RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

PERSONAL FINANCE MANAGEMENT

A MINI PROJECT REPORT

SUBMITTED BY

ANANTHA KHRISHNAN N	231501013
ANTO ASHIK U H	231501016
ANTONY JADRIEN A	231501017
EDMOND ALLAN A	231501044

In Partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024-2025

BONAFIDE CERTIFICATE

Certified that this project report **“PERSONAL FINANCE MANAGEMENT**” is the bonafide work of **“ANANTHA KHRISHNAN(231501013), ANTO ASHIK (231501016), ANTONY JADRIEN(231501017), EDMOND ALLAN(231501017)”**who carried out the project work under my supervision.

Submitted for the Practical Examination held on

SIGNATURE

Mr. U. Kumaran,

Assistant Professor

AIML,

Rajalakshmi Engineering College,

(Autonomous),

Thandalam, Chennai - 602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The Personal Finance Management System is a comprehensive platform designed to simplify and optimize the management of personal finances by providing users with tailored access to key features. The system allows users to track their income, expenses, savings, and investments, offering insights to improve financial decision-making. By categorizing transactions and generating reports, it empowers users to manage their budget effectively and plan for future financial goals. The system's user-friendly interface enhances accessibility, and its secure access model ensures that each individual’s financial data is protected. The Personal Finance Management System is designed to enhance financial literacy, improve budgeting efficiency, and promote smarter financial decisions in everyday life.

TABLE OF CONTENTS

1.1 INTRODUCTION

Introduction to Personal Finance Management

Personal finance management involves the process of budgeting, saving, investing, and overseeing all financial aspects of an individual's life. It helps individuals make informed decisions about their money, ensuring they can meet both short-term and long-term financial goals. Effective personal finance management includes tracking income, controlling spending, planning for retirement, and managing debt. With the rise of digital tools, individuals can now access easy-to-use systems that help them organize their finances, monitor expenses, and create detailed financial plans. The goal is to achieve financial stability and security by making proactive financial decisions. Ultimately, mastering personal finance management empowers individuals to take control of their financial future.

Importance of Finance Management in Markets

Finance management is vital for the growth and stability of markets, helping businesses and investors allocate resources efficiently, maximize profits, and minimize risks.For individuals, it leads to better portfolio diversification, risk management, and alignment with financial goals, driving economic growth and wealth creation.

i. **Resource Allocation**: Efficient finance management ensures optimal allocation of resources, which is essential for maximizing profitability and minimizing waste in both businesses and investments Inventory Control.
ii. **Risk Management**: By carefully monitoring financial activities, businesses and individuals can identify and mitigate financial risks, protecting against market volatility and unforeseen event.
iii. **Market Stability**: Strong finance management practices contribute to overall market stability by ensuring businesses can withstand economic downturns and continue to operate efficiently.

Purpose of the Project

With a comprehensive and user-friendly platform to effectively manage their personal finances. This system aims to help users track their income, expenses, savings, and investments, enabling them to make informed financial decisions. By offering features such as budgeting The purpose of the Personal Finance Management project is to provide individuals tools, expense categorization, and financial reporting, the project seeks to improve financial literacy and promote better financial planning. Ultimately, the goal is to empower users to achieve financial stability, meet their financial goals, and enhance their overall financial well-being.

The purpose of the Personal Finance Management project is to help individuals effectively manage their finances by providing a user-friendly platform for tracking income, expenses, and savings. It aims to improve financial decision-making through budgeting tools and personalized insights. The system promotes savings, investment goals, and debt management, empowering users to achieve long-term financial stability. By consolidating financial data from multiple accounts, the project offers a comprehensive view of personal finances. It also enhances financial literacy, helping users make informed financial choices. Ultimately, the goal is to provide individuals with the tools to secure their financial future and reduce financial stress.

Scope of the Project

The scope of the Personal Finance Management project covers a wide range of features and functionalities aimed at helping individuals manage their personal finances more effectively. The key components of the project include:

1. **Income and Expense Tracking:**

   The system will support multiple user roles, including admin, cashier, and stock manager, each with specific access rights and functionalities. This role-based access ensures data security and efficient management of operations.

2. **Budgeting Tools**:

   The system allows users to create personalized budgets, set financial goals, and track progress over time, ensuring better control over their finances.

3. **Sales Management**:

   The HMS will enable cashiers to process sales transactions swiftly. Features will include scanning barcodes, managing shopping carts, applying discounts, and generating invoices. This will enhance the checkout experience for customers.

4. **Debt Management**:

   Users can track outstanding debts, manage repayment schedules, and set goals for reducing liabilities, improving their financial health.

5. **Saving and Investment management**:

   The platform supports users in setting up savings goals and tracking investments, helping them plan for future financial milestones such as retirement or major purchases.

6. **Financial Goal Setting:**

   The project generates detailed financial reports and analytics to give users insights into their financial health, income-expense ratios, and areas for improvement.

7. **Multi-Account Integration**:

   Users can link and manage multiple bank accounts, credit cards, and investment portfolios within a single platform, providing a comprehensive view of their finances.

Need for the Project

The need for the Personal Finance Management project arises from the growing complexity of managing personal finances in today's fast-paced world. With increasing financial responsibilities, such as budgeting, saving for retirement, managing debt, and investing, individuals often struggle to maintain control over their finances. Many people lack the tools or knowledge to make informed financial decisions, leading to poor financial habits and, ultimately, financial insecurity.

Gaps in Financial Management: Many individuals struggle with managing their finances due to a lack of tools or knowledge, leading to poor financial habits and insecurity. This project addresses that gap by providing a comprehensive platform for effective finance management.

Budgeting and Goal Tracking: The platform helps users create and maintain budgets, track income and expenses, and monitor progress toward financial goals, empowering better decision-making and financial discipline.

Personalized Insights and Consolidation: By consolidating financial information from multiple accounts in one place, the system provides users with a clear view of their finances and personalized recommendations to improve financial health.

Proactive Financial Management: The project includes built-in reminders and alerts, ensuring users stay on track with bill payments, budget limits, and financial goals, preventing missed deadlines or overspending.

Financial Literacy Education: The system provides resources to improve financial literacy, helping users understand key financial concepts such as budgeting, saving, and investing.

1.2 OBJECTIVES

The objectives of your project (Personal Finance Management System) based on your focus on streamlining financial management and empowering users with the right tools to improve their financial well-being:

1. Streamline Financial Management

- Process Automation: Automating repetitive financial tasks such as bill payments, expense tracking, and income categorization to improve efficiency.
- Centralized Financial Dashboard: Integrating all financial data (bank accounts, investments, expenses) into a single dashboard for easy access and management.
- Task Management Tools: Implementing tools for setting financial goals, creating budgets, and tracking progress to ensure accountability in achieving personal financial objectives.

2. Enhance Budgeting and Expense Tracking

- Real-time Expense Tracking: Automatically tracking and categorizing expenses in real time, providing users with an accurate picture of their spending habits.
- Customizable Budgeting Tools: Allowing users to create personalized budgets, set spending limits, and receive alerts when they approach budget thresholds.
- Expense Reports: Generating detailed monthly or yearly expense reports to help users identify areas for cost-saving and better financial planning.

3. Facilitate Investment and Savings Management

- Investment Tracking: Providing tools for users to track their investment portfolios, including stocks, bonds, mutual funds, and retirement accounts.
- Automated Savings Plans: Setting up automated savings and investment contributions based on the user’s financial goals (e.g., emergency fund, retirement).
- Goal-Oriented Savings: Enabling users to create specific savings goals (e.g., home purchase, education) and track progress toward meeting those goals.

4. Improve Debt Management and Credit Monitoring

- Debt Tracker: Helping users manage and track outstanding debts, including credit cards, loans, and mortgages, with tools for setting up payment schedules.
- Credit Score Monitoring: Providing users with regular updates on their credit score and offering tips on improving it by managing credit responsibly.
- Debt Reduction Strategies: Offering strategies such as debt snowball or debt avalanche methods to help users pay off their debts faster.

5. Ensure Data Accuracy and Security

- Data Validation: Implementing validation checks to ensure accurate data entry and prevent errors in financial tracking and reporting.
- Encryption and Security Protocols: Using encryption, multi-factor authentication, and access controls to protect users' financial data and personal information.
- Regular Backups: Ensuring that all financial data is regularly backed up and protected from potential data loss due to technical failures.

6. Enhance User Experience

- User-Friendly Interface: Designing an intuitive and easy-to-navigate interface that simplifies financial tracking, goal-setting, and reporting for all users.
- Personalized Dashboards: Providing users with customizable dashboards that display the most relevant financial information, such as account balances, budgets, and goals.

- Training and Support: Offering resources, tutorials, and customer support to ensure users can effectively utilize all features of the system.

7. Provide Real-Time Updates and Notifications

- Alerts and Notifications: Sending notifications for important financial events such as bill payments, upcoming due dates, low balances, or unusual spending activity.
- Mobile Accessibility: Ensuring users can access their financial data and receive updates through mobile devices for convenience and timely action.
- Automated Reminders: Sending reminders for upcoming payments, savings goals, and other important financial tasks to keep users on track.

8. Support Financial Planning and Forecasting

- Financial Planning Tools: Providing users with tools to create long-term financial plans, including retirement planning, major purchases, and educational expenses.
- Forecasting and Projections: Using financial algorithms to project future savings, investment growth, and cash flow based on current habits and goals.
- Scenario Analysis: Allowing users to simulate different financial scenarios (e.g., changing income, adjusting savings) to understand potential outcomes.

9. Ensure Scalability and Flexibility

- Modular System Design: Designing the system with a flexible, modular approach to allow for the addition of new features, such as advanced investment tracking or tax filing, as needed.
- Performance Monitoring: Continuously monitoring the system's performance to ensure fast and reliable operation, even as users' financial data grows over time.
- Adaptability to Financial Trends: Keeping the system up-to-date with changes in financial regulations, new financial products, and emerging technologies in personal finance management.

1.2 MODULES

1. Login Module

- Overview: The login module is the entry point for users, ensuring secure access to the system.
- Functionality: **User Authentication**: Requires users to enter their credentials to access the site.
- **Error Handling**: Provides feedback for incorrect login attempts and allows for password recovery if necessary.

2. Home Dashboard Module

- Overview: The home dashboard provides users with a summary of their financial information.
- Functionality:
  - **Balance Overview**: Displays the current balance and summarizes monthly income and expenses.
  - **Transaction History**: Shows recent transactions, categorized by income and expense, for easy reference.

3. Income Management Module

- Overview: This module is tailored for cashiers, streamlining the sales process and enhancing customer interaction at the point of sale.
- Functionality:
- **Income Entry**: Allows users to add details about income, such as source, amount, and date.
- **Edit/Delete Income**: Users can manage their income records by editing or deleting entries as needed.

4. Expense Management Module

- Overview: Allows users to record and categorize their expenses for better financial tracking.
- Functionality:
  - **Expense Entry**: Users can add expenses, specifying category, amount, date, and account name.
  - **Edit/Delete Expenses**: Users can update or remove expense entries for accurate records.

5. Category Management Module

- Overview: Aids users in organizing their finances by categorizing income and expenses.
- Functionality:
  - **Category Customization**: Users can select or customize categories for both income and expenses, supporting personalized tracking.
  - **Spending Analysis**: Offers insights into spending by category to help users understand spending habits.

6. Monthly Summary Module

- Overview: Provides a summary of income and expenses over the month.
- Functionality:
  - **Monthly Balance Calculation**: Shows total income and expenses, calculating the month's balance.
  - **Monthly Reports**: Users can view monthly summaries for budgeting purposes.

7. User Management Module

- Overview: Allows users to manage their account details and maintain secure access. Functionality:
  - **Profile Management**: Users can update personal information, such as email and password.
  - **Account Security**: Users have the option to log out and change their password to maintain security.

8.Database Management Module

Overview: Manages data storage and retrieval, ensuring data integrity across all forms.

Functionality:

SQL Integration: Data entered through income and expense forms is stored securely in an SQL database.

Data Retrieval: Ensures efficient data retrieval for accurate display on the home dashboard and transaction history.

9. Frontend and Styling Module

- Overview: Manages the look and feel of the website for a seamless user experience.

Functionality:

Responsive Design: Uses CSS and JavaScript for layout and styling, optimizing display for various screen sizes.

Interactive Elements: JavaScript provides feedback and interaction on buttons and form elements, enhancing user experience.

2. SURVEY OF TECHNOLOGIES

Survey of Technologies for the Personal Finance Management System

The development of a personal finance management website requires integrating various technologies to ensure a smooth user experience, reliable data storage, and scalability. Below is an overview of the key technologies used in designing and implementing the system.

1.Programming Language: PHP

- Overview: PHP is a widely-used, open-source scripting language especially suited for web development and can be embedded directly into HTML.
- Benefits:

    - Server-Side Processing: PHP efficiently handles server-side tasks, such as data processing and form handling.

    - Community Support: PHP has extensive community support and resources, making it easy to find libraries and troubleshoot.
    - Integration Capabilities: Works well with various databases, making it ideal for data-driven applications.

2. Frontend Technologies: HTML, CSS, JavaScript

- Overview: HTML, CSS, and JavaScript form the foundation of front-end development, enabling structure, styling, and interactivity for web applications.
- Benefits:
    - Responsive Design: CSS provides styling flexibility, allowing for responsive designs across devices.
    - Interactivity: JavaScript enables dynamic interactions and enhances user experience.

    - Customization: Offers complete control over the layout, design, and interactivity to meet project requirements.

3. Database Management System: MySQL

- Overview: MySQL is a reliable, scalable RDBMS integrated within XAMPP for managing application data locally during development.
- Benefits:
    - Data Integrity: MySQL supports robust data management, essential for handling financial records accurately.
    - Efficiency: As the hypermarket grows, MySQL can handle increased data loads efficiently.
    - Community Support and Documentation: Extensive documentation and community support facilitate troubleshooting and development.

4. Web Server: Apache (via XAMPP)

- Overview: Apache, part of the XAMPP stack, is an open-source web server that hosts web applications locally and aids in testing during development.
- Benefits:
    - Local Development: These tools help in generating graphical representations of sales data, customer behavior, and inventory statistics.
    - Flexibility and Configuration: Offers customization options to tailor visualizations to specific requirements.

    - Compatibility with PHP and MySQL: Works seamlessly with PHP and MySQL for efficient web hosting.

5. Version Control: Git

- Overview: Git is a version control system that helps manage changes to the project codebase, enabling collaboration among developers.
- Benefits:
    - Change Tracking: Tracks all changes in the code, providing a history and enabling easy reversion if necessary.
    - Branching and Merging: Supports branching to work on new features without impacting the main codebase.

- Collaboration: Facilitates collaboration among team members, allowing multiple people to work on the project.

2.1 SOFTWARE DESCRIPTION

The Personal Finance Management Website is a comprehensive tool designed to help users effectively manage their finances. It provides a simple interface for tracking income, expenses, and budgeting. Below is a detailed description of the software components and functionalities:

1.User Roles and Access Control

- **User Role (General):**
  - Access to a personal finance dashboard that displays balance, recent transactions, and monthly income/expenses.
  - Ability to add income and expenses, categorize transactions, and view financial summaries.
  - Access to reports on spending patterns, savings progress, and monthly summaries.

2.Core Functionalities

- **Budgeting and Expense Management:**
  - Tools for tracking expenses by category (e.g., food, transportation, health) monthly spending.
  - Monthly summaries showing income and expenses to aid in budgeting.
  - Ability to view recent transactions for easy tracking of spending habits.
- **Income and Expense Tracking**:
  - Simple forms for adding income from various sources and logging expenses.

- SQL-based storage of all transaction data, ensuring reliable record-keeping.
- Monthly balance calculation, showing users a quick snapshot of their financial health.

- **Category Management:**
  - Customizable categories for both income and expenses, supporting personalized tracking.
  - Expense categorization for insight into spending habits, helping users identify areas to save.

3.User Interface Design:

- **Graphical User Interface (GUI)**:

- Developed with HTML, CSS, and JavaScript to ensure a responsive and user-friendly experience.
- Clear and intuitive design with a responsive layout for different screen sizes.
- Simple navigation paths that allow easy access to income, expense forms, and balance summary.

4. Database Management

- **Database System::**
  - Utilizes MySQL for secure and reliable data storage within the XAMPP stack.
  - Structured database with tables for income, expenses, and user details.
  - Efficient SQL queries to retrieve and display information on the home dashboard quickly.

5. Reporting and Analytics

- **Database Summary:**
  - Displays summaries on the homepage, including current balance, monthly income, and expense totals.
  - Monthly summary provides a clear overview of income vs. expenses to aid budgeting.

    - Simple transaction history showing recent income and expenses for easy tracking.

6.Security Features

- **Access Control:**
    - Login and signup functionality to ensure only authenticated users can access financial data..
    - Secure password storage and authentication to protect user accounts and data.

2.2 LANGUAGES

Languages Used in the Personal Finance Management System

Frontend Technologies

1. HTML, CSS & JavaScript

Overview: HTML, CSS, and JavaScript are the foundational technologies for web development. HTML provides the structure and content of a web page, CSS styles and arranges those elements for a visually appealing layout, and JavaScript adds interactivity and dynamic functionality, making the website responsive to user actions. Together, they enable the creation of modern, user-friendly, and interactive websites.

Usage: In the Personal Finance Management Website, HTML is used to structure the content, such as forms, tables, and navigation menus. CSS styles the layout, ensuring a responsive and visually appealing design across devices. JavaScript adds interactivity by validating user input, updating content dynamically, and visualizing financial data through charts and graphs

Advantages:

- Structured & Accessible Content: HTML provides the framework for organizing financial data, ensuring it’s easy to access and SEO-friendly, while making the website accessible to all users, including those with disabilities.

- Responsive & Visually Appealing Design: CSS enhances the user experience by making the website visually appealing, ensuring it is mobile-friendly, and providing a consistent, clean layout across devices.

- Dynamic and Interactive Actions: JavaScript adds interactivity, allowing for real-time updates, dynamic financial visualizations, and input validation.

Backend Technologies

1. PHP

Overview: PHP is a widely-used server-side scripting language designed for web development. It handles backend processing, interacting with the database, and controlling user authentication.

Usage: PHP is used in your personal finance management website to handle user requests, such as logging in, signing up, and submitting forms for adding income and expenses. It processes the data and communicates with the MySQL database for storage and retrieval.

Advantages:

- Ease of Integration: PHP integrates seamlessly with MySQL for database management and is supported by most web servers.

- Wide Community Support: PHP has extensive documentation, and a large community provides tutorials, resources, and libraries.
- Cross-Platform Compatibility: PHP can run on various operating systems like Windows, Linux, and macOS.

Database Technologies

1. MySQL

Overview: MySQL is an open-source relational database management system (RDBMS), widely used for managing and storing data in web applications. It ensures data integrity and high performance.

Usage: MySQL is used to store and manage critical application data, including user credentials, income and expense records, balance information, and other financial data. SQL queries are used to interact with the data from PHP scripts.

Advantages:

- Scalability: MySQL is well-suited for handling large amounts of financial data, ensuring smooth performance even with high traffic.
- ACID Compliance: Guarantees data consistency and integrity with support for transactions, which is essential for financial data management.
- Cross-Platform Compatibility: MySQL can run on various operating systems, providing flexibility for deployment and hosting.

2.2.1 SQL

SQL in the Personal Finance Management System

SQL (Structured Query Language) is a standardized programming language used to manage and manipulate relational databases. In the context of the Personal Finance Management System, SQL plays a crucial role in defining the database structure, performing data operations, and ensuring data integrity. This section explains the role of SQL in the project, its advantages, and how it enables various functionalities.

1. Overview of SQL

- **Definition**: SQL is used to interact with relational databases, allowing users to perform CRUD (Create, Read, Update, Delete) operations.

- **Types of SQL Statements**:
  - **DDL (Data Definition Language)**: Used to define and manage database structures (e.g., CREATE, ALTER, DROP).
  - **DML (Data Manipulation Language)**: Used for inserting, updating, and deleting data (e.g., INSERT, UPDATE, DELETE).
  - **DQL (Data Query Language)**: Used for querying data from the database (e.g., SELECT).
  - **DCL (Data Control Language)**: Used to control access to data (e.g., GRANT, REVOKE).

2. Database Structure

The Personal Finance Management System relies on a well-structured database to store and manage data effectively. The database typically consists of several tables, each serving a distinct purpose. The primary tables in the project include:

2.1 Users Table

- **Purpose:** Stores information about users of the system, including login details and their roles (e.g., admin, regular user).
- **Columns:**
  - user_id: Unique identifier for each user (Primary Key).
  - username: User's chosen username.
  - password: Hashed password for authentication.
  - email: User's email address for communication.
  - phone: User's contact number.
  - role: Specifies whether the user is a participant or organizer.

2.2 Income Table

- **Purpose:** Stores details about the income entries made by users.
- **Columns:**
  - income_id: Unique identifier for each income entry(Primary Key).
  - user_id: Identifier for the user who added the income (Foreign Key).
  - source: Source of income (e.g., salary, freelance)
  - amount: The income amount.
  - Date: Date when the income was recorded.
  - Description: Additional details about the income.

2.3 Expense Table

- **Purpose:** Records details of expenses entered by the users.
- **Columns:**
  - Expense_id: Unique identifier for each expense entry (Primary Key).
  - User_id: Identifier for the user who added the expense (Foreign Key).
  - Category: Category of the expense.
  - Amount: The amount of the expense.
  - Date: Date when the expense was recorded.
  - Description: Additional details about the expense.

2.4 Transaction History Table

- **Purpose:** Maintains a history of all financial transactions (income and expenses) for auditing purposes.
- **Columns:**
  - Transaction_id: Unique identifier for each transaction (Primary Key).
  - User_id: Identifier for the user associated with the transaction (Foreign Key)..
  - Transaction_type: Type of transaction.
  - amount: The amount of the transaction.
  - date: Date of the Transcation.
  - balance_after_transaction: User's balance after the transaction.

2.5 Categories Table

- **Purpose:**Stores categories for organizing income and expense.
- **Columns:**
  - category_id: Unique identifier for each category (Primary Key).
  - category_name: Name of the category (e.g., salary, groceries, entertainment).
  - description: Optional description of the category.

2.6 Monthly Reports Table

- **Purpose:** Stores monthly summaries of income, expenses, and balance for each user.
- **Columns:**
  - report_id: Unique identifier for each monthly report (Primary Key).
  - user_id: Identifier for the user associated with the report (Foreign Key).
  - month: The month of the report.
  - total_income: Total income recorded in the month.
  - total_expenses: Total expenses recorded in the month.
  - Balance: Net balance after income and expenses.

4. SQL Operations in the Project

SQL operations play a key role in managing data within the Personal Finance Management System. Below are some essential operations performed using SQL in your project**:**

1) **User Management**

- **Retrieve All User:**

```
SELECT user_id, username, email, phone, role FROM
users;

FROM users;
```

Purpose: This query retrieves all employee details from the employees table to display them in the admin dashboard.

- **Insert New User:**

```
INSERT INTO users (username, password, email, phone,
role)

VALUES (%s, %s, %s, %s, %s);
```

Purpose: This query adds a new employee's information into the employees table.

- **Update User Details:**

```
UPDATE user

SET username=%s, password=%s, email=%s, phone=%s,
role=%s

 WHERE user_id=%s;
```

Purpose: This query updates the existing employee's information in the employees table.

- **Delete User Records:**

```
DELETE FROM users

WHERE user_id=%s
```

Purpose: This query deletes an employee's record from the employees table.

2. Income and Expense Management

- **Fetch Income Data:**

```
SELECT * FROM income;
```

Purpose: This query retrieves all columns and rows from the inventory table, which is used to populate the tree view with the stock details.

- **Fetch Expense Data:**

  ```
  SELECT expense_id, category, amount, date, description FROM expenses
  WHERE user_id = %s;
  ```

  Purpose: This query retrieves rows from the inventory table where the column specified by opt matches the search value (val).

3. **Report Generation**

- **Generate Monthly report for User:**

  ```
  SELECT SUM(amount) AS total_income FROM income
  WHERE user_id = %s AND MONTH(date) = %s AND YEAR(date) = %s;
  ```

  Purpose: This query retrieves the sale date and total amount for each sale from the sales table, which is used to create a line chart for sales by date.

- **Get User's Balance:**

  ```
  SELECT (SUM(i.amount) - SUM(e.amount)) AS balance FROM income i
  LEFT JOIN expenses e ON i.user_id = e.user_id
  WHERE i.user_id = %s;
  ```

Purpose: This query retrieves the employee IDs and their corresponding total sales, ordering them by total sales in descending order, to identify the top 5 employees by sales.

4. Data Integrity and Security

Data integrity and security are critical in maintaining reliable and accurate financial data. Below are the key strategies implemented to ensure data protection:

Data Integrity:

- Ensured through validation checks and constraints in the database schema (e.g., NOT NULL, unique constraints).
- Referential integrity maintained via foreign key relationships.

1) **Security Measures:**

- User authentication through hashed passwords.
- Role-based access control (admin, cashier, stock manager) to restrict data access.
- Regular backups and updates to protect data from loss or breaches.

5. SQL Performance Optimization

To ensure efficient data handling, performance optimization techniques may be employed:

1) **Indexing:**

- Use indexes on frequently queried columns to speed up data retrieval.
- Balance between read and write performance when creating indexes.

2) **Query Optimization**

- Writing efficient SQL queries and analyzing their execution plans can help identify and eliminate bottlenecks.

3) **Regular Maintenance:**

- Perform regular database maintenance tasks like vacuuming and analyzing.
- Monitor performance metrics to identify areas for improvement.

2.2.2 PHP, CSS, JavaScript

Use of PHP, CSS, JavaScript in the Personal Finance Management System

Introduction to PHP, CSS, JavaScript

PHP, CSS, JavaScript, and SQL to create a dynamic, user-friendly, and secure application. Each technology plays a specific role:

- PHP serves as the server-side scripting language, handling data processing, database interactions, and business logic.
- CSS ensures the front-end is visually appealing and responsive, adapting to various screen sizes.
- JavaScript enables dynamic content updates, form validations, and interactive features without requiring page reloads.
- SQL is used to manage and manipulate the data in the backend, handling user authentication, financial records, and transactions.

Application Structure

The system is organized into several components based on user roles (Admin, Client), each with specific features:

1. Clients can manage their income, expenses, and view financial summaries.
2. Users authenticate themselves through secure login or sign-up pages.

Creating the User Interface

The frontend of the system is designed using CSS and JavaScript:

Main Application Window

The main application window acts as the entry point for all users. When the application starts, the window is initialized with the following elements:

- Login Frame: This includes input fields for the user ID and password, allowing users to authenticate themselves before accessing the system. A "Login" button is provided for users to submit their credentials.
- JavaScript Validation: Before submission, JavaScript checks if the fields are filled correctly (e.g., valid email format for the username).

Admin Dashboard

After successful login, the admin is redirected to the admin dashboard, where they can perform various administrative functions, including:

- User Management:
    - Retrieve All Clients: Admins can see a complete list of clients, including their details such as username, email, and role.
    - Add New Client: Admins can add new clients by filling out a form. The backend handles adding this information to the database.
    - Update Client Details: Admins can edit client information, such as username or email, which is reflected in the database after the changes are saved.
    - DeleteClient: Admins can delete a client's record from the system..
- Generate Financial Reports: Admins can generate reports on total income, expenses, and the overall balance of all clients, making it easy to analyze financial trends.
- View System Overview: Admins can view basic system statistics, such as the number of clients and the total income or expenses in the system**.**

Client Dashboard

The Client Dashboard allows clients to manage their personal finances:

- Clients can add new income or expense records via forms. The backend processes these entries and updates the database.
- Clients can generate summaries of their income, expenses, and balance for specific time periods (e.g., monthly). This provides a snapshot of their financial situation.

Login and Authentication System

The login and sign-up system ensures secure user authentication:

- Password Hashing: When users sign up or change their passwords, the password is securely hashed to protect it from unauthorized access..
- Login Validation: During login, the entered password is verified against the stored password to confirm user identity.

Interaction with the Database

SQL is used extensively to manage data in the backend

Database Operations

- Retrieving DataSQL queries are used to fetch user data, income, expenses, and reports. For example, when an admin views client records, the system queries the database to retrieve relevant details.
- Inserting Data: New entries, such as income or expense records, are added to the database using SQL commands..
- Updating Data: When a client updates their information or adds a new income/expense, SQL queries update the relevant database records.

- Deleting DataWhen clients delete income or expense records, the system removes those records from the database.
- Input Validation: Before submitting forms (e.g., adding new income or expense), the system checks if fields contain valid data. For example, the amount field must contain a valid number.
- PHP Validation: On the server-side, the application ensures that all necessary fields are filled and that the data is in the correct format before saving it to the database.
- Confirmation Prompts: The system asks users for confirmation before performing irreversible actions, such as deleting records, to avoid accidental data loss.

3.REQUIREMENT AND ANALYSIS

Requirements and Analysis of the Personal Finance Management System Project

3.1 Introduction

The successful development of the Personal Finance Management System (PFMS) relies on clear requirements gathering and analysis. This phase establishes the blueprint for the project, outlining key features, functionalities, and constraints. Properly defined requirements ensure the system meets user needs and provides a streamlined approach to managing personal finances.

3.2 Project Objectives

The primary objectives of the Personal Finance Management System are as follows:

- Streamline Financial Management: Simplify personal finance tracking by automating income, expense, and budget management processes..
- Enhance User Experience: Create an intuitive, user-friendly interface for managing finances, with easy navigation for different users (admin, client).
- Improve Data Accuracy and Accessibility: Maintain real-time and accurate financial records while ensuring easy retrieval and manipulation of financial data.
- Provide Insights and Analysis: Offer tools for generating reports and budget analyses, enabling users to make informed financial decisions based on their financial history.

3.3 Functional Requirements

The functional requirements detail the specific features and operations the Personal Finance Management System must support:

1. User Authentication:
   - Users must log in with credentials (username and password).
   - Different user roles (Admin, Client) will have access to distinct features based on their role.
2. User Management:
   - Admins can add, update, and delete user accounts.
   - Admins should be able to retrieve user details (name, email, balance, etc.).
   - Admins must manage client roles and permissions for secure access.
3. Income and Expense Management:
   - Clients can access and search their transaction history by date range or category.

- Clients can view all income and expense entries, along with totals and categories.
- Clients should be able to categorize expenses and track their financial progress.

4. Transaction History:
   - Clients can access and search their transaction history by date range or category.
   - Admins must be able to view transaction details for all clients.
5. Budget Tracking:
   - Clients can set up and track monthly budgets for various categories (e.g., food, entertainment).
   - The system should provide alerts when the budget limit is exceeded.
6. Report Generation:
   - Admins should be able to generate reports for users based on specific criteria (e.g., month, user).
   - Clients should be able to generate a summary report of their financial status, including income, expenses, and budget performance.
7. Notifications:
   - The system must notify clients when they exceed their set budget or when certain financial thresholds are reached.
   - Admins should receive notifications for user activity, such as account changes or payment issues.

3.4 Non-Functional Requirements

Non-functional requirements focus on the system's quality attributes and constraints:

1. Usability:
   - The interface must be intuitive and easy to navigate for both admin and client users.
   - Clear feedback should be provided for actions such as login errors, form submissions, and report generation.
2. Performance:
   - The system should provide quick responses to user actions, such as login, data entry, and report generation, ensuring smooth navigation.
3. Scalability:
   - The system should handle increasing numbers of users and financial transactions without performance degradation.

4. Security:
    - User credentials must be securely stored, with proper encryption for passwords.
    - Role-based access control should ensure that different users can only access the appropriate system features.
    - The system should protect sensitive financial data from unauthorized access.
5. Reliability:
    - The system must be stable and reliable, minimizing downtime and ensuring continuous operation for users.
6. Data Integrity:
    - The system should maintain the accuracy and consistency of data throughout its lifecycle.
    - Validation mechanisms should prevent erroneous entries (e.g., non-numeric values in the amount field).

3.5 User Roles and Responsibilities

The system includes different roles, each with its set of responsibilities:

1. Admin:
    - Manages user accounts (create, update, delete).
    - Can generate financial reports and summaries for clients.
    - Oversees system security and access management.
2. Client:
    - Manages personal finances (income, expenses, budgets).
    - Views transaction history and reports.
    - Can set financial goals and receive alerts for exceeding budget limits.

3.6 System Constraints

Several constraints must be considered during the development of the Personal Finance Management System:

1. Technological Constraints:
    - The system must be developed using PHP (for server-side processing), CSS (for frontend design), JavaScript (for client-side interactivity), and SQL (for database management).

2. Budget Constraints:
    - The system should be developed within a defined budget, which may influence the scope of certain features or third-party integrations.
3. Time Constraints:
    - The system must be delivered within a specified timeline to meet the operational requirements of the users.

3.1 REQUIREMENT SPECIFICATION

The Requirement Specification document serves as a comprehensive guide for developing the Personal Finance Management Website. This document outlines the functionalities and performance expectations to ensure the application meets users' needs for tracking and managing personal finances..

3.1 Introduction

The Personal Finance Management Website is designed to help users monitor their financial activities, including income, expenses, and budgeting. The system provides a user-friendly interface for logging income and expenses, tracking financial health, and generating monthly summaries. This specification details the functional and non-functional requirements the system must fulfill to ensure a reliable, secure, and user-friendly application.

3.2 Functional Requirements

Functional requirements define the specific behaviors and capabilities the system must provide. They focus on what the system should do.

3.2.1 User Management

- User Registration: The system must allow new users to register with unique usernames, passwords, and email addresses.
- User Authentication: Users must be able to log in with their credentials, with secure password storage through hashing.
- Role Management: Users should be able to update their profile information, including email and password.

3.2.2 Income and Employee Management

- Income Entry: Users must be able to add income entries with details like source, amount, date, and category.
- Expense Entry: Users must be able to log expenses, specifying category, amount, date, and account name.
- Edit/Delete Transactions: Users should be able to edit or delete their income and expense records as needed.

3.2.3 Budgeting and Analysis

- Transaction Display: Users should be able to view a history of income and expense transactions, sorted by date and category.
- Category Tracking: Users should be able to categorize their expenses and income (e.g., food, transportation) for easy tracking.
- Monthly Summary: The system should display a summary of income and expenses for the month.

3.2.4 Transaction History

- Transaction Display: Users should be able to view a history of income and expense transactions, sorted by date and category.
- Search and Filter: The system should allow users to filter transaction history by date and category for easier analysis.

3.2.5 Security Management

- Password Management: Users should be able to change their password securely.
- Access Control: The system must enforce secure login to protect user data.

3.3 Performance Requirements

Performance requirements define the expected operational characteristics of the system. They address the system's efficiency and responsiveness under various conditions.

- Response Time: The system must respond to user actions, such as login and data entry, within 2 seconds.

- Transaction Throughput: The website should handle at least 50 transactions per minute without performance degradation.
- Concurrent Users: The system must support at least 20 concurrent users without noticeable lag.
- Data Integrity: The system must ensure data integrity during transactions, preventing data loss or duplication.
- Availability: The website should maintain a minimum uptime of 99.5% to ensure user access without interruptions.
- Backup and Recovery: The system must implement regular backups and provide recovery options to restore data within one hour after any unexpected failure.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

- The hardware requirements for the Personal Finance Management System are moderate, given that the application is web-based and uses PHP, MySQL, and client-side technologies like HTML, CSS, and JavaScript. The following specifications are recommended:
- Processor: Any modern dual-core processor or higher (e.g., Intel Core i3 or AMD Ryzen 3).
- RAM: At least 4 GB (8 GB recommended for smooth multitasking).
- Storage: Minimum of 500 MB of available disk space for the application and database files.
- Display: A screen with a minimum resolution of 1366x768 pixels for better usability.
- Input Device: Standard keyboard and mouse for interaction.
- Software Requirements
- The software requirements include the following components:
- Operating System: The system can run on any operating system that supports Python, such as:
- Windows 10 or later

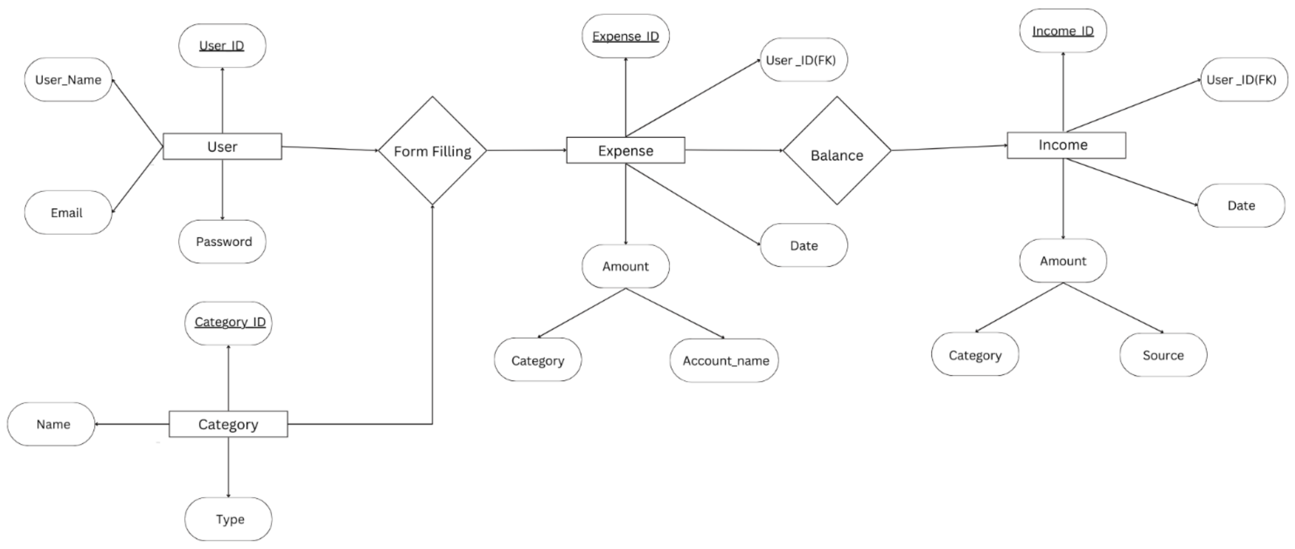
- macOS Mojave or later
- Linux (Ubuntu 18.04 or later)
- PHP: To handle server-side logic and interact with the database.
- Apache web server to serve the PHP code.
- MySQL Database: The MySQL Workbench or phpMyAdmin can be used for managing the database during development.
- Development Tools:
- A Visual Studio Code: Popular, lightweight, with support for PHP, CSS, JavaScript, and HTML.
- PHP Storm: A dedicated IDE for PHP with advanced features and debugging tools.
- Sublime Text or Notepad++: For basic editing if you prefer lighter editors.
- Additional Libraries
- The following libraries may be used for additional functionality:
- CSS Framework: Tailwind CSS for responsive, minimal, and modern design of the frontend interface.
- HTML5: For creating the structure and content of your web pages (forms, tables, etc.).
- These hardware and software requirements ensure that the Personal Finance Management System operates efficiently, providing users with a responsive and functional interface for managing the personal finance operations.
- 3.3 ARCHITECTURE DIAGRAM
- The The architecture of the Personal Finance Management Website (PFMS) is structured into three primary layers: Frontend, Backend, and Database. This modular architecture ensures the system is scalable, maintainable, and easy to troubleshoot. Each layer plays a distinct role in supporting the system's functionality, ensuring smooth operation and a seamless user experience..
- 1. Frontend Layer

- The The Frontend Layer is the user interface (UI) of the Personal Finance Management Website, designed to provide an intuitive, responsive, and user-friendly experience. It is built using HTML, CSS, and JavaScript, and it consists of the following key components:
- Dashboards: Displays key financial data such as balance, total income, and total expenses. It provides a summary view to help users monitor their financial status at a glance.
- Forms and Input Controls: Forms for adding income and expense records. These include fields for date, category, amount, and description. The forms use various input controls like text boxes, dropdowns, and date pickers.
- Transaction History Table: A table that lists all recent transactions (income and expenses) in a structured format, allowing users to track their financial activities..
- Error Messages: Displayed when users input invalid data (e.g., incorrect format, empty fields, etc.). For example, if an expense amount is missing or incorrect, the system will display an error message.
- Success Notifications: Confirmation messages appear after successful data entries, such as adding a new expense or income, helping users understand that their input was processed successfully.
- 2. Backend Layer
- The Backend Layer serves as the core application logic of the system. It is responsible for handling business rules, processing user inputs, and managing database interactions. The backend is implemented in PHP, and it communicates with both the frontend and the database to deliver the required functionality.
- Application Logic and Business Rules:
- Income and Expense Management: The backend handles the logic for adding, updating, and deleting income and expense records. It ensures that transactions are correctly logged and that the balance is updated accordingly.

- User Authentication: The backend validates user credentials during login, ensuring that users are authenticated before accessing their personal finance data. It also handles session management to track logged-in users.
- Balance Calculation: The backend calculates the current balance by subtracting the total expenses from the total income, ensuring real-time updates of the financial balance on the dashboard.
- Data Validation and Processing: The backend validates the data received from the frontend. For example, it checks if the amount entered in an expense form is a valid numeric value and whether all required fields (e.g., date, category) are filled out.
- Role-Based Access Control: The system supports different user roles (e.g., Admin and User). The backend manages which functionalities are accessible based on the user's role (e.g., admins can view all data, while regular users only see their own income/expense records).
- 3. Database Layer
- The Database Layer is responsible for storing, retrieving, and managing all the financial data of the system. MySQL is used as the Database Management System (DBMS), which ensures that data is well-organized, accessible, and safe.
- Database Tables: The core tables in the database include:
- Users: Stores user data, including usernames, passwords (hashed), roles (admin/user), and contact information.
- Income: Stores details of income transactions, including amount, category, date, and user ID.
- Expenses: Stores expense details, including amount, category, description, and date
- Transactions: A table that records all transactions, linking income and expense entries to the user.
- Data Integrity and Relationships Primary and Foreign Keys: Ensures data integrity by linking tables together. For example, the Income and Expense tables reference the Users table via foreign keys, ensuring that transactions are associated with the correct user.
- Database Operations: The backend communicates with the database to execute SQL commands that perform operations such as:

- CRUD Operations: The backend interacts with the database to execute Create, Read, Update, and Delete (CRUD) operations. For instance, when a user adds an income record, the backend constructs an SQL INSERT query to add the data to the Income table.
- Interaction Between Layers
- The layers in the Personal Finance Management System interact in the following way::
- Frontend to Backend Communication: When a user interacts with the frontend (e.g., submitting a form to add income or expense), the frontend sends an HTTP request to the backend using AJAX (or regular form submission) to process the data.
- Backend to Database Queries The backend processes the data (e.g., validating the form inputs) and interacts with the MySQL database by sending SQL queries (e.g., INSERT, UPDATE, SELECT) to store or retrieve data.
- Database to Backend Data Retrieval: The database returns the necessary data (e.g., income and expense records) to the backend. The backend processes the data (e.g., calculating total income, total expenses, and balance) and formats it for display.
- Backend to Frontend Response: After processing the data, the backend sends the results (e.g., success message or data updates) back to the frontend, which then updates the user interface. For example, once an expense is added, the Transaction History Table is updated to show the new entry..
- Benefits of the PFMS Architecture
- This layered architecture offers several advantages:
- Modularity: Each layer operates independently, making the system easier to maintain and extend. For example, the frontend can be updated without affecting the backend or database layers, and vice versa.
- Scalability: This architecture allows the system to scale horizontally by adding more database storage or vertical scaling by upgrading the backend server if the number of users increases.
- Maintainability: With clearly defined responsibilities for each layer, the system is easier to maintain. Changes or bug fixes can be implemented in one layer without impacting others, making development more efficient.

3.4 ER DIAGRAM



3.5 NORMALIZATION

Normalization is essential in database design to reduce redundancy and ensure data integrity. By organizing data in a logical and efficient way, normalization enhances the performance and reliability of the Personal Finance Management Website database. The following outlines the importance of normalization for the website's database and describes how various normal forms are applied.

6.1 Importance of Normalization

1. **Elimination of Redundancy:**
   Redundant data can cause inconsistencies. For example, storing a user's name in both the Income and Expense tables would mean updating the name in multiple locations, which could lead to discrepancies. Normalization ensures that user details are stored in a single **User** table, eliminating redundancy.

2. **Improved Data Integrity:**
   By separating data into distinct tables and establishing relationships between them, normalization helps maintain accurate and consistent data. For instance, changes in the **User** table, such as updating an email address, are reflected across related tables without the risk of inconsistency.
3. **Ease of Maintenance:**
   A normalized database simplifies data updates. When changes need to be made, such as updating an income category, it only needs to be done once in the **Category** table, reducing the risk of errors or omissions across the database.
4. **Efficient Querying:**
   Normalized tables improve data retrieval speed. A well-structured database allows the system to retrieve user transactions, balances, and summaries more efficiently, providing a better response time for users.

5. **Facilitated Data Relationships:**
   Normalization supports the establishment of relationships between different data entities, helping to maintain referential integrity. For example, the relationship between users and their transactions is maintained accurately when User, Income, and Expense entities are stored separately, making it easy to link and retrieve transactions for each user.

6.2 Normal Forms

Normalization is achieved through stages known as normal forms, each addressing specific data anomalies. Here's how these normal forms are applied to your website database:

1. **First Normal Form (1NF):**
   - Definition: A table is in 1NF if all columns contain atomic values, with each entry in a column being of the same data type and no repeating groups or arrays.
   - Application in Website: In the User table, each user attribute (e.g., username, email) is stored in separate columns, ensuring atomicity.
2. **Second Normal Form (2NF):**
   - Definition: A table is in 2NF if it's in 1NF, and all non-key attributes are fully dependent on the primary key.
   - Application in Website: In the Income table, attributes like amount and date depend entirely on the primary key (income_id), ensuring no partial dependencies.

3. **Third Normal Form (3NF):**
    - Definition: A table is in 3NF if it’s in 2NF and all attributes are non-transitively dependent on the primary key, meaning non-key attributes do not depend on other non-key attributes.
    - Application in Website: In the Expense table, expense details like date and amount should not depend on the category name, maintaining clear separation and preventing update anomalies.

6.3 Practical Application of Normalization in HMS

Normalization is applied to various tables in the Personal Finance Management Website database to create an optimal data structure:

- User Table: Stores unique user information, preventing redundant entries and ensuring that user interactions, such as login, are securely managed.
- Income Table: Records individual income entries, linking each to a specific user, and stores details like amount, date, and category, making it easy to query and analyze income data.
- Expense Table: Captures details of each expense, linking it to the user and storing information such as amount, date, category, and account name. This ensures accurate expense tracking.
- Category Table: Organizes income and expense categories in a single location, allowing for easy updates and ensuring consistency across income and expense entries.

- NTransaction History View: An optional view could be created to summarize income and expense entries per user, simplifying data retrieval for generating reports without storing redundant data.

4.PROGRAM CODE

PHP,HTML,CSS,JAVASCRIPT,SQL CODE:

```
<?php
include 'dbconnect.php'; // Include database connection
?>
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sign Up Page</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap"
rel="stylesheet">
    <style>
        body {
  font-family: 'Roboto', sans-serif;
  background-color: #f4f4f4;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}
.signup-container {
  background-color: #fff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  width: 300px;
}
.signup-container h2 {
  text-align: center;
  margin-bottom: 20px;
  color: #333;
}
.signup-container .input-group {
  margin-bottom: 15px;
}
.signup-container .input-group label {
  display: block;
  margin-bottom: 5px;
  color: #555;
}
.signup-container .input-group input {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 4px;
}
.signup-container .input-group input:focus {
  border-color: #007bff;
  outline: none;
}
.signup-container .btn {
  width: 100%;
  padding: 10px;
  background-color: #007bff;
  border: none;
  border-radius: 4px;
  color: #fff;
  font-size: 16px;
  cursor: pointer;
}
.signup-container .btn:hover {
  background-color: #0056b3;
}
.signup-container .login-link {
  text-align: center;
  margin-top: 20px;
}
.signup-container .login-link a {
  color: #007bff;
  text-decoration: none;
```

```
}
.signup-container .login-link a:hover {
  text-decoration: underline;
}
    </style>
</head>
<body>
    <div class="signup-container">
        <h2>Sign Up</h2>
        <?php
        include 'dbconnect.php'; // Include database connection

        $message = "";
        if ($_SERVER["REQUEST_METHOD"] == "POST") {
            $user = mysqli_real_escape_string($conn, $_POST["username"]);
            $email = mysqli_real_escape_string($conn, $_POST["email"]);
            $pass = mysqli_real_escape_string($conn, $_POST["password"]);
            $hashed_password = password_hash($pass, PASSWORD_DEFAULT);

            $sql = "INSERT INTO users (username, email, password) VALUES ('$user', '$email',
'$hashed_password')";
            if ($conn->query($sql) === TRUE) {
                $message = "<p style='color: green;'>Sign up successful!</p>";
            } else {
                $message = "<p style='color: red;'>Error: " . $sql . "<br>" . $conn->error . "</p>";
            }
            $conn->close();
        }
        ?>
        <?php echo $message; ?>
        <form method="POST" action="">
            <div class="input-group">
                <label for="username">Username</label>
                <input type="text" id="username" name="username" required>
            </div>
            <div class="input-group">
                <label for="email">Email</label>
                <input type="email" id="email" name="email" required>
            </div>
            <div class="input-group">
                <label for="password">Password</label>
                <input type="password" id="password" name="password" required>
            </div>
            <button type="submit" class="btn">Sign Up</button>
        </form>
        <div class="login-link">
            <p>Already have an account? <a href="login.php">Login</a></p>
        </div>
    </div>
</body>
</html>
<?php
include 'dbconnect.php'; // Include database connection
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Page</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap"
rel="stylesheet">
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            background-color: #f4f4f4;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .login-container {
            background-color: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
            width: 300px;
        }
        h2 {
            text-align: center;
            margin-bottom: 20px;
        }
        input[type="text"],
        input[type="password"] {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        button {
            width: 100%;
            padding: 10px;
            background-color: #007bff;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            font-size: 16px;
        }
        button:hover {
```

```
            background-color: #0056b3;
        }
        .signup-link {
            text-align: center;
            margin-top: 10px;
        }
        .signup-link a {
            color: #007bff;
            text-decoration: none;
        }
        .signup-link a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <div class="login-container">
        <h2>Login</h2>
        <?php
        $message = ""; // Initialize message variable
        if ($_SERVER["REQUEST_METHOD"] == "POST") {
            // Escape user inputs for security
            $user = mysqli_real_escape_string($conn, $_POST["username"]);
            $pass = mysqli_real_escape_string($conn, $_POST["password"]);

            // Query to check if user exists
            $sql = "SELECT * FROM users WHERE username='$user'";
            $result = mysqli_query($conn, $sql);
            if (mysqli_num_rows($result) > 0) {
                $row = mysqli_fetch_assoc($result);
                // Verify the password
                if (password_verify($pass, $row['password'])) {
                    // Start session and redirect to welcome page
                    session_start();
                    $_SESSION['username'] = $user;
                    header("Location: welcome.php");
                    exit();
                } else {
                    $message = "Invalid password.";
                }
            } else {
                $message = "No user found with that username.";
            }
        }
        ?>
        <form method="POST" action="">
            <input type="text" name="username" placeholder="Username" required>
            <input type="password" name="password" placeholder="Password" required>
            <button type="submit">Login</button>
            <?php if (!empty($message)): ?>
                <p style="color: red; text-align: center;"><?php echo $message; ?></p>
            <?php endif; ?>
        </form>
        <div class="signup-link">
            <p>Don't have an account? <a href="signup.php">Sign up</a></p>
        </div>
    </div>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Multiple Forms</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap"
rel="stylesheet">
```

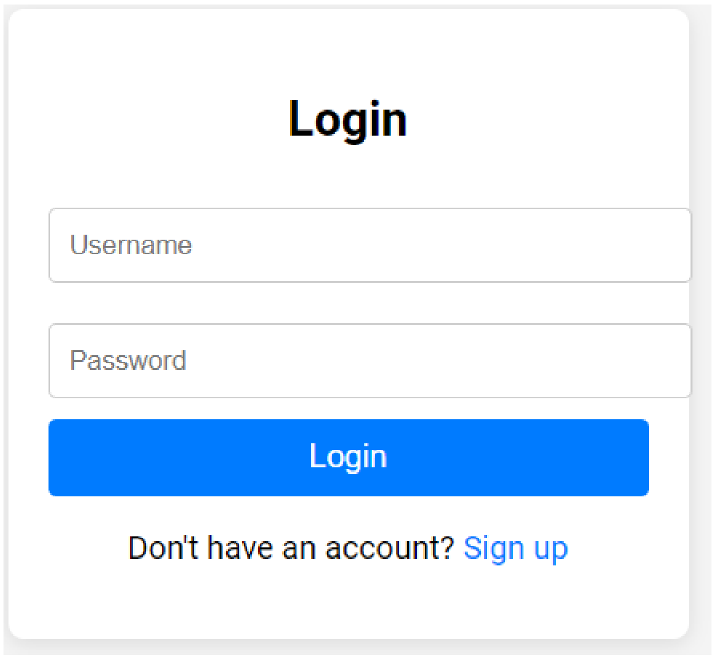
```
<style>
  /* General Styles */
  body {
    font-family: 'Roboto', sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
  }
  .form-container {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    width: 300px;
  }
  .form-container h2 {
    margin-bottom: 20px;
    font-size: 24px;
    color: #333;
  }
  .form-container label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
    color: #555;
  }
  .form-container input[type="text"],
  .form-container input[type="date"],
  .form-container input[type="number"] {
    width: 100%;
```

```
            padding: 10px;
            margin-bottom: 15px;
            border: 1px solid #ccc;
            border-radius: 4px;
            font-size: 14px;
        }
        .form-container input[type="submit"] {
            width: 100%;
            padding: 10px;
            background-color: #28a745;
            border: none;
            border-radius: 4px;
            color: #fff;
            font-size: 16px;
            cursor: pointer;
            transition: background-color 0.3s;
        }
        .form-container input[type="submit"]:hover {
            background-color: #218838;
        }
    </style>
</head>
<body>
    <div class="form-container">
        <h2>Income Form</h2>
        <form action="income/data" method="post">
            <label for="date">Date *</label>
            <input class="date1" type="date" id="date" name="date" required>

            <label for="bank_name">Bank Name *</label>
            <input type="text" id="bank_name" name="bank_name" required>

            <label for="source">Source *</label>
            <input type="text" id="source" name="source" required>
            <label for="amount">Amount *</label>

            <input type="number" id="amount" name="amount" required placeholder="Enter Amount"
min="-999999999" max="999999999">


            <input type="submit" value="Submit">

        </form>

    </div>

</body>

</html>

<html>

<head>

    <title>Personal Finance Management</title>

    <script src="https://cdn.tailwindcss.com"></script>

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">

    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap"
rel="stylesheet">

    <style>

        body {

            font-family: 'Roboto', sans-serif;

            background-color: #f4f4f4;

            margin: 0;

            padding: 0;

        }

        .container {

            max-width: 1200px;

            margin: 0 auto;

            padding: 20px;

        }

        .card {

            background-color: #fff;

            padding: 20px;

            border-radius: 8px;

            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            margin-bottom: 20px;
        }
        .card h2 {
            font-size: 24px;
            font-weight: 500;
            margin-bottom: 10px;
        }
        .card p {
            font-size: 18px;
            color: #555;
        }
        .transaction-history {
            max-height: 400px;
            overflow-y: auto;
        }
        .transaction-item {
            display: flex;
            justify-content: space-between;
            padding: 10px 0;
            border-bottom: 1px solid #ddd;
        }
        .transaction-item:last-child {
            border-bottom: none;
        }
        .transaction-item p {
            margin: 0;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="grid grid-cols-1 md:grid-cols-2 gap-4">
            <div class="card">
                <h2>Current Balance</h2>
                <p>$5,000.00</p>
            </div>
            <div class="card">
                <h2>This Month's Income</h2>
                <p>$2,500.00</p>
            </div>
            <div class="card">
                <h2>This Month's Expense</h2>
                <p>$1,200.00</p>
            </div>
            <div class="card md:col-span-2">
                <h2>Transaction History</h2>
                <div class="transaction-history">
                    <div class="transaction-item">
                        <p>01/10/2023 - Salary</p>
                        <p class="text-green-500">+ $2,500.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>02/10/2023 - Grocery</p>
                        <p class="text-red-500">- $150.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>03/10/2023 - Electricity Bill</p>
                        <p class="text-red-500">- $100.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>04/10/2023 - Dinner</p>
                        <p class="text-red-500">- $50.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>05/10/2023 - Gym Membership</p>
                        <p class="text-red-500">- $60.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>06/10/2023 - Freelance Work</p>
                        <p class="text-green-500">+ $300.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>07/10/2023 - Coffee</p>
                        <p class="text-red-500">- $5.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>08/10/2023 - Internet Bill</p>
                        <p class="text-red-500">- $40.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>09/10/2023 - Movie</p>
                        <p class="text-red-500">- $20.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>10/10/2023 - Shopping</p>
                        <p class="text-red-500">- $200.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>11/10/2023 - Car Maintenance</p>
                        <p class="text-red-500">- $300.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>12/10/2023 - Rent</p>
                        <p class="text-red-500">- $800.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>13/10/2023 - Bonus</p>
                        <p class="text-green-500">+ $500.00</p>
                    </div>
                    <div class="transaction-item">
                        <p>14/10/2023 - Donation</p>
                        <p class="text-red-500">- $100.00</p>
```

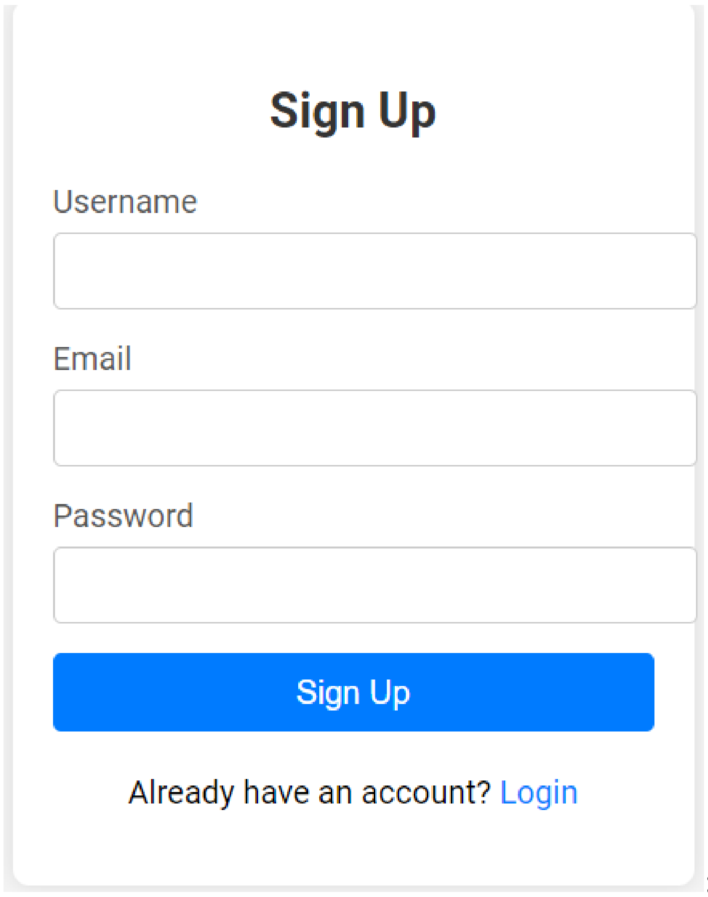
```
                    </div>
                    <div class="transaction-item">
                        <p>15/10/2023 - Subscription</p>
                        <p class="text-red-500">- $10.00</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script>
        function adjustGrid() {
            const container = document.querySelector('.container');
            const grid = container.querySelector('.grid');
            const width = window.innerWidth;

            if (width < 768) {
                grid.classList.remove('md:grid-cols-2');
                grid.classList.add('grid-cols-1');
            } else {
                grid.classList.remove('grid-cols-1');
                grid.classList.add('md:grid-cols-2');
            }
        }

        window.addEventListener('resize', adjustGrid);
        window.addEventListener('load', adjustGrid);
    </script>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Expense Form for Personal Finance Management</title>

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">

    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap"
rel="stylesheet">

    <style>

        /* General Styles */

        body {

            font-family: 'Roboto', sans-serif;

            background-color: #f4f4f4;

            margin: 0;

            padding: 0;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

        }

        .expense-form-container {

            background-color: #fff;

            padding: 20px;

            border-radius: 8px;

            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

            width: 100%;

            max-width: 500px;

        }

        .expense-form-container h2 {

            margin-bottom: 20px;

            font-weight: 500;

            color: #333;

            font-size: 24px;

        }

        .expense-form-container label {

            display: block;

            margin-bottom: 5px;
    font-weight: 500;
    color: #555;
  }
  .expense-form-container input,
  .expense-form-container select,
  .expense-form-container textarea {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ddd;
    border-radius: 4px;
    font-size: 16px;
  }
  .expense-form-container button {
    width: 48%;
    padding: 10px;
    border: none;
    border-radius: 4px;
    color: #fff;
    font-size: 16px;
    cursor: pointer;
    display: inline-flex;
    align-items: center;
    justify-content: center;
  }
  .expense-form-container .save-button {
    background-color: #28a745;
    transition: background-color 0.3s;
  }
  .expense-form-container .save-button:hover {
    background-color: #218838;
  }
  .expense-form-container .reset-button {
    background-color: #dc3545;
```

```
            transition: background-color 0.3s;
        }
        .expense-form-container .reset-button:hover {
            background-color: #c82333;
        }
        .button-group {
            display: flex;
            justify-content: space-between;
        }
    </style>
</head>
<body>
    <div class="expense-form-container">
        <h2>Expense Form</h2>
        <form>
            <label for="date">Date <span style="color: red;">*</span></label>
            <input type="date" id="date" name="date" required>

            <label for="category">Category <span style="color: red;">*</span></label>
            <select id="category" name="category" required>
                <option value="food">Food</option>
                <option value="transportation">Transportation</option>
                <option value="utilities">Utilities</option>
                <option value="entertainment">Entertainment</option>
                <option value="health">Health</option>
                <option value="other">Other</option>
            </select>

            <label for="amount">Amount <span style="color: red;">*</span></label>
            <input type="number" id="amount" name="amount" step="0.01" required>

            <label for="account">Account Name <span style="color: red;">*</span></label>
            <input type="text" id="account" name="account" required>
            <div class="button-group">
                <button type="submit" class="save-button"><i class="fas fa-save"></i> Save
Expense</button>
                <button type="reset" class="reset-button"><i class="fas fa-undo"></i> Reset</button>
            </div>
        </form>
    </div>
</body>
</html>
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "my_database";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

login page:

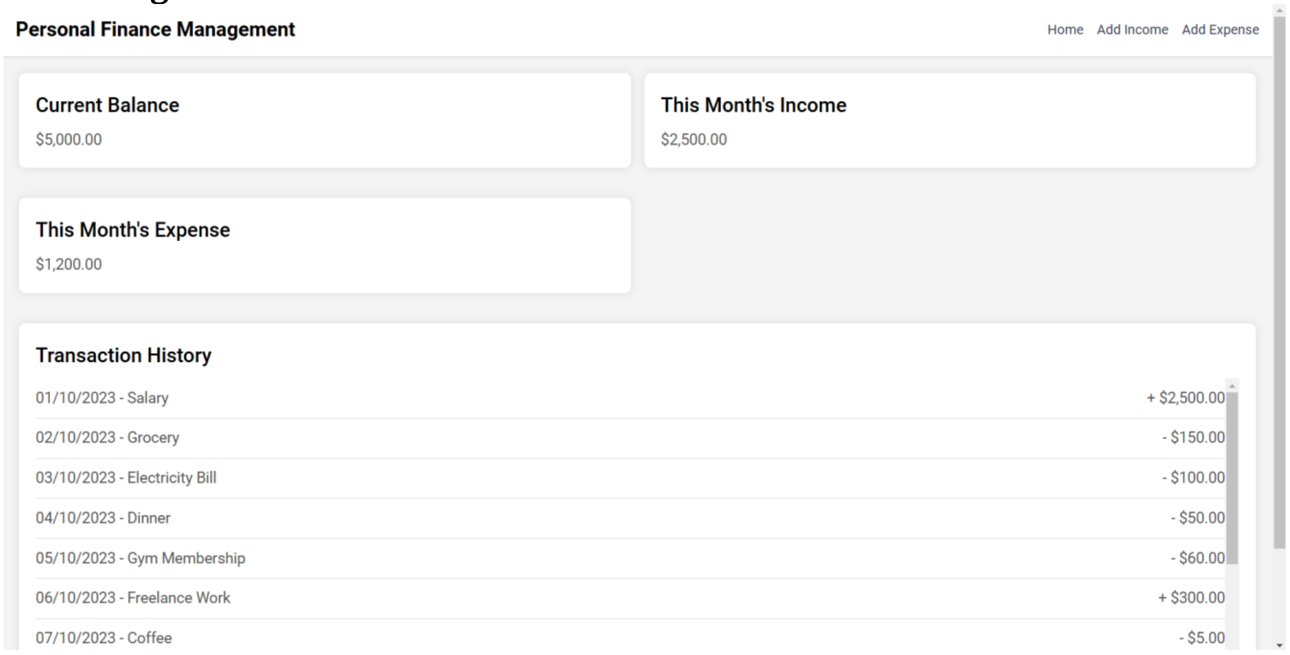

Login
Username
Password
Login
Don't have an account? Sign up

Signup Page:


Sign Up
Username
Email
Password
Sign Up
Already have an account? Login

:

Home Page:


Personal Finance Management

Home Add Income Add Expense

Current Balance

$5,000.00

This Month's Income

$2,500.00

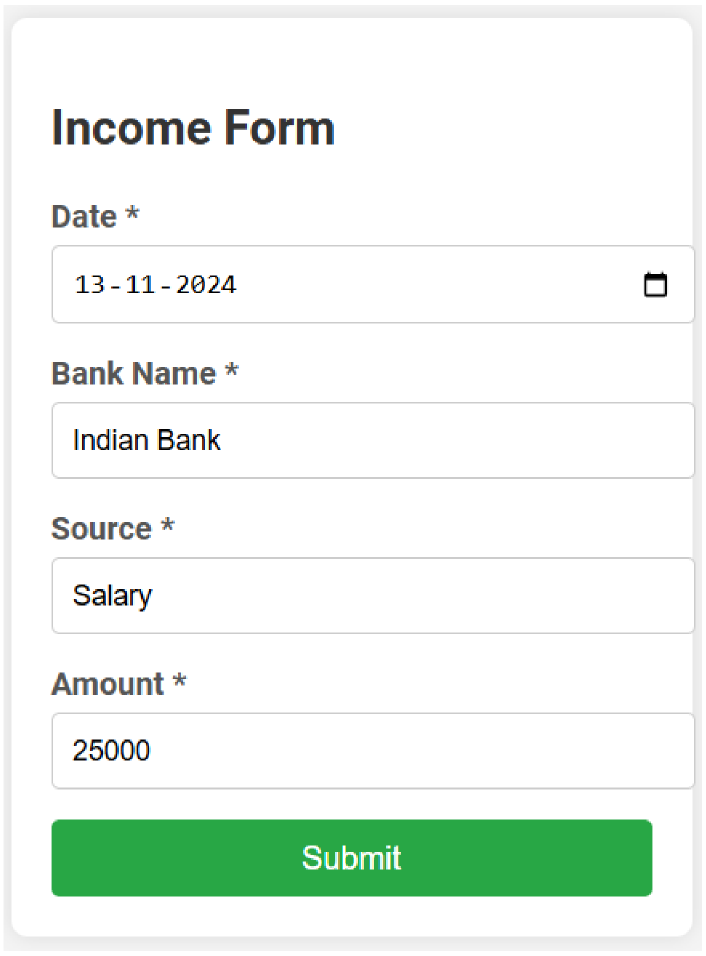
This Month's Expense

$1,200.00

Transaction History

01/10/2023 - Salary	+ $2,500.00
02/10/2023 - Grocery	- $150.00
03/10/2023 - Electricity Bill	- $100.00
04/10/2023 - Dinner	- $50.00
05/10/2023 - Gym Membership	- $60.00
06/10/2023 - Freelance Work	+ $300.00
07/10/2023 - Coffee	- $5.00

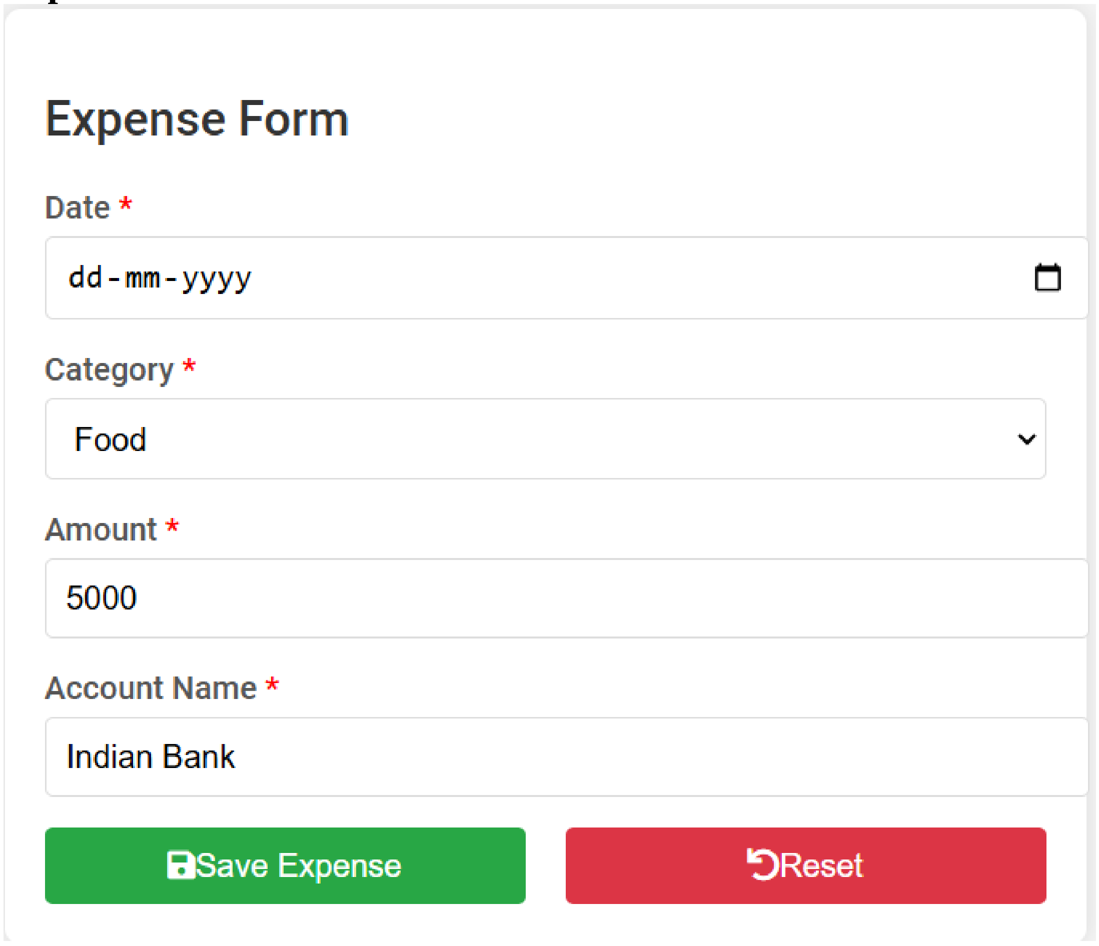
Income Form:


Income Form

Date *	13-11-2024
Bank Name *	Indian Bank
Source *	Salary
Amount *	25000

Submit

Expense Form



Expense Form

Date *

dd-mm-yyyy

Category *

Food

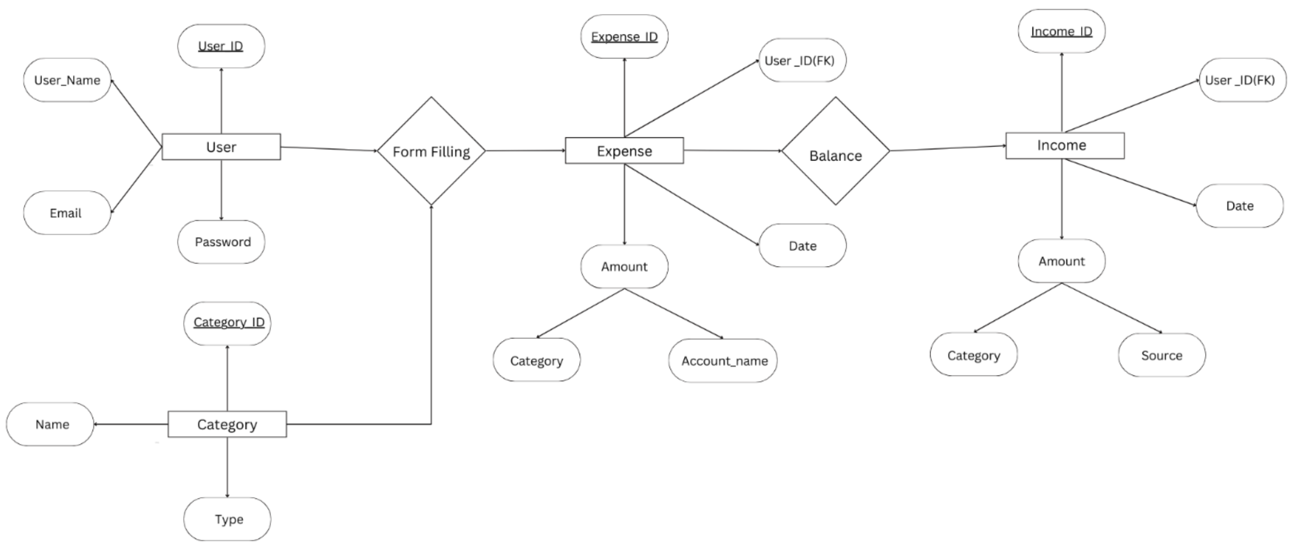
Amount *

5000

Account Name *

Indian Bank

5.RESULTS A



ND DISCUSSION

The Personal Finance Management Website was designed to assist users in tracking their finances, including income, expenses, and monthly summaries. This section provides an overview of the results achieved, insights gained during development, and considerations for future improvements.

5.1 System Functionality

The Personal Finance Management Website integrates several core functionalities to meet user needs for financial tracking and organization:

- Dashboard Overview: The dashboard provides a summary of the user's current balance, total monthly income, and expenses. Users can view their financial status at a glance, which helps them stay on top of their finances and make informed decisions.

- Income and Expense Management: The system allows users to log income and expenses with details such as amount, date, category, and source or account name. These functions make it easy for users to accurately track their finances in real-time.

- Transaction History: Users can access a transaction history displaying recent income and expense records, sorted by date. This feature supports financial transparency and helps users review and manage their past transactions effectively.

5.2 User Experience

The user experience (UX) was a primary consideration during development, with feedback collected from initial user testing. Key aspects of the user experience included:

- Simple and Intuitive Design: The interface was designed for simplicity and ease of navigation. Users reported that the layout was easy to understand and use, allowing them to quickly log their income and expenses without confusion.

- Responsiveness: The website was noted for its responsiveness, with quick load times and minimal delay in form submissions and data retrieval. This smooth experience contributes to ease of use and user satisfaction.

- Error Prevention: The website incorporates input validation to prevent common errors, such as ensuring numerical values in the amount field. Feedback mechanisms, like confirmation messages and error alerts, help guide users through their actions and prevent mistakes.

5.3 Performance Analysis

- A performance analysis was conducted to assess the speed, responsiveness, and reliability of the website. The following observations were noted:

- Database Efficiency: The integration of MySQL for data storage allowed for efficient data retrieval and storage. Optimized SQL queries facilitated quick access to user transaction history and balance calculations, providing users with timely responses.

- Data Security: Security measures were prioritized, especially for user passwords, which are securely hashed to protect sensitive data. This ensures data integrity and protects user privacy, which is essential for financial applications.

- Error Handling: Comprehensive error handling was implemented, with user-friendly error messages guiding users to resolve issues like incorrect login details or invalid data entries.

5.4 Challenges Encountered

Several challenges arose during the project's development:

- Database Connectivity Issues: Initial setup of the MySQL database within XAMPP encountered configuration challenges. This was resolved through detailed testing and adjustments, resulting in stable connectivity.

- UI Consistency: Ensuring consistent design across the site’s different modules required iterative testing. Attention to detail was essential to maintain visual coherence and a professional look.

- Scalability Considerations: As the project developed, considerations for scaling emerged. While the current implementation meets individual user needs, future growth may require database and interface adjustments to handle more extensive data, such as daily or annual summaries.

5.5 Future Enhancements

Based on feedback and performance analysis, the following enhancements are proposed for future development:

- Budgeting and Goals: Adding tools for setting monthly budgets and tracking financial goals would offer users more ways to manage and plan their finances effectively

- Category Analysis and Trends: By incorporating insights into spending habits across categories, users could gain a clearer understanding of their expenses, helping them make more informed financial decisions.

- Mobile Compatibility: Developing a mobile-friendly version of the website would improve accessibility, enabling users to log transactions on the go and check their financial status from their mobile devices.

- Export to CSV: Providing an option to export transaction history as a CSV file would allow users to analyze their data in external applications or share it with financial advisors.

- User Support Resources: Creating a FAQ section or brief tutorials on navigating the website would enhance user support and confidence, ensuring that users can maximize the website’s features.

CONCLUSION

Conclusion of the Personal Finance Management System Project

The Personal Finance Management Website project has successfully created an intuitive, user-centric platform designed to help individuals manage their finances more effectively. By integrating key features such as real-time expense tracking, customizable budgeting tools, savings goal management, and investment monitoring, the website empowers users to make informed financial decisions and achieve their financial goals.

During development, a strong emphasis was placed on providing a seamless user experience, with a clean and easy-to-navigate interface that ensures accessibility for all users, regardless of their financial knowledge. The use of secure, scalable technologies ensures that the system remains reliable and can handle increasing user data while maintaining privacy and security.

Key outcomes of the project include enhanced financial awareness for users, improved financial discipline through budgeting and goal-setting, and the ability to track financial progress in real time. Users have expressed high satisfaction with the platform's ease of use and its ability to consolidate multiple financial tasks in one place.

Looking ahead, the Personal Finance Management Website presents opportunities for further development, including mobile app integration, AI-driven financial recommendations, and advanced reporting features. These improvements will enhance user experience and further support financial literacy and management.

In conclusion, the Personal Finance Management Website has proven to be a valuable tool in helping individuals take control of their financial lives, providing them with the tools and insights needed to make smarter financial decisions. The platform is well-positioned to evolve and meet the ever-changing needs of users, ultimately promoting financial well-being and long-term security.

7. REFERENCES

https://developer.mozilla.org/en-US/docs/Web/HTML

Database Management SystemCS23332

https://developer.mozilla.org/en-US/docs/Web/CSS

MySQL Documentation (2023). "MySQL Reference Manual." Retrieved from MySQL.com.