system management features.

Manage Users: This use case allows the administrator to create, update, or delete user accounts in the system.

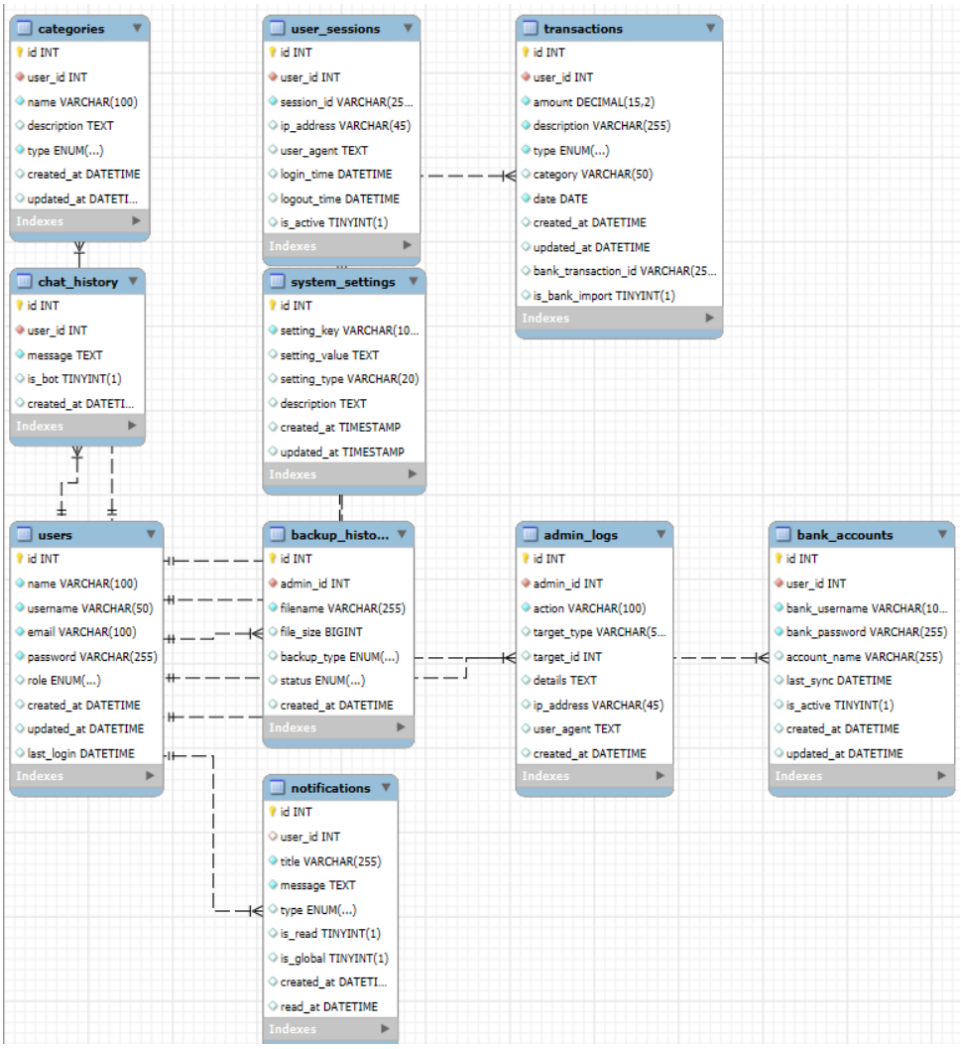
Manage System Configuration: This use case allows the administrator to view and modify system configuration settings.

Send Notification: This use case allows the administrator to send notifications or announcements to system users.

Backup Database: This use case allows the administrator to perform database backup operations for data safety and recovery.

View System Statistics: This use case allows the administrator to view statistics and reports related to system performance and usage.

- **Class diagram**



b) Implementation

1. Backend API

Chat Interaction

Method	Endpoint	Description
POST	/chat	Send chat messages from authenticated users (id_user, message, role) and receive responses from chatbots.
GET	/history	Retrieve the chat history of an authenticated user (id_user).

POST	/delete	Delete the data of an authenticated user (id_user), including chat history and related files.

Voice Generation

Method	Endpoint	Description
POST	/voice/stream	Stream audio files (.wav) created from text (voice_type) directly.

Image/Bill Analysis

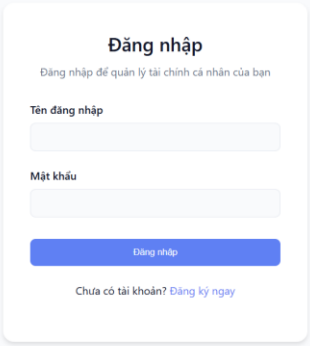
Method	Endpoint	Description
POST	/analyze-bill	Analyze the image of the uploaded invoice (file) with the input text (input_text) to extract the information.

Bank Data Retrieval

Method	Endpoint	Description
GET	/bank	Retrieve MBBank bank account information comprehensively, including balance and transaction history (username, password, proxy optional).

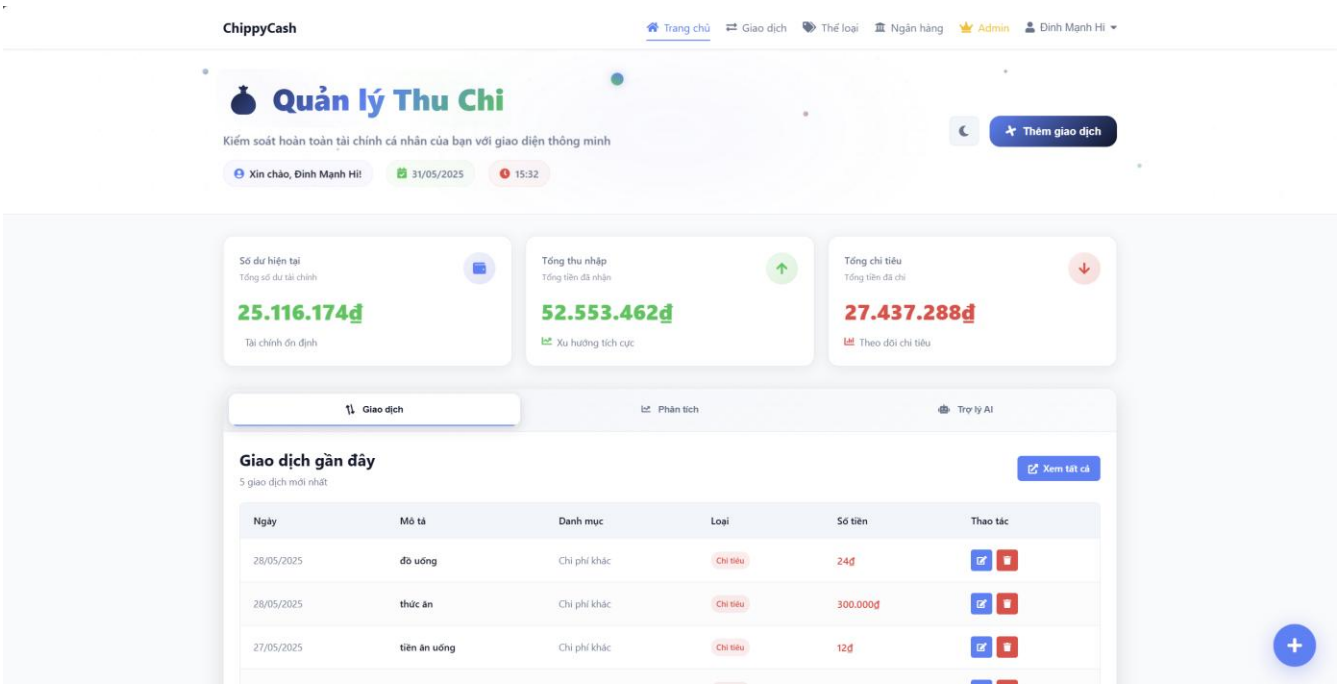
2. **Frontend**

User:

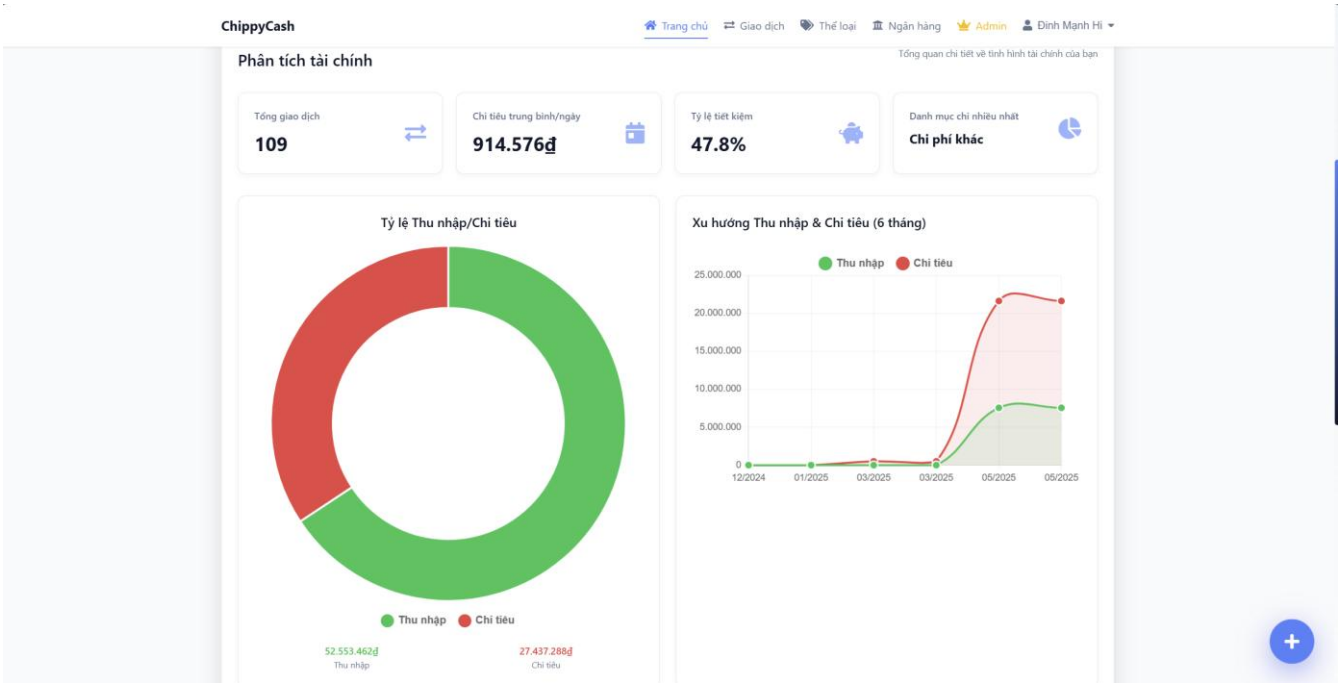


Login page interface

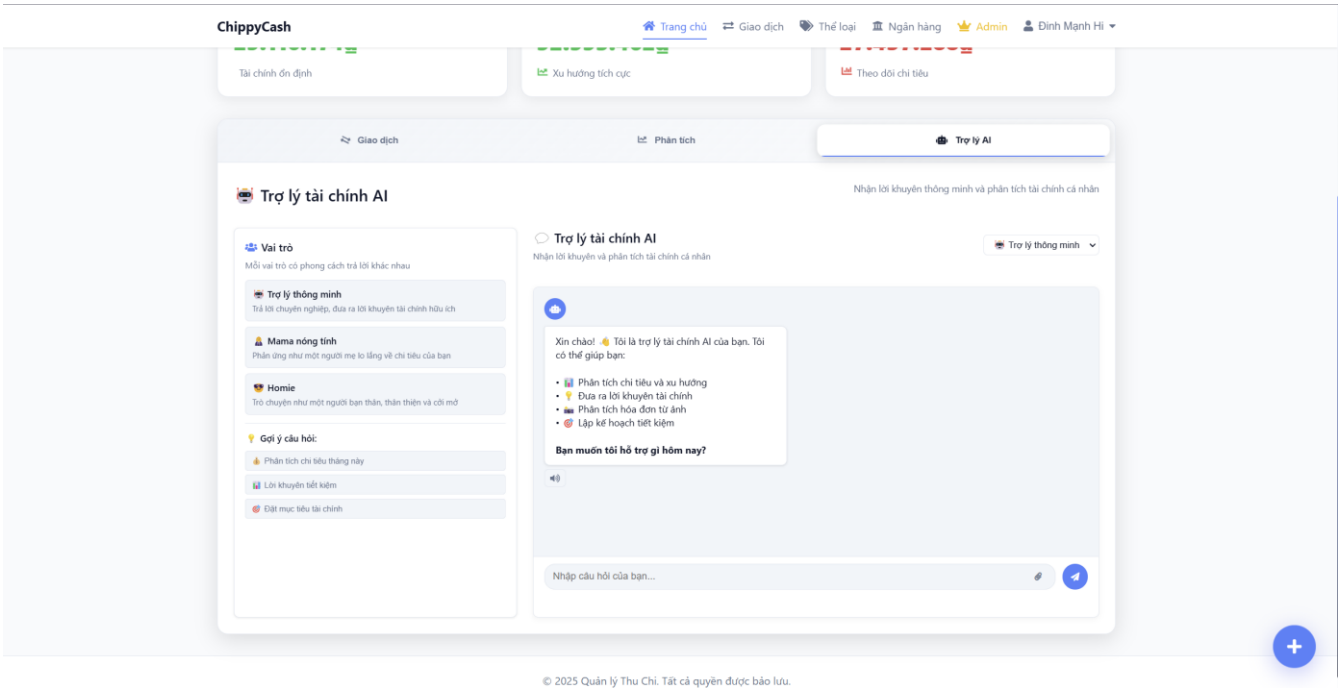
Registration page interface

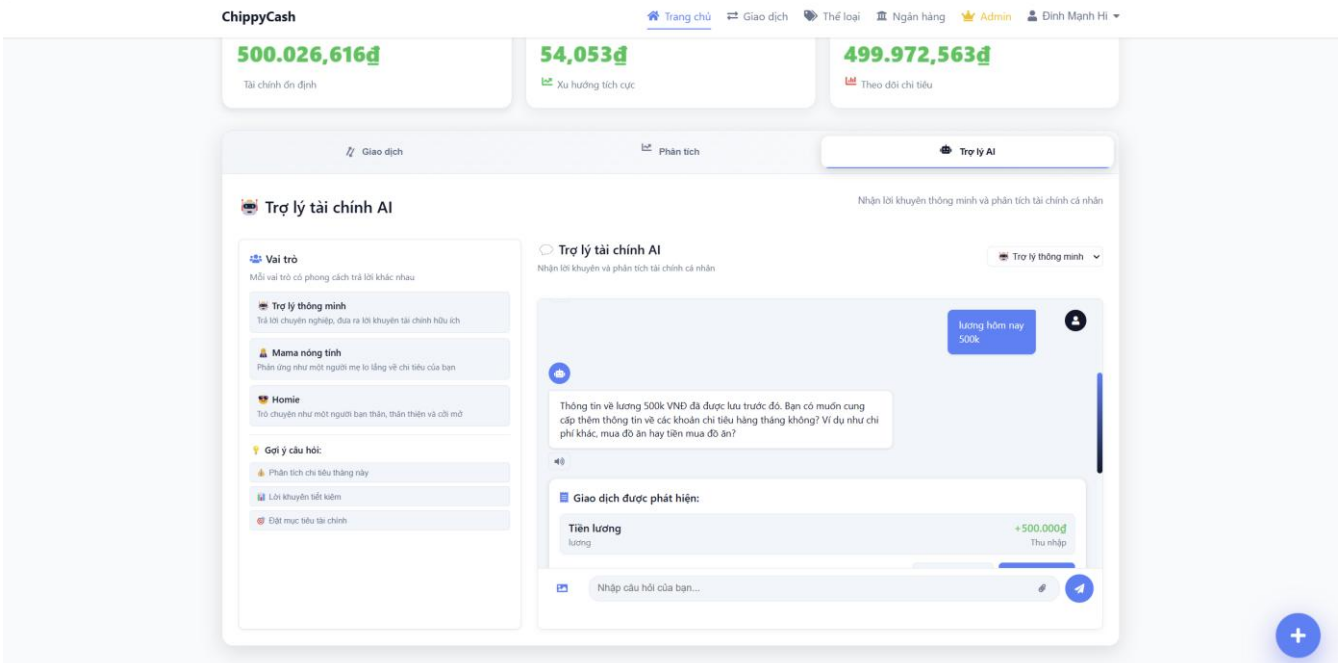


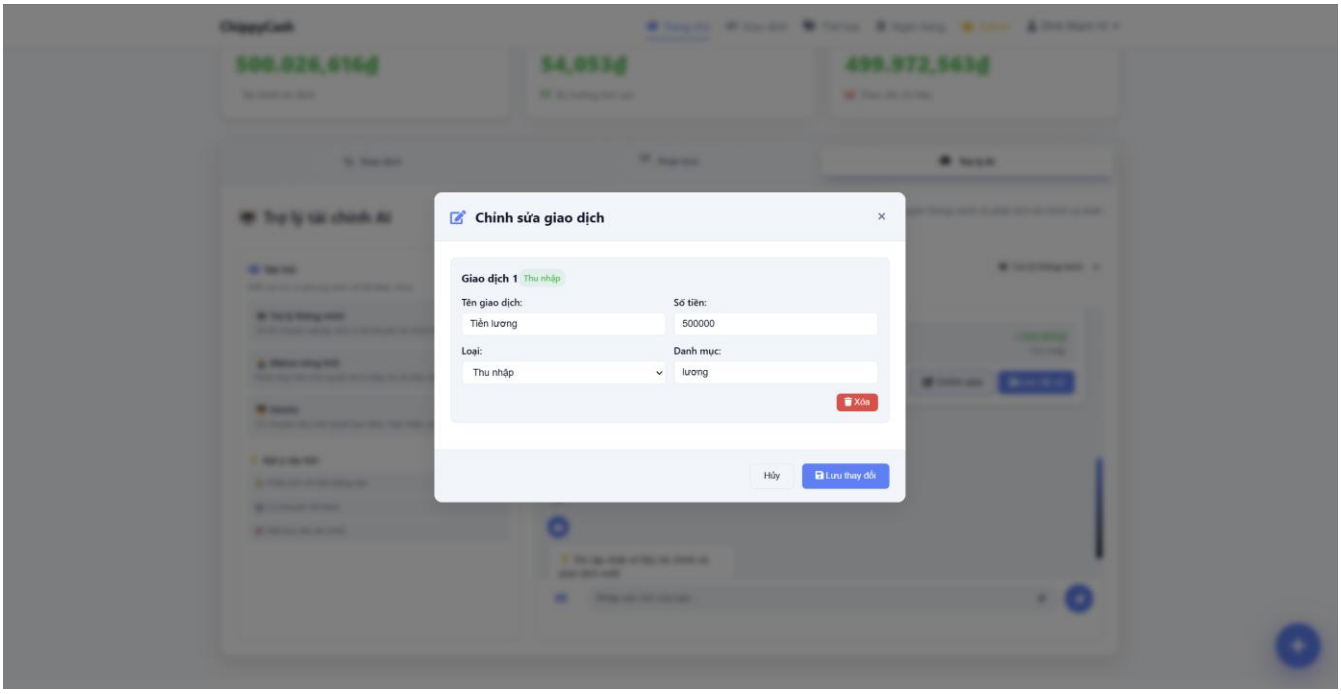
Home page



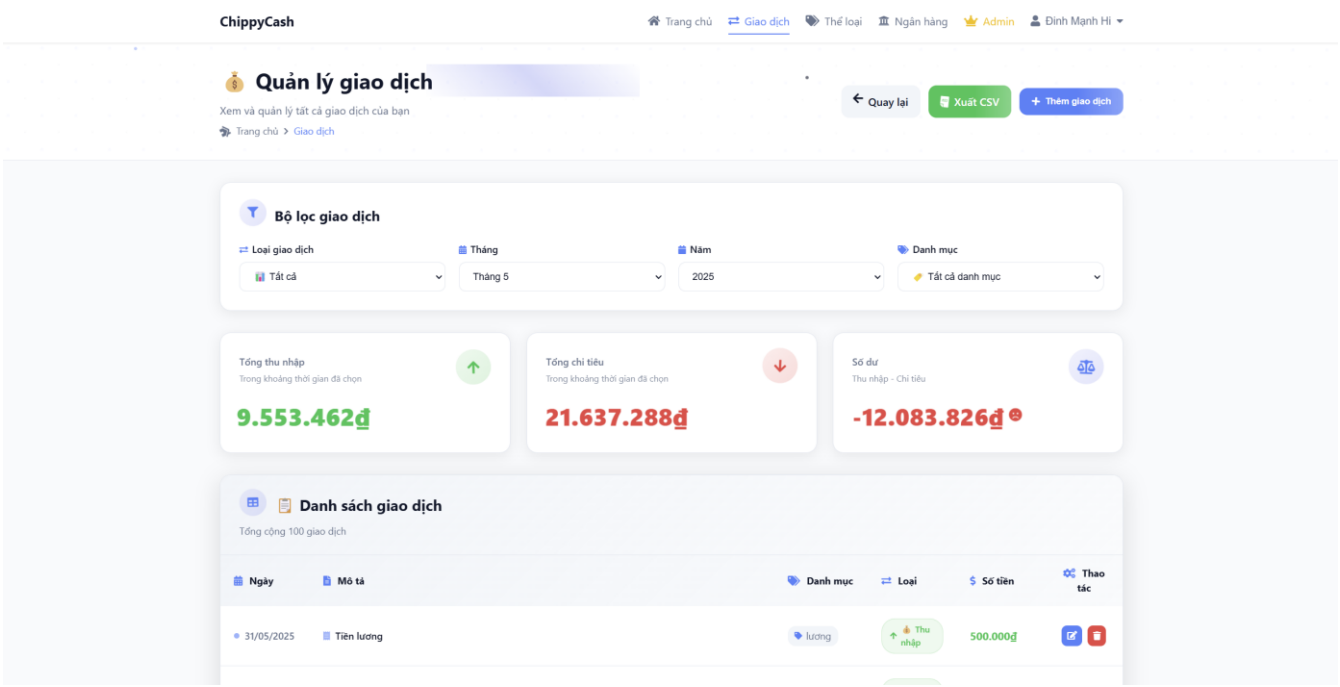
Statistics page



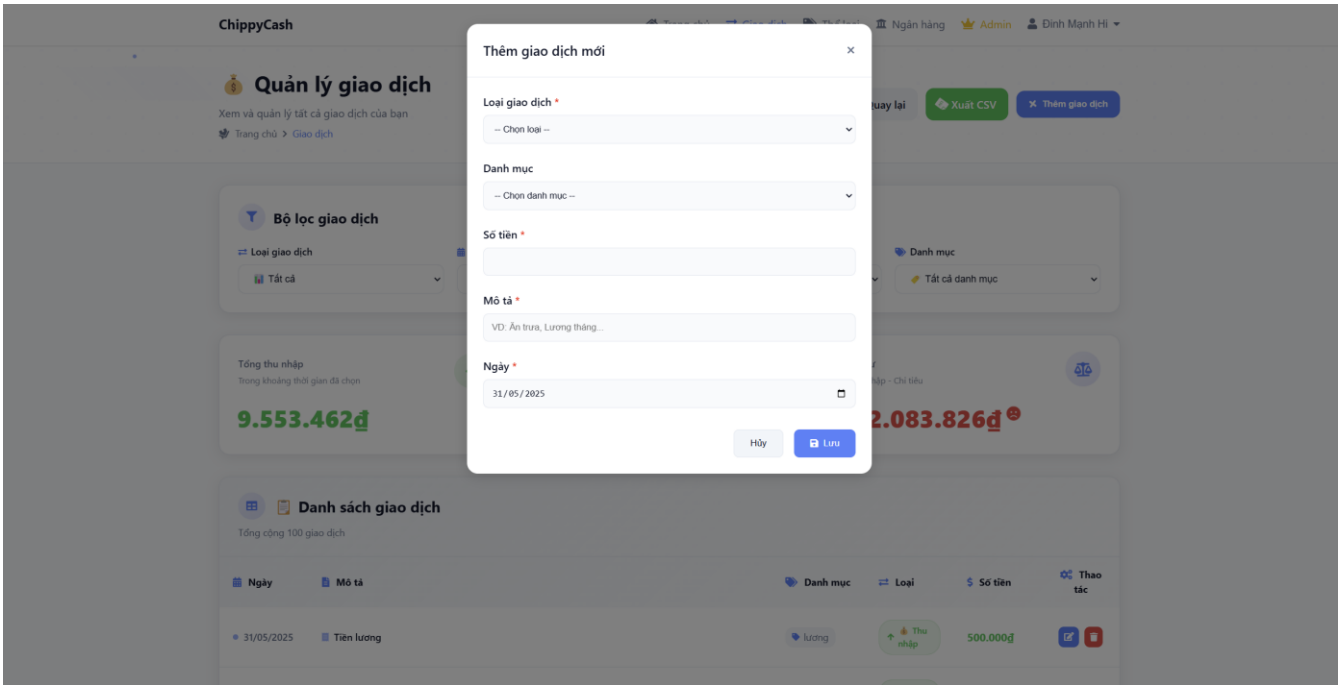




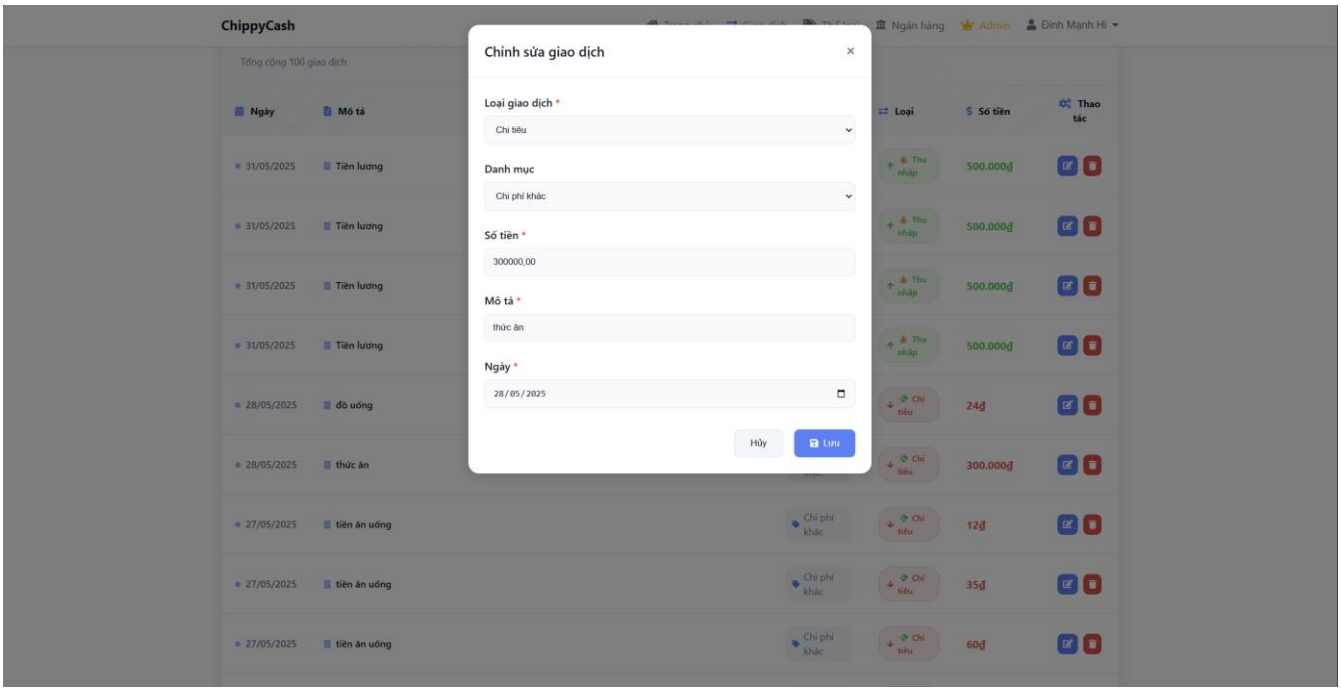
Chatbot Page



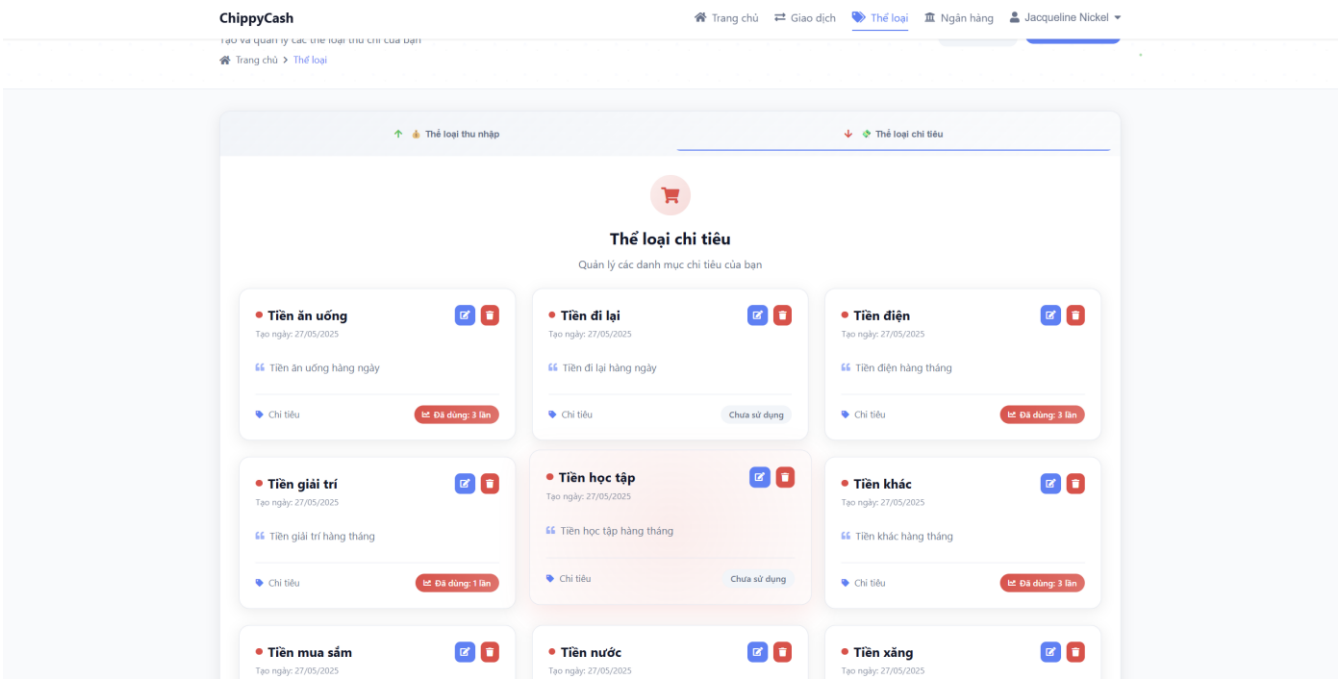
Manage Transaction Page



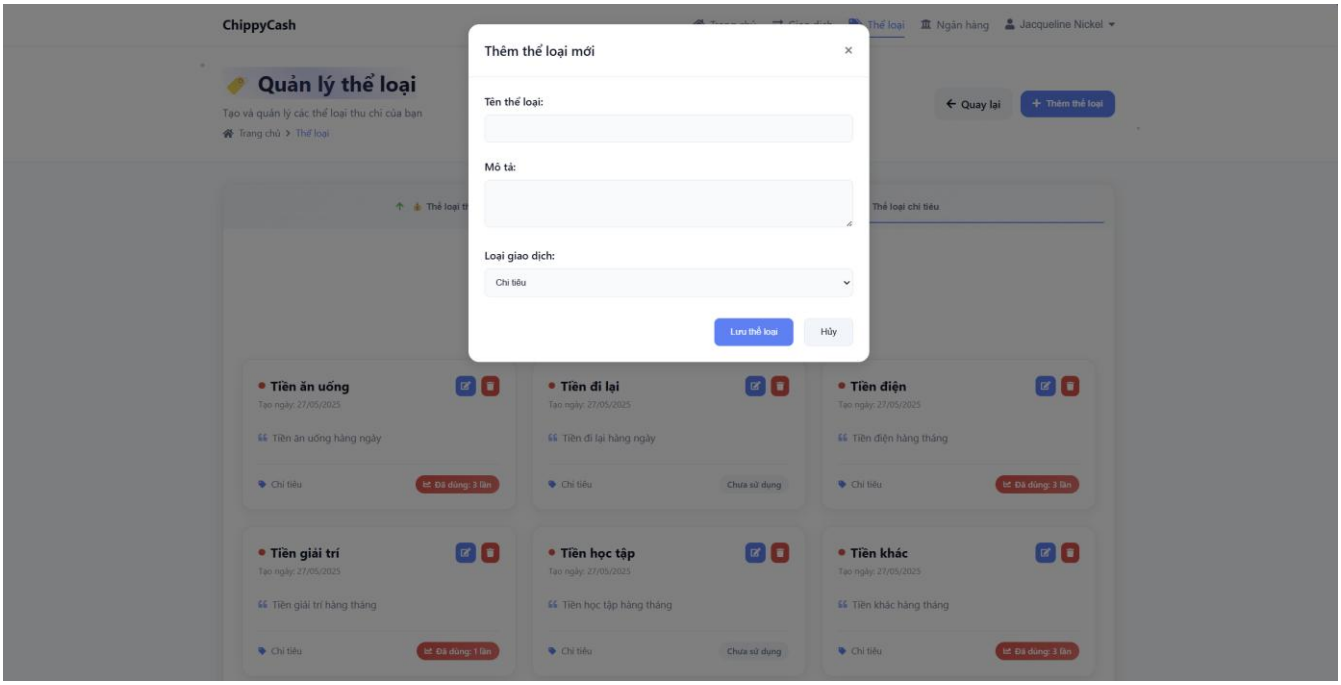
Add Transaction Page



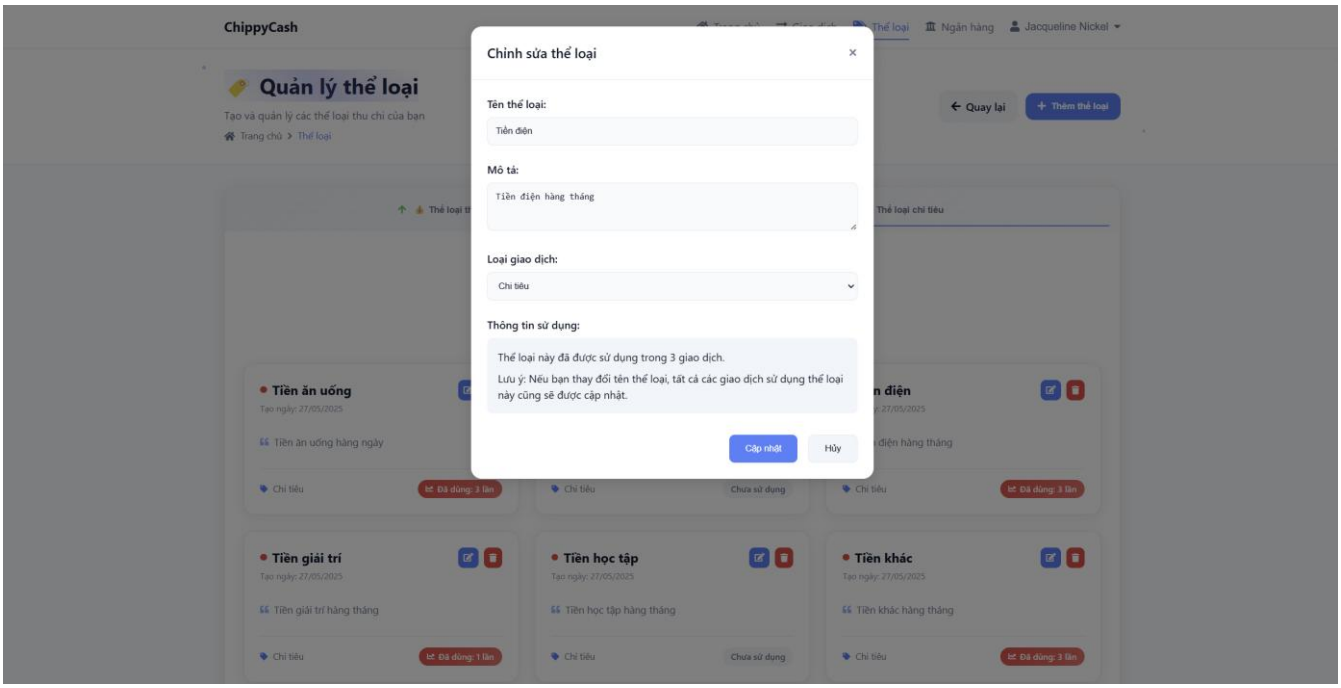
Edit Transaction Page



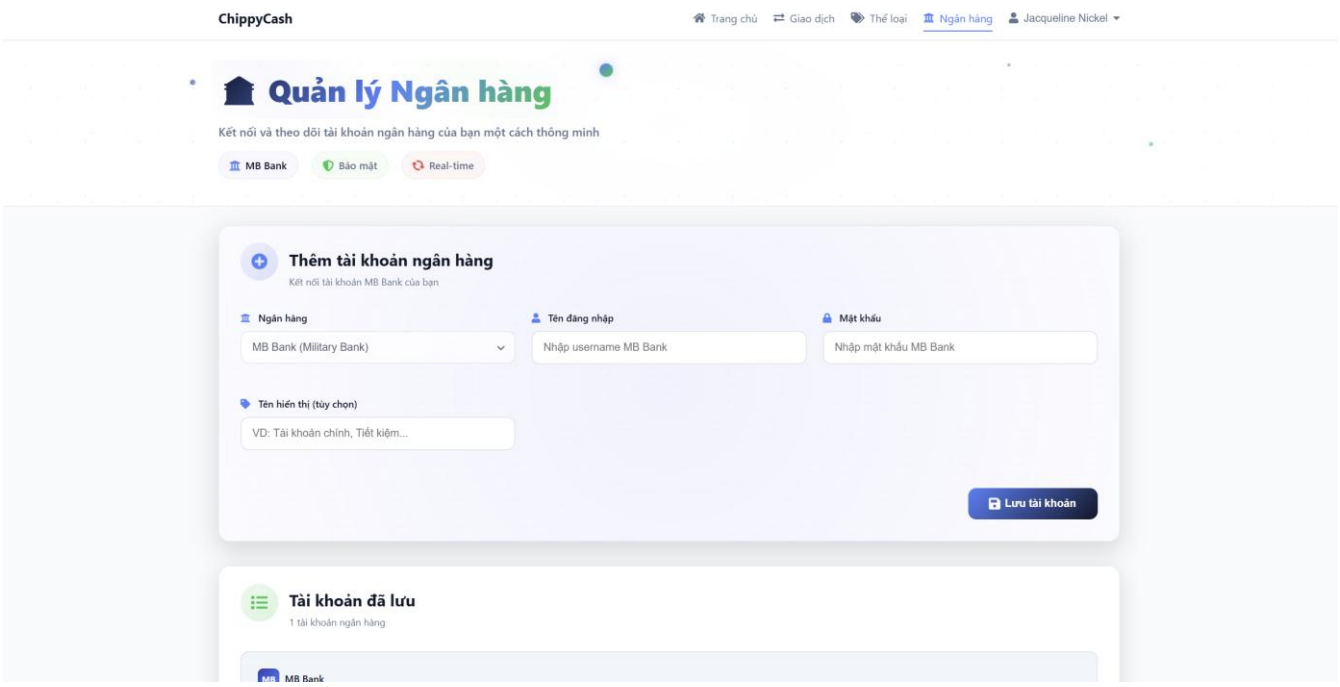
Manage Category Page



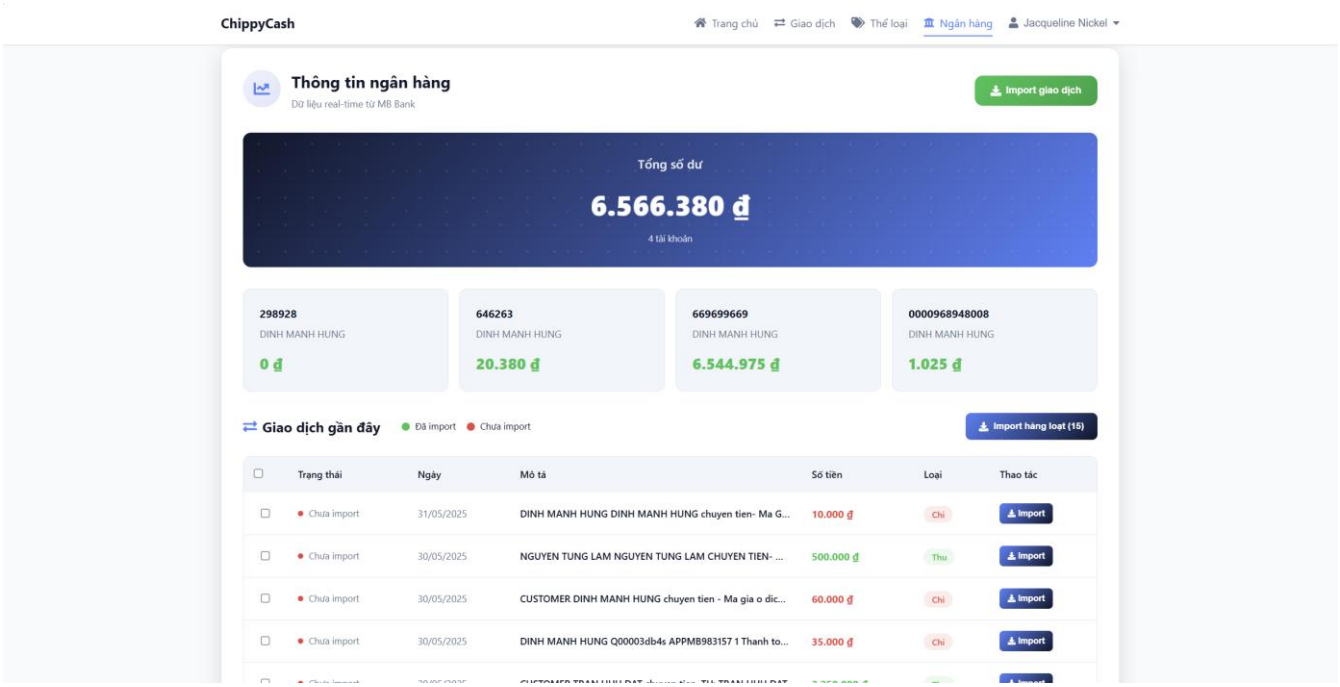
Add Category Page



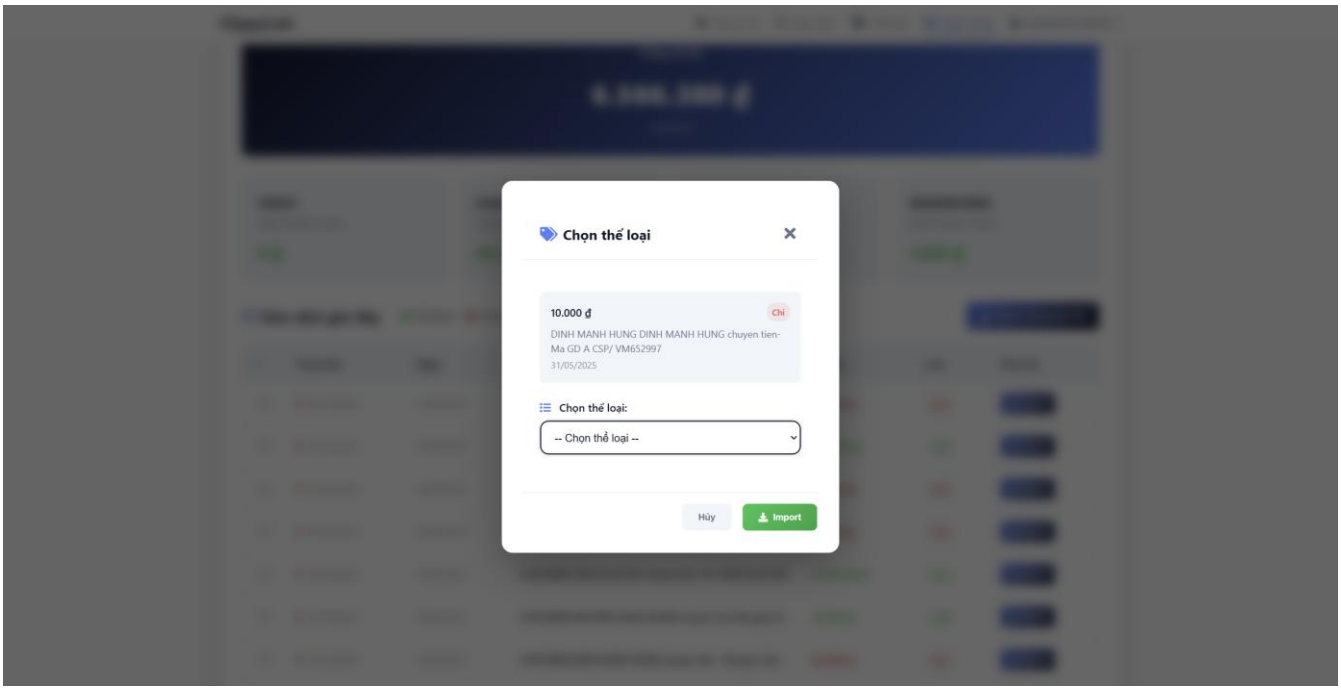
Edit Category Page



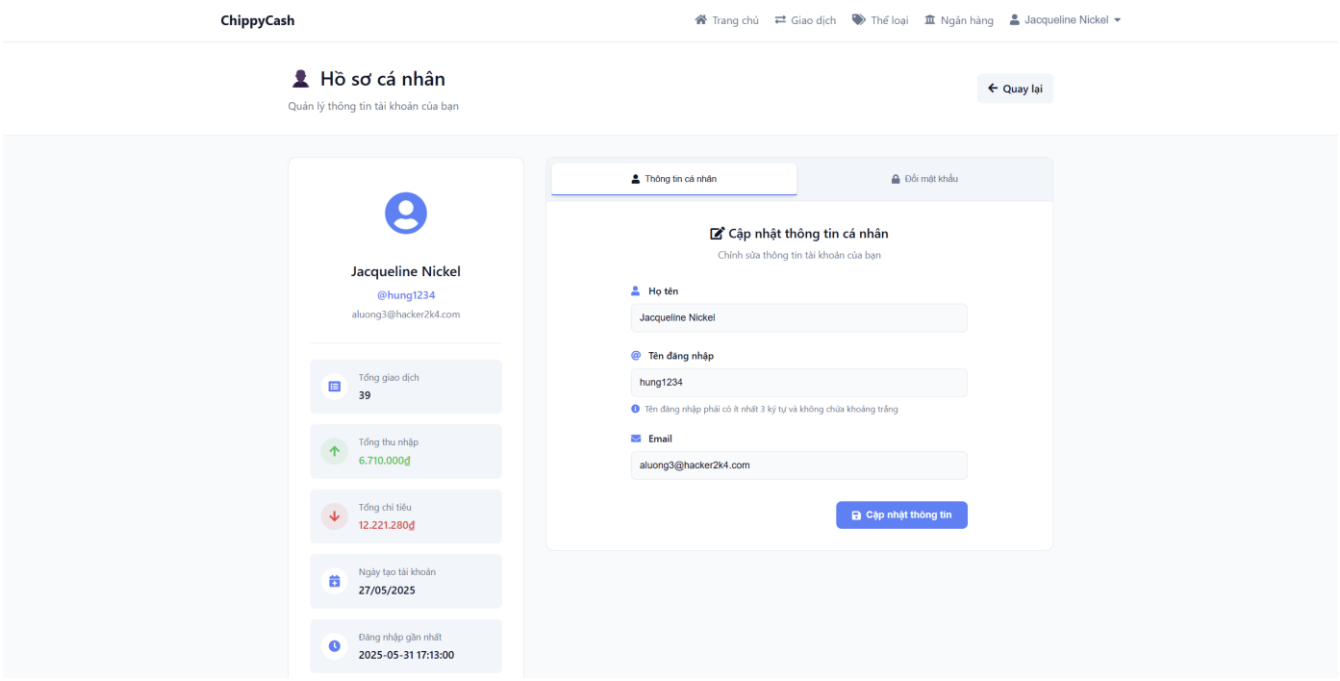
Manage Bank Page



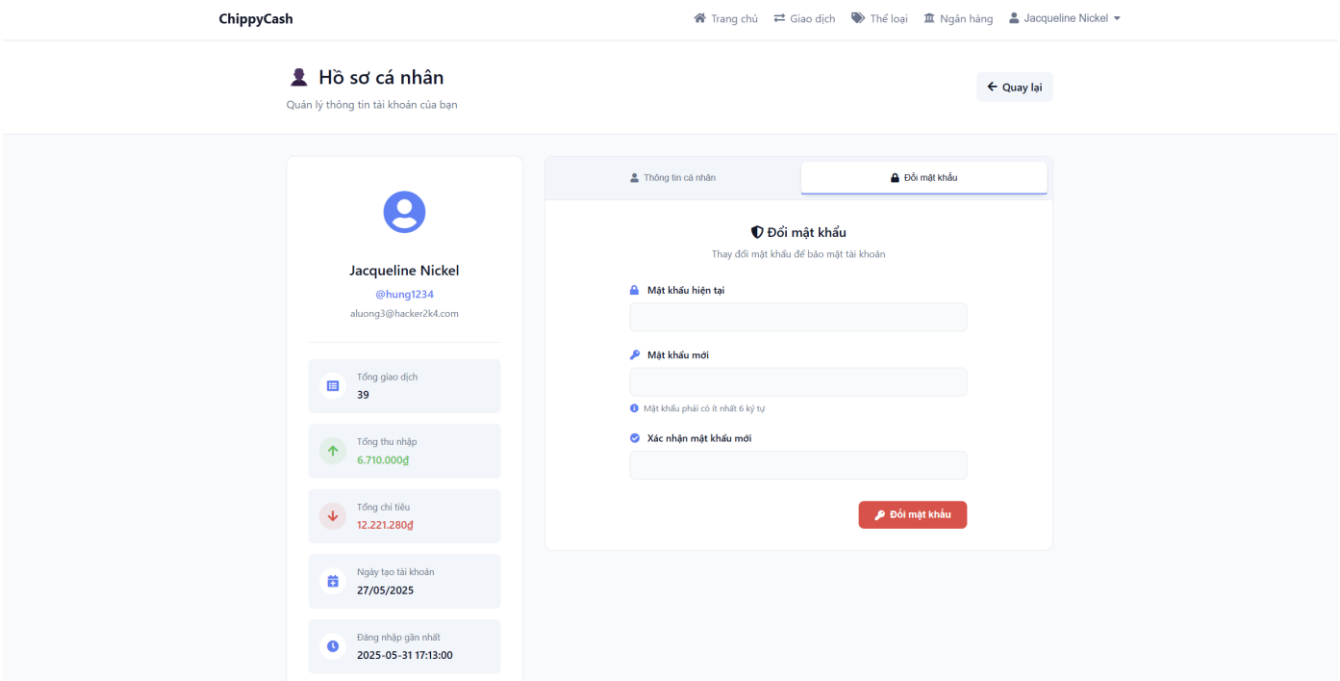
Bank Sync Page



Import Bank's Transaction Page

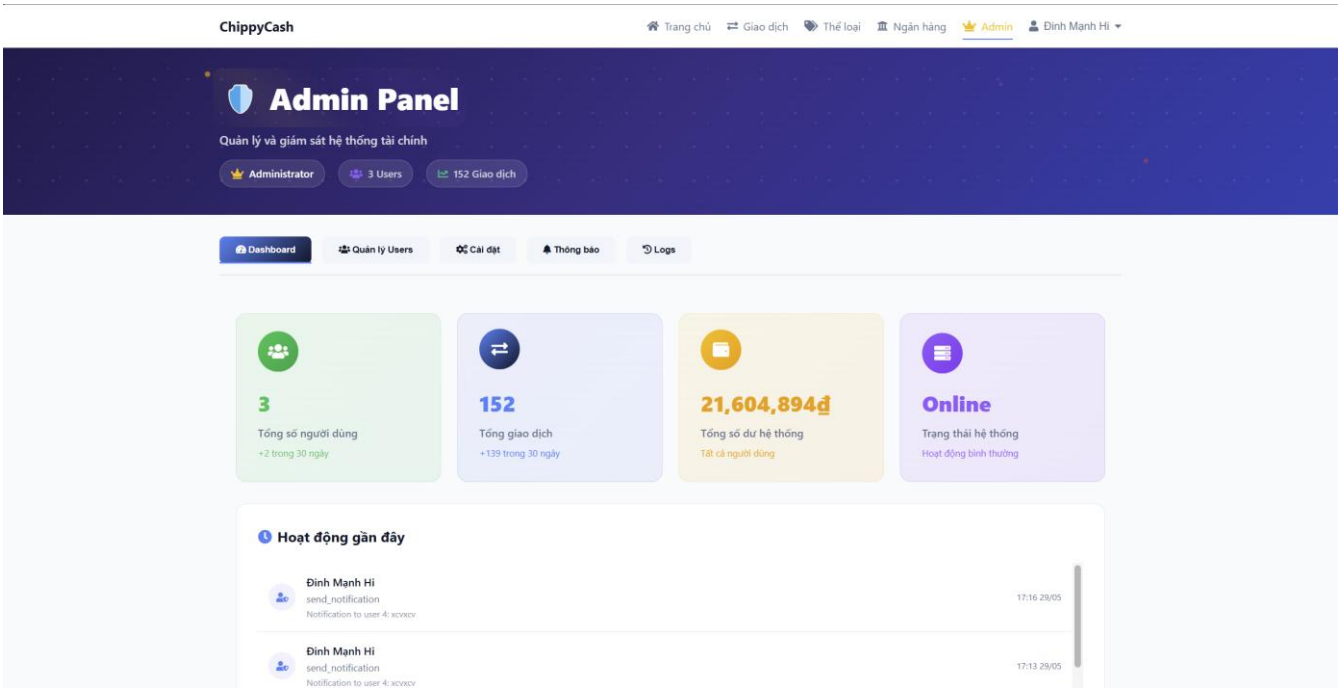


Manage Information Page

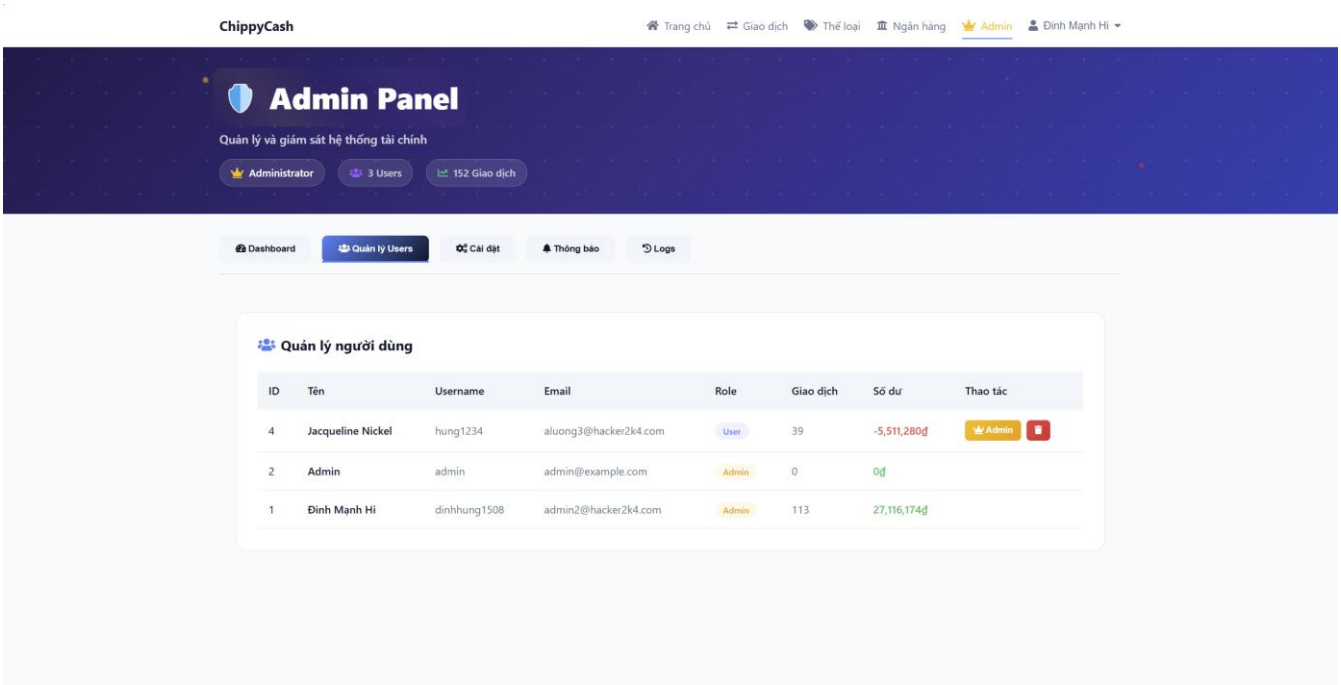


Change Password Page

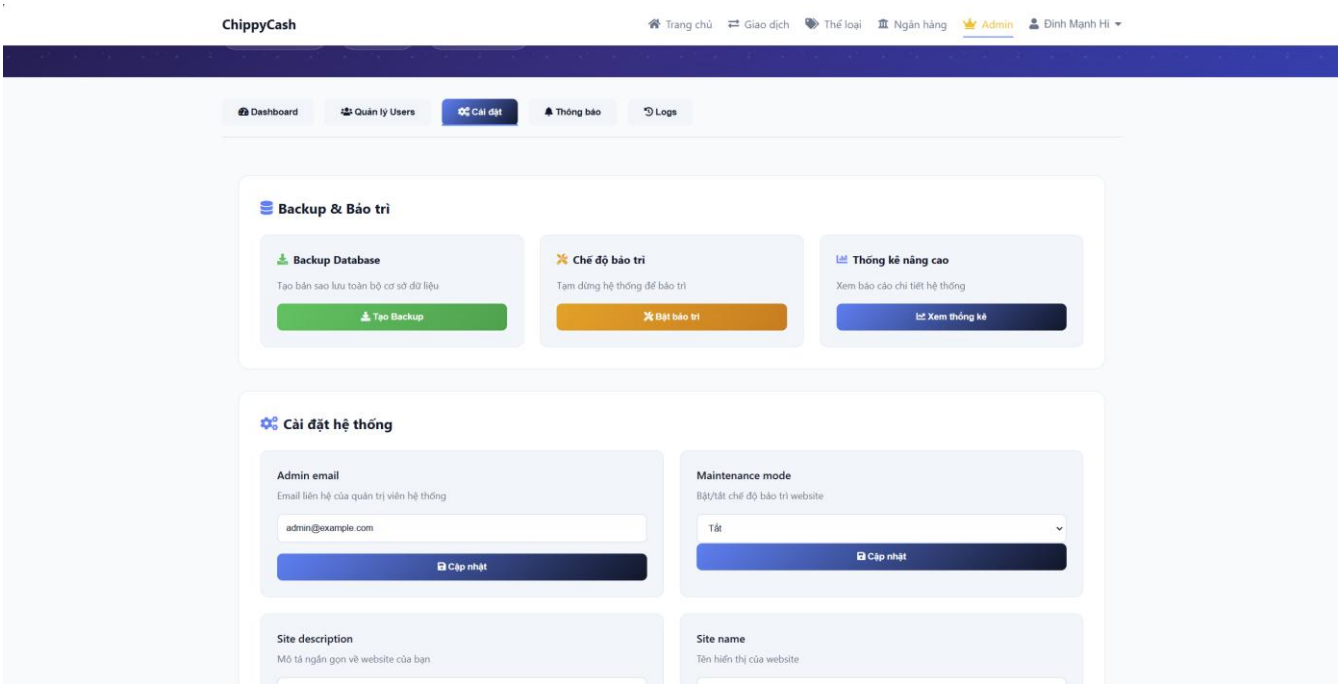
- *Admin*



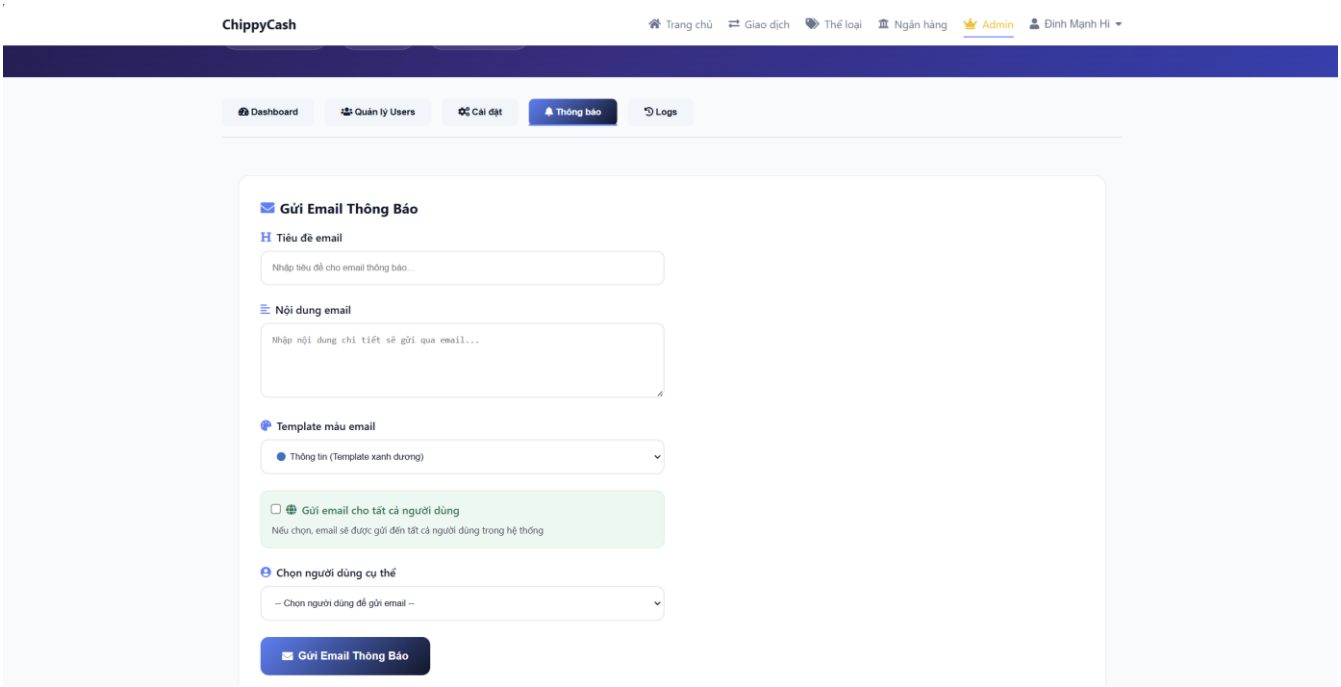
Admin Home Page



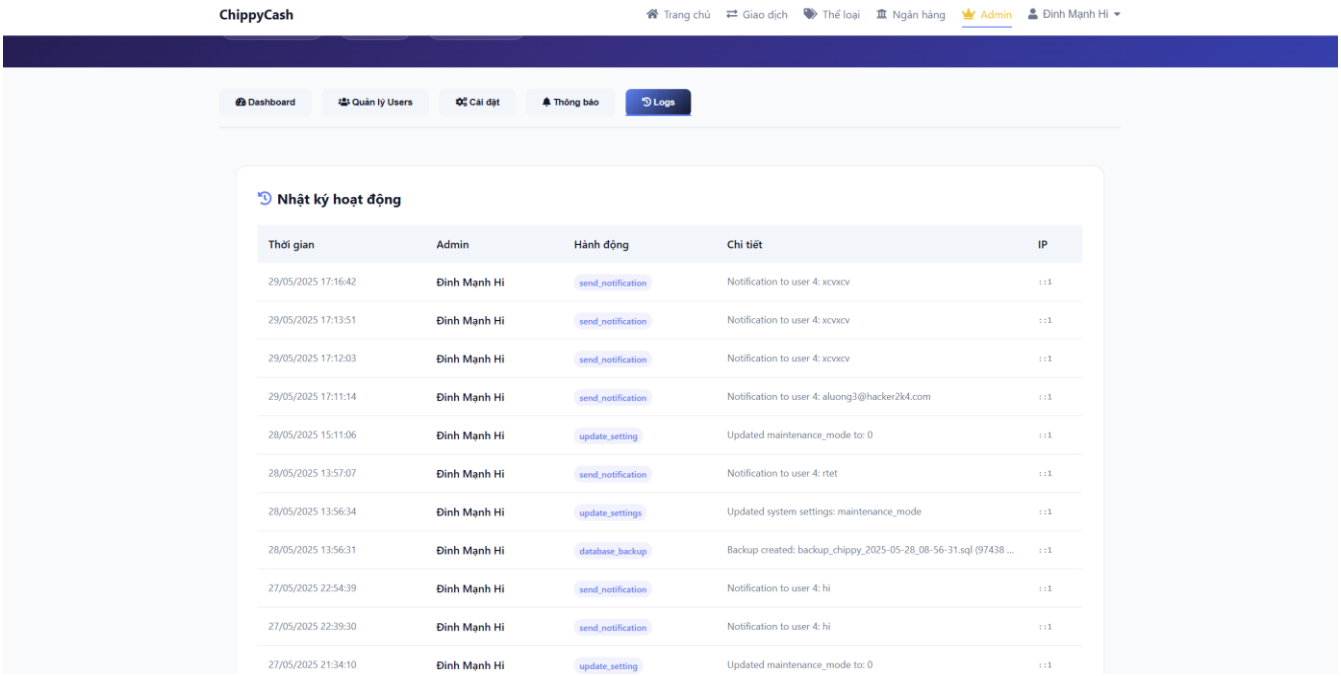
Manager User Page



Admin Config Page



Admin Notification Page



Admin Log Page

c) Testing

- Usability Testing

The website content is accurate, with no spelling or grammatical errors.

All fonts follow the required specifications.

All text is properly aligned.

All input fields are properly aligned.

All buttons are correctly formatted and sized.

Other pages are correctly linked to the homepage and vice-versa.

Links and images are not broken.

A message is displayed when an add, edit, or delete action (e.g., for transactions, categories) is performed.

These messages contain no spelling or grammatical errors.

Tabs (if any, e.g., in statistics or management pages) function correctly.

The scroll bar is displayed as required.

The ChippyCash application logo is correctly displayed on the browser's title tab.

Each web page displays the navbar consistently.

Data in dropdown lists (e.g., selecting a category when adding a transaction) is not cut off due to field size constraints.

The interface is clear, easy to understand, helping users to easily find financial management functions.

The user workflow (e.g., from login to adding a transaction, viewing reports) is intuitive and logical.

- **Functional Testing**
  - **User Authentication:**
    - Login:
      - Users can successfully log in with a valid username and password.
      - The system displays an error message for incorrect login credentials.
      - Users are redirected to the overview page after successful login.
    - Sign up:
      - New users can successfully create an account with valid information (username, password, email).
      - The system validates data (email format, password strength) and ensures username uniqueness.
      - Users are redirected to the login or overview page after successful registration.
    - Logout:
      - Users can securely log out of the system.
      - The user's session is terminated.
  - **Transaction Management (Income/Expense):**
    - Users can add new transactions with complete information (amount, description, type, category, date).

    - The system accurately stores transaction information (e.g., check save-transactions.php file functionality).
    - Users can view a list of transactions and filter by date, type, and category.
    - Users can edit and delete existing transactions.
- **Category Management:**
    - Users can create new income/expense categories.
    - Users can view, modify, and delete existing categories.
    - The system retrieves category information accurately (e.g., check get-categories.php API functionality).
- **Financial Statistics and Charts:**
    - Charts (pie, bar) accurately display spending structure by category.
    - Line charts correctly represent income/expense trends over time.
    - Statistics for total income, total expense, and balance are calculated accurately.
    - The function to export statistics to an Excel file works correctly.
- **Personal Information and Settings:**
    - Users can view and update their personal information.
    - The password change function works correctly.
    - Changes are stored correctly (e.g., check profile.php page, settings.php API).
- **Chatbot Interaction:**
    - Users can send text messages and receive responses from the chatbot.
    - The chatbot can record transactions via natural language descriptions.
    - The chatbot provides information about the financial situation when asked.
    - The chatbot provides financial advice based on user data.
    - (Advanced) The bill image upload feature works, and the chatbot can extract information.
    - Different chatbot "roles" (Smart Assistant, Grumpy Mama,

Homie) can be selected and display appropriate responses.
    - Chat history is saved and can be reloaded (e.g., check load_chat.py functionality).
    - Users can interact with the chatbot within an acceptable timeframe (no excessive delays).
- **Bank Connection (MBBank):**
    - Users can securely enter MBBank login information.
    - The system handles authentication with MBBank (including captcha solving if present).
    - Transaction history from MBBank is accurately synchronized into the ChippyCash system.
- **Administrator (Admin) Functions:**
    - Admin can log into the admin area.
    - Admin can view the list of users, edit, and delete user accounts.
    - Admin can view system statistics (total users, total transactions).
    - The database backup function operates.
    - Admin can adjust general application settings.
- **Mandatory Field Checks:**
    - All mandatory fields (e.g., username, password during registration; amount when adding a transaction) display error messages when left empty.
- **Non-mandatory Field Checks:**
    - For non-mandatory fields, no error messages are displayed when left empty.
- **Numeric Field Checks:**
    - For numeric fields (e.g., transaction amount), non-numeric characters are not accepted, and an appropriate error message is shown.
    - User information, transactions, and categories are fully and accurately stored in the database.

    - Users are only allowed to perform certain functions after successful login.

- **Compatibility Testing**
    - The website functions and displays correctly across different browsers (e.g., Chrome, Firefox, Safari, Edge).
    - Images are displayed correctly across different browsers.
    - Fonts are displayed correctly in different browsers.
    - JavaScript code works properly across different browsers.
    - Test the website's responsiveness on various screen sizes (desktop, tablet, mobile), ensuring the interface does not break or become difficult to use. (Related to the use of Bootstrap)

3. Project Schedule Management

Task Name	Started on	Due on	Progress
Initiation Phase			
Define project scope and	10/03/2025	13/03/2025	100%
Requirement analysis	14/03/2025	20/03/2025	100%
Planning Phase			
Develop a project schedule	21/03/2025	22/03/2025	100%
Design Phase			
User experience (UX) design	23/03/2025	25/03/2025	100%
User interface (UI) design	26/03/2025	28/03/2025	100%
Development Phase			
API development	28/03/2025	15/04/2025	98%
Front-end implementation	04/04/2025	19/04/2024	100%
Chatbot development	20/04/2025	01/05/2025	93%
Integration of components	02/05/2025	10/05/2025	100%
Testing Phase			
Usability Testing	06/05/2025	20/05/2025	100%
Functional Testing	11/05/2025	27/05/2025	100%
Compatibility Testing	11/05/2025	30/05/2025	100%

4. Results of the project

- **Completed Features:**
    - User Login, Registration, and Logout.
    - View personal financial overview homepage.
    - Transaction Management: Add, view, edit, and delete income/expense transactions.
    - Income/Expense Category Management.
    - Chatbot interaction for:
        - Quick transaction recording via chat (text).
        - Basic financial information queries.
        - Information recognition from bill images (basic OCR).
        - Receiving responses with voice (text-to-speech).
    - View financial reports and statistics (structural charts, trends).
    - Save and review chat history with the chatbot.
    - Connect bank account (MBBank) to view balance and import transactions.
    - Personal information management.
    - Basic Admin functionalities: User management, view general statistics.
- **Features Not Implemented:**
    - In-depth personalized financial advice and smart suggestions based on advanced analysis of spending data.
    - Detailed budget management, setting financial goals, and progress tracking.
    - Fully automated and robust bank transaction synchronization with advanced error handling.
    - More flexible customization of financial reports and charts.
    - Some advanced Admin features like detailed system configuration management and data recovery.
- **Project Completion Level:** Approximately 90% (Most core functionalities

have been completed; some advanced features or supplementary utilities require further development)

IV. Conclusion

1. Advantages:

- The web application has completed most of the core personal financial management features as required.
- The website layout is suitable, easy to read, and user-friendly, helping users to easily operate and monitor their financial situation.
- Features are easy to use and familiar to users, especially the integration of a chatbot for quick data entry and queries.

2. Disadvantages:

- The application is still under development, so some advanced financial management features or auxiliary utilities are still basic or incomplete.
- Some features outlined in the detailed plan (e.g., in-depth financial analysis, automated budget management) are still missing or need further improvement.

3. Development Direction:

- Add missing features, expand the web application's system functionalities, especially smarter and more personalized financial analysis tools.
- Enhance the accuracy of OCR for bills, improve bank synchronization capabilities, and ensure system stability and security.
- Research and develop additional utility features such as detailed budget planning, savings goal tracking, and proactive financial alerts and recommendations for users.

4. Summary:

During the process of learning and building the project, I have tried to learn and develop many features to create a useful personal financial management tool.

However, the project still has many shortcomings and areas for

improvement. In the future, I will continue to research and learn more to complete this web application system.

I sincerely thank my instructor for the guidance and support in completing this project.

V. References

[1] Welling, L., & Thomson, L. (2008). *PHP and MySQL Web Development* (4th ed.). Addison-Wesley Professional. (Assumed general reference for PHP/MySQL development)

[2] Mozilla Developer Network. *HTML: HyperText Markup Language*, from https://developer.mozilla.org/en-US/docs/Web/HTML

[3] Bootstrap. *Bootstrap Documentation*. Retrieved May 31, 2025, from https://getbootstrap.com/docs/5.3/getting-started/introduction/

[4] MySQL. *MySQL Documentation*, from https://dev.mysql.com/doc/

[5] Ramírez, S. *FastAPI Documentation*, from https://fastapi.tiangolo.com/

[6] OpenAI. *OpenAI API Documentation*, from https://platform.openai.com/docs (General reference for OpenAI API, as GPT-4 might be too specific if other models are used)

[7] Liu, J. (2022). *LlamaIndex (GPT Index) Documentation*, from https://gpt-index.readthedocs.io/en/latest/ (Adjusted to a more general LlamaIndex reference)

[8] TensorFlow. *TensorFlow OCR*, from https://www.tensorflow.org/hub/tutorials/tfhub_object_detection_tf2 (Example of a general OCR technology; replace if a specific library like Tesseract OCR or a cloud service API was used)..