# PERSONAL FINANCE MANAGER

**A Project Report**

***Submitted by***

## KRISHNA JAIN

## PRATIK JAIN

## YASH KIKANI

***Under the Guidance of***

Prof. Vijayetha Thoday/Prof. Rojal Tuscano

for the course “Web Programming”

***in partial fulfillment for the award of the degree of***

**BTECH** /**MBATECH**

At



**Department of Computer Engineering,**

**MPSTME, NMIMS.**

**APRIL,2025**

## DECLARATION

We Krishna Jain, Pratik Jain and Yash Kikani Roll No. N030, N031 and N121 respectively from BTECH/MBATECH, IV semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. ( Source: IEEE, The institute, Dec. 2004)
4. We have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. We affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Names: Krishna Jain, Pratik Jain and Yash Kikan

Roll Nos. : N030, N031 and N121

Place: Mumbai

Date:

## CERTIFICATE

This is to certify that the project entitled “Personal Finance Tracker” is the bonafide work carried out by Krishna Jain, Pratik Jain and Yash Kikani of BTECH/MBATECH, MPSTME (NMIMS), Mumbai, during the VI semester of the academic year 2024-2025, in partial fulfillment of the requirements for the Course Web Programming

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Vijayetha Thoday/Prof. Rojal Tuscano

Internal Mentor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Examiner 1 Examiner 2

## Table of contents

**CHAPTER NO. TITLE PAGE NO.**

1. INTRODUCTION

SCOPE

PURPOSE

1. DESCRIPTION OF MODULES
2. SOFTWARES/API’s/ DATABASE USED WITH DESCRIPTION
3. METHODS IMPLEMENTED
4. CODE SNIPPETs
5. GUI/DATABASE SCREENSHOTS
6. CONCLUSION & FUTURE SCOPE
7. APPLICATIONs IF ANY

Chapter 1: Introduction

1.1 Project Overview

FinTrack is a comprehensive personal finance management system designed to help users track, analyze, and optimize their financial activities. In today's fast-paced world, managing personal finances effectively has become increasingly important. FinTrack addresses this need by providing a user-friendly platform that enables individuals to monitor their income, expenses, budgets, and financial goals in one centralized location.

The system offers a range of features including expense tracking, budget planning, financial goal setting, and detailed reporting capabilities. By providing users with insights into their spending habits and financial health, FinTrack empowers them to make informed financial decisions and work towards achieving their financial objectives.

1.2 Objectives

The primary objectives of the FinTrack project are:

1. Simplify Financial Management\*\*: Create an intuitive platform that simplifies the process of tracking personal finances.

2. Promote Financial Awareness\*\*: Help users gain better visibility into their spending patterns and financial habits.

3. Enable Budget Planning\*\*: Provide tools for creating and monitoring budgets across different expense categories.

4. Support Goal Setting\*\*: Allow users to set financial goals and track progress towards achieving them.

5. Deliver Actionable Insights\*\*: Generate meaningful reports and analytics that help users optimize their financial decisions.

6. Ensure Data Security\*\*: Implement robust security measures to protect sensitive financial information.

1.3 Scope

The scope of the FinTrack project encompasses:

- User authentication and account management

- Transaction recording and categorization

- Budget creation and monitoring

- Financial goal setting and tracking

- Dashboard with financial summaries and visualizations

- Detailed reporting and analytics

- Responsive design for desktop and mobile devices

- Data persistence using a relational database

1.4 Technologies Used

The FinTrack system is built using the following technologies:

Frontend:

- HTML5 for structure

- CSS3 for styling and responsive design

- AngularJS for dynamic client-side functionality

- Chart.js for data visualization

- Backend:

- Node.js as the server-side runtime environment

- Express.js for API development and routing

- MySQL for database management

- Security:

- bcrypt for password hashing

- Express-session for session management

- CORS for cross-origin resource sharing

- Development Tools:

- Git for version control

- npm for package management

Chapter 2: System Analysis and Design

2.1 Requirements Analysis

2.1.1 Functional Requirements

1. User Authentication:

1. User registration with email and password

2. Secure login and logout functionality

3. Password recovery options

2. Transaction Management:

1. Add, edit, and delete financial transactions

2. Categorize transactions (e.g., food, transport, entertainment)

3. Filter and search transactions by various criteria

3. Budget Management:

1. Create budgets for different expense categories

2. Set budget periods (monthly, quarterly, yearly)

3. Track spending against budget limits

4. Receive notifications for budget overruns

4. Goal Management:

1. Create financial goals with target amounts and deadlines

2. Track progress towards goals

3. Add contributions to goals

5. Reporting and Analytics:

1. Generate financial summaries

2. Visualize spending patterns

3. Compare income vs. expenses over time

4. Calculate savings rate

2.1.2 Non-Functional Requirements

1. Performance:

1. Fast page load times

2. Responsive user interface

3. Efficient database queries

2. Security:

1. Secure storage of user credentials

2. Protection against common web vulnerabilities

3. Session management

3. Usability:

1. Intuitive user interface

2. Responsive design for various devices

3. Clear navigation and workflow

4. Reliability:

1. Data consistency and integrity

2. Error handling and recovery

3. Regular backups

2.2 System Architecture

FinTrack follows a client-server architecture with a clear separation between the frontend and backend components:

2.2.1 Frontend Architecture

The frontend is built using the Model-View-Controller (MVC) pattern implemented through AngularJS:

-Model: Represents the data structures and business logic

-View: HTML templates that render the user interface

-Controller: JavaScript functions that handle user interactions and update the model and view

2.2.2 Backend Architecture

The backend follows a layered architecture:

-API Layer: Express.js routes that handle HTTP requests

- Service Layer: Business logic and data processing

-Data Access Layer: Database interactions using MySQL

2.2.3 System Flow

1. User interacts with the frontend interface

2. AngularJS controllers process user actions

3. HTTP requests are sent to the backend API

4. Express routes handle the requests

5. Business logic is applied

6. Database operations are performed

7. Results are returned to the frontend

8. UI is updated to reflect changes

2.3 Database Design

The FinTrack database is designed using a relational model with MySQL. The key entities and their relationships are:

2.3.1 Entity Relationship Diagram

The database consists of the following main tables:

1. user: Stores user account information

1. id (Primary Key)

2. name

3. email

4. password\_hash

5. created\_at

2. transactions: Records financial transactions

1. id (Primary Key)

2. user\_id (Foreign Key)

3. description

4. amount

5. category\_id (Foreign Key)

6. type (income/expense)

7. transaction\_date

8. created\_at

3. categories: Defines transaction categories

1. id (Primary Key)

2. name

3. created\_at

4. budgets: Stores budget information

1. id (Primary Key)

2. user\_id (Foreign Key)

3. category\_id (Foreign Key)

4. amount

5. created\_at

5. dashboard: Stores user dashboard preferences

1. id (Primary Key)

2. user\_id (Foreign Key)

3. last\_month\_change

4. created\_at

2.3.2 Data Normalization

The database is designed following normalization principles to minimize redundancy and improve data integrity:

- First Normal Form (1NF): All tables have a primary key and atomic values

- Second Normal Form (2NF): Non-key attributes depend on the entire primary key

- Third Normal Form (3NF): Non-key attributes do not depend on other non-key attributes

2.4 User Interface Design

The FinTrack user interface is designed with a focus on usability, clarity, and responsiveness. Key design principles include:

2.4.1 Layout Structure

- Landing Page: Introduces the application and its features

- Dashboard: Provides an overview of financial status

- Transactions Page: Lists and manages financial transactions

- Budgets Page: Displays and manages budget information

- Goals Page: Shows progress towards financial goals

- Reports Page: Presents detailed financial analytics

- Settings Page: Allows customization of user preferences

2.4.2 Design Elements

- Color Scheme: Professional and calming colors with clear contrast

- Typography: Readable fonts with appropriate sizing hierarchy

- Icons: Intuitive icons to represent different functions

- Charts and Graphs: Visual representations of financial data

- Forms: Clean and structured input forms with validation

- Responsive Design: Adapts to different screen sizes and devices

Chapter 3: Implementation and Features

3.1 Frontend Implementation

3.1.1 AngularJS Framework

The frontend is implemented using AngularJS, which provides a structured framework for building dynamic web applications. Key aspects of the implementation include:

- Module Structure: The application is organized into modules for better maintainability

- Controllers: Handle user interactions and manage the application state

- Directives: Extend HTML with custom attributes and elements

- Services: Share data and functionality across the application

- Filters: Format data for display

3.1.2 User Interface Components

1. Navigation System:

1. Sidebar navigation for desktop

2. Collapsible menu for mobile devices

3. Breadcrumb navigation for complex workflows

2. Dashboard Widgets:

1. Financial summary cards

2. Recent transactions list

3. Budget progress indicators

4. Goal tracking visualizations

3. Forms and Modal:

1. Transaction entry form

2. Budget creation form

3. Goal setting form

4. Settings configuration panels

4.Data Visualization:

1. Spending charts by category

2. Income vs. expenses comparison

3. Budget utilization graphs

4. Goal progress indicators

3.1.3 Responsive Design

The application is designed to be fully responsive, adapting to different screen sizes and devices:

- Fluid grid layout

- Flexible images and media

- Media queries for different breakpoints

- Touch-friendly interface elements

3.2 Backend Implementation

3.2.1 Node.js and Express

The backend is built using Node.js with Express.js framework, providing a robust foundation for the API:

- Server Configuration: Setup of the Express application with necessary middleware

- Routing: Definition of API endpoints for different resources

- Middleware: Implementation of authentication, error handling, and request processing

- Session Management: User session handling with express-session

3.2.2 Database Integration

The application integrates with MySQL database using the mysql2 library:

- Connection Pool: Efficient management of database connections

- Query Execution: Parameterized queries to prevent SQL injection

- Transaction Support: Ensuring data consistency for complex operations

- Error Handling: Graceful handling of database errors

3.2.3 Authentication System

User authentication is implemented with security best practices:

- Password Hashing: Using bcrypt for secure password storage

- Session-based Authentication: Managing user sessions securely

- Access Control: Protecting routes with authentication middleware

- CSRF Protection: Preventing cross-site request forgery attacks

3.3 Key Features

3.3.1 Transaction Management

The transaction management system allows users to:

- Record income and expenses

- Categorize transactions

- Add notes and details

- Filter and search transactions

- View transaction history

Implementation details:

```javascript

// Add transaction function

app.post("/api/transactions/add", isAuthenticated, async (req, res) => {

try {

const { description, amount, category, type, date, notes } = req.body

if (!description || !amount || !category || !type || !date) {

return res.status(400).send("Missing required fields")

}

const connection = await pool.getConnection()

// Get category ID

const [categories] = await connection.execute("SELECT id FROM categories WHERE name = ?", [category])

if (categories.length === 0) {

connection.release()

return res.status(400).send("Invalid category")

}

const categoryId = categories[0].id

// Insert transaction

const [result] = await connection.execute(

"INSERT INTO transactions (user\_id, description, amount, category\_id, type, transaction\_date) VALUES (?, ?, ?, ?, ?, ?)",

[req.session.userId, description, amount, categoryId, type, date],

)

connection.release()

res.status(201).send("success")

} catch (error) {

console.error("Error creating transaction:", error)

res.status(500).send("Server error")

}

})

```

#### 3.3.2 Budget Management

The budget management system enables users to:

- Create budgets for different categories

- Set budget limits

- Track spending against budgets

- Visualize budget utilization

Implementation details:

```javascript

// Update budget spending based on transactions

$scope.updateBudgetSpending = () => {

// Reset all spent amounts

$scope.budgets.forEach((budget) => {

budget.spent = 0

})

// Calculate spent amount for each budget

$scope.allTransactions.forEach((transaction) => {

if (transaction.type === "expense") {

const category = transaction.category

const budget = $scope.budgets.find((b) => b.category.toLowerCase() === category.toLowerCase())

if (budget) {

budget.spent += Number.parseFloat(transaction.amount)

}

}

})

}

```

3.3.3 Goal Tracking

The goal tracking feature allows users to:

- Set financial goals with target amounts

- Track progress towards goals

- Add contributions to goals

- Visualize goal completion percentage

3.3.4 Reporting and Analytics

The reporting system provides:

- Financial summaries

- Spending breakdowns by category

- Income vs. expenses comparisons

- Savings rate calculations

- Exportable reports

3.4 Security Considerations

3.4.1 Data Protection

- Password Security: Passwords are hashed using bcrypt before storage

- Input Validation: All user inputs are validated to prevent injection attacks

- Parameterized Queries: SQL queries use parameters to prevent SQL injection

3.4.2 Authentication and Authorization

- Session Management: Secure session handling with express-session

- Route Protection: Authentication middleware for protected routes

- CORS Configuration: Proper configuration to prevent unauthorized cross-origin requests

3.4.3 Error Handling

- Graceful Error Handling: Proper handling of errors to prevent information leakage

- Logging: Comprehensive logging for monitoring and debugging

- User Feedback: Appropriate error messages for users without revealing system details

Chapter 4: Conclusion and Future Scope

4.1 Summary of Achievements

The FinTrack project has successfully delivered a comprehensive personal finance management system with the following achievements:

1. Functional Implementation: All core features have been implemented, including transaction management, budget planning, goal tracking, and reporting.

2. User-Friendly Interface: The application provides an intuitive and responsive user interface that works across different devices.

3. Secure Architecture: The system implements security best practices for data protection and user authentication.

4. Performance Optimization: Database queries and frontend rendering have been optimized for performance.

5. Modular Design: The codebase follows a modular design that facilitates maintenance and future enhancements.

4.2 Limitations

Despite the successful implementation, the current version of FinTrack has some limitations:

1. Limited Integration: The system does not integrate with external financial services or banks for automatic transaction import.

2. Basic Reporting: While functional, the reporting capabilities could be more comprehensive with additional analytics.

3. Mobile Experience: Although responsive, a dedicated mobile application would provide a better experience on smartphones.

4. Offline Functionality: The application requires an internet connection to function properly.

5. Limited Customization: Users have limited options to customize categories and reporting preferences.

4.3 Future Enhancements

Based on the current limitations and potential user needs, the following enhancements are proposed for future versions:

1. Bank Integration: Implement APIs to connect with banking services for automatic transaction import and reconciliation.

2. Advanced Analytics: Enhance reporting capabilities with predictive analytics, trend analysis, and personalized recommendations.

3. Mobile Application: Develop native mobile applications for iOS and Android platforms.

4. Offline Mode: Implement offline functionality using service workers and local storage.

5. Multi-Currency Support: Add support for multiple currencies and exchange rate conversions.

6. Collaborative Features: Enable sharing of financial information with family members or financial advisors.

7. Investment Tracking: Add features to track investments, stocks, and retirement accounts.

8. Tax Planning: Implement tax planning and estimation features.

9. Machine Learning Integration: Use machine learning for automatic categorization of transactions and spending pattern analysis.

10.Notification System: Enhance the notification system with email alerts, push notifications, and scheduled reminders.

4.4 Conclusion

The FinTrack Personal Finance Management System represents a significant step towards helping individuals gain better control over their financial lives. By providing tools for tracking expenses, planning budgets, setting goals, and analyzing financial patterns, the application empowers users to make informed financial decisions.

The project demonstrates the effective application of modern web development technologies, including AngularJS, Node.js, Express, and MySQL, to create a secure, responsive, and user-friendly financial management solution. The modular architecture and clean code structure provide a solid foundation for future enhancements and extensions.

As financial management continues to be a critical aspect of personal well-being, tools like FinTrack play an important role in promoting financial literacy and responsible financial behavior. With the proposed future enhancements, the system has the potential to evolve into a comprehensive financial wellness platform that addresses a wide range of user needs and preferences.

In conclusion, the FinTrack project has successfully achieved its objectives of simplifying financial management, promoting financial awareness, and providing actionable insights to users. The system stands as a testament to the power of modern web technologies in creating practical solutions for everyday challenges.

---

References

1. Node.js Documentation. (2023). Retrieved from [https://nodejs.org/en/docs/](https://nodejs.org/en/docs/)

2. Express.js Documentation. (2023). Retrieved from [https://expressjs.com/](https://expressjs.com/)

3. AngularJS Documentation. (2023). Retrieved from [https://docs.angularjs.org/](https://docs.angularjs.org/)

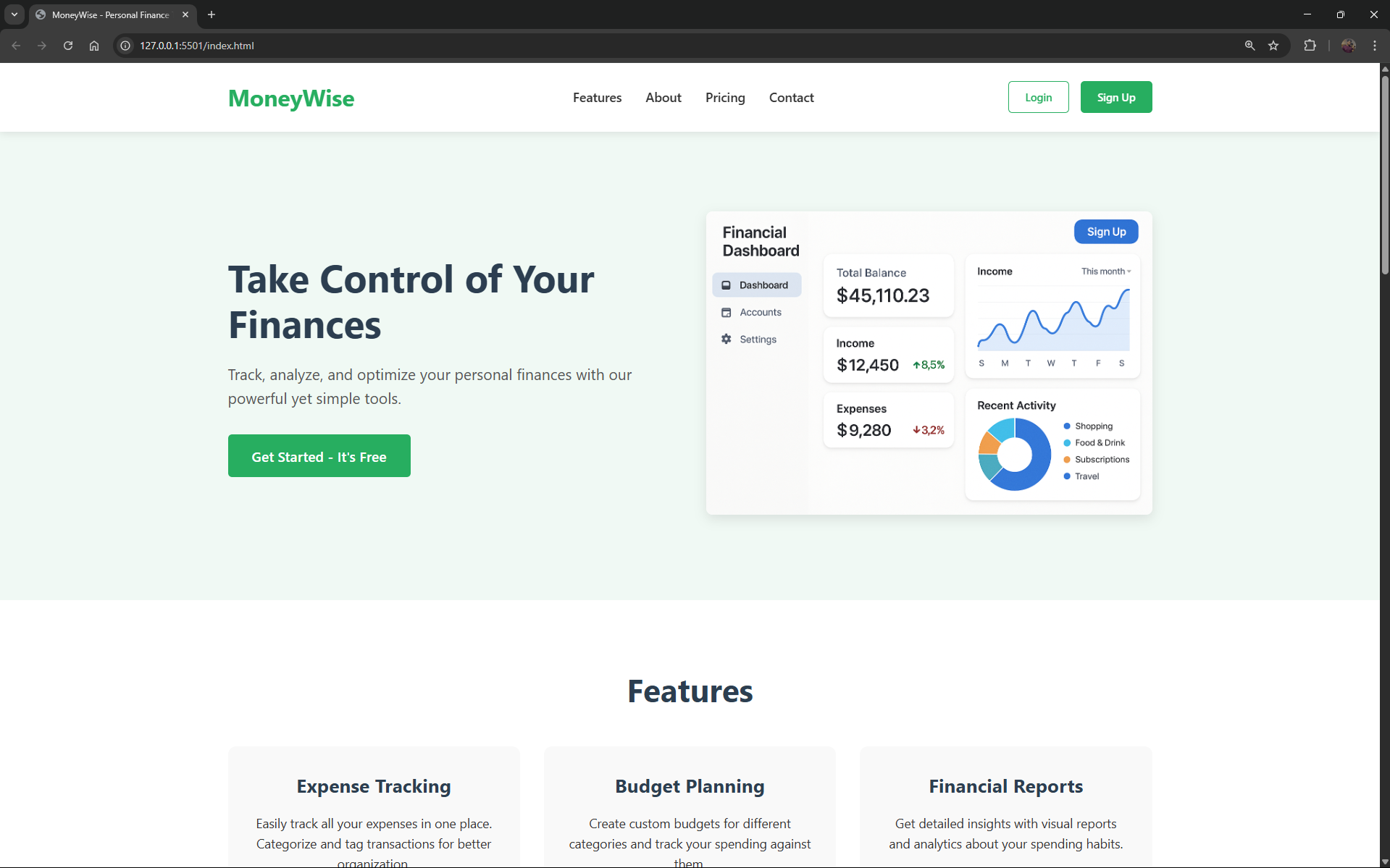
4. MySQL Documentation. (2023). Retrieved from [https://dev.mysql.com/doc/](https://dev.mysql.com/doc/)

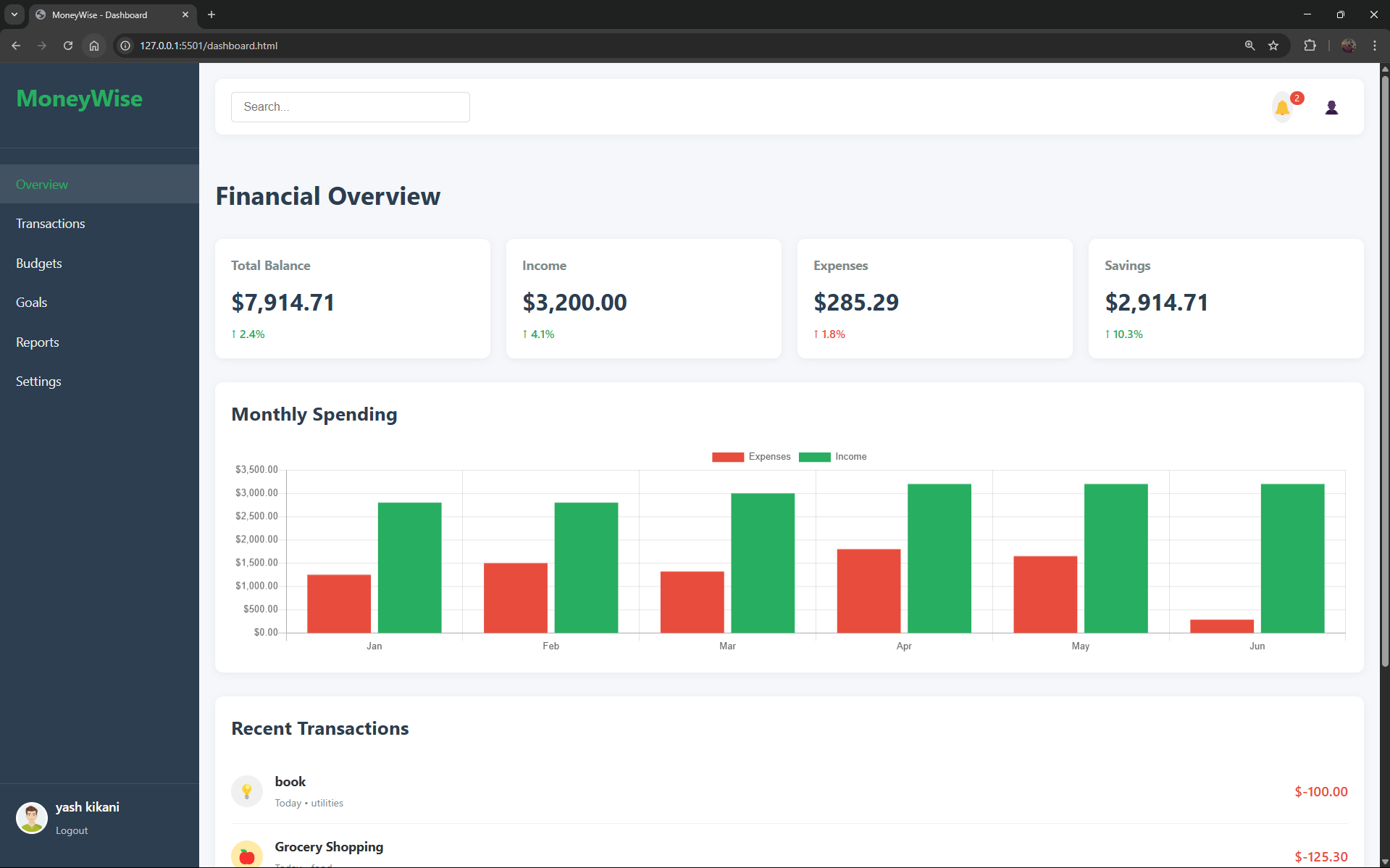
5. Chart.js Documentation. (2023). Retrieved from [https://www.chartjs.org/docs/latest/](https://www.chartjs.org/docs/latest/)

6. Bcrypt Documentation. (2023). Retrieved from [https://github.com/kelektiv/node.bcrypt.js](https://github.com/kelektiv/node.bcrypt.js)

7. Express-session Documentation. (2023). Retrieved from [https://github.com/expressjs/session](https://github.com/expressjs/session)

8. CORS Documentation. (2023). Retrieved from [https://github.com/expressjs/cors](https://github.com/expressjs/cors)





FILES LINK-  
https://drive.google.com/drive/folders/11mLD9rkxK0i9PPl0su6WipXhiGnFg\_tU?usp=sharing