

A
PROJECT REPORT
ON
SmartSpend: Personal Finance Manager

Submitted in partial fulfilment of the requirements
of the degree of

Bachelor of Engineering

In

Information Technology

by

Nandini Nichite - (Roll No. 34)

Sairaj Pai - (Roll No. 35)

Yash Patil - (Roll No. 43)

Supervisor:

Asst. Prof. Rachana Borole



Department of Information Technology

K.C. College of Engineering and Management Studies And

Research, Thane (E)

University of Mumbai

2024-25

CERTIFICATE

This is to certify that the project entitled "**SmartSpend: Personal Finance Manager**" is a bonafide work of "**Nandini Nichite - 34, Sairaj Pai - 35, Yash Patil - 43**" submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of "**Bachelor of Engineering**" in "**Information Technology**".

Asst. Prof. Rachana Borole
Supervisor/Guide



Prof.Amarja Adgaonkar
Head of Department

Dr.Vilas Nitnaware
Principal

Project Report Approval for T.E.

This project report entitled ***Book Recommender System*** by ***Yash Patil, Sairaj Pai, Nandini Nichite*** is approved for the degree of Bachelor of Engineering in **Information Technology**.

Examiners

1.-----

2.-----

Date:

Place: Thane

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Nandini Nichite - 34

Sairaj Pai -35

Yash Patil - 43

Date:

ACKNOWLEDGEMENT

We would like to express special thanks of gratitude to our guide **Asst Prof. Mrs. Rachana Borole** as well as our Project Coordinator **Asst Prof. Mrs. Priyanka Sananse** gave us the golden opportunity to do this wonderful project on the topic of **Book Recommender System.**, which also helped us in doing a lot of research and we came to know about so many new things. We are very grateful to our Head of the Department **Mrs.Amarja Adgaonkar** for extending her help directly and indirectly through various channels in our project work. We would also like to thank Principal **Dr. Vilas Nitnaware** for providing us the opportunity to implement our project. We are really thankful to them. Finally we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

Thanking You.

TABLE OF CONTENT

Sr. No.	Topic	Page No.
1	Certificate.....	1
2	Project Approval Sheet.....	2
3	Declaration.....	3
4	Acknowledgement	4
5	Table of Content.....	5
6	Abstract.....	8
7	1. Introduction	
	1.1 Introduction	9
	1.2 Motivation	9
	1.3 Problem Statement	10
	1.4 Organization of Report	11
	1.5 Abbreviations	12
8	2. Literature Survey	13
	2.1 Survey of Existing System	17
	2.2 Limitations of Existing System	18
	2.3 Mini Project Implementation Plan	19
	2.4 Advantages	20
	2.5 Disadvantages	20
9	3. Proposed System	21
	3.1 Introduction	22
	3.2 Architecture/Framework	23
	3.3 Flowchart & Algorithm	27
	3.4 Operation Environment	29
	3.5 Experiment and Results	30

	3.6 Conclusion	39
11	Future Scope.....	40
12	References.....	41

LIST OF FIGURES

Figure no	Name of Figure	Page No
3.2.1	MindMap of Technologies Used	23
3.3.1	FrontEnd Flowchart	27
3.3.2	Backend Flowchart	28
3.3.3	Database Schema	29
3.5.1	DataFlow Diagram (DFD)	30
3.5.2	Backend Code Snippet	32
3.5.3	Frontend Code Snippet	32
3.5.4	Database Model Code Snippet	33
3.5.5	API Call Code Snippet	33
3.5.6	Frontend Router Code Snippet	34
3.5.7	HomePage Code Snippet	34
3.5.8	Swagger UI API Docs Snippet	35
3.5.9	API Call Snippet	35
3.5.10	Landing Page Snippet	36
3.5.11	SignUp Page Snippet	36
3.5.12	Dashboard Snippet	37

3.5.13	User Balance Info Snippet	37
3.5.14	Investments Snippet	38
3.5.15	Expenses Snippet	38
3.5.16	Account Settings Snippet	39

LIST OF TABLES

Table No.	Name of Table	Page No
1	Literature Survey	13

ABSTRACT

This project is a comprehensive personal finance planner developed using the MERN stack (MongoDB, Express, React, and Node.js). It provides users with the ability to manage their financial activities such as investments, expenses, transactions, loans, and EMIs, all within an intuitive and secure platform. The front-end, designed with React and styled with HTML and CSS, incorporates React Router DOM for seamless navigation, while Axios is used for efficient data fetching and session management. The back-end, powered by Express, employs RESTful APIs to interact with a MongoDB database via Mongoose for efficient data handling.

Authentication is securely implemented using JWT and bearer tokens, stored in HTTP-only cookies. The platform offers users insightful data visualization through Chart.js, presenting an intuitive dashboard with detailed financial graphs. OpenAPI v3/Swagger UI is integrated for comprehensive API documentation, and Postman is used to test and verify the API endpoints. Additionally, the design was prototyped in Figma, and icons for a polished user experience were sourced from IconMonstr and Icons8.

This project effectively combines modern web development technologies and design principles to deliver a powerful tool for personal finance management.

1. INTRODUCTION

1.1 INTRODUCTION

In today's digital era, the vast availability of books has made it increasingly challenging for readers to discover the right books that match their interests. A Book Recommender System addresses this issue by providing personalized suggestions based on user preferences, ratings, and reviews. These systems leverage data analysis and machine learning techniques to identify patterns in user behavior and recommend books that align with their reading habits.

This project utilizes a dataset containing information about books, users, and their ratings to build an efficient recommendation system. The system applies data filtering techniques to analyze user interactions and generate meaningful recommendations. By considering key factors such as the number of reviews and average ratings, the model highlights the most popular and highly-rated books. This ensures that users receive recommendations that are both relevant and well-regarded by the reading community..

The goal of this project is to create a data-driven book recommendation system that enhances the user experience by making book discovery more engaging and effortless. Whether a reader is looking for bestsellers or hidden literary gems, this system aims to provide accurate and insightful recommendations based on real user feedback.

1.2 MOTIVATION

Overwhelming Book Choices – With millions of books available worldwide, readers often find it difficult to choose what to read next. A book recommender system helps users navigate this vast collection by providing personalized suggestions based on their interests and past preferences.

Enhancing User Experience – Traditional methods of book discovery, such as browsing through categories or relying on bestseller lists, can be time-consuming and inefficient. A recommendation system improves user engagement by offering tailored suggestions that align with individual reading habits.

Leveraging Data for Insights – The availability of large datasets containing book ratings and reviews allows us to analyze user behavior effectively. By identifying patterns in reading preferences, the system can provide more accurate and meaningful recommendations.

Encouraging Reading Habits – Many readers struggle to maintain a consistent reading habit due to a lack of motivation or difficulty in finding books they enjoy. A recommender system helps overcome this challenge by introducing users to books that match their tastes, making reading more enjoyable and accessible.

Application of Machine Learning – The development of a book recommender system provides a valuable opportunity to apply machine learning and data filtering techniques. By implementing collaborative filtering, content-based filtering, or hybrid approaches, we can improve the accuracy and efficiency of recommendations.

Real-World Impact – Major online platforms like Amazon, Goodreads, and Google Books use similar recommendation systems to enhance user experience and increase engagement. This project serves as a practical implementation of such a system, demonstrating its importance in the modern digital landscape.

1.3 PROBLEM STATEMENT

- Finding the right book is challenging due to the vast number of options. This project builds a Book Recommender System that suggests books based on user ratings and reviews, ensuring better and more personalized recommendations.

OBJECTIVES

Develop an Intelligent Book Recommender System – Build a system that provides personalized book recommendations to users based on their reading preferences, ratings, and reviews. By analyzing user interactions, the system will help readers discover books that match their interests without the hassle of manual searching.

Filter Books with More Than 50 Reviews – Ensure that only books with significant user engagement are considered for recommendations. This helps in avoiding books with limited or unreliable reviews and instead prioritizes books that have been widely read and rated by multiple users, making the recommendations more trustworthy.

Analyze User Ratings and Preferences – Study the reading behavior of users by analyzing book ratings, reviews, and preferences. Identifying patterns in user interactions will help improve recommendation accuracy and ensure that the suggested books align with their tastes.

Implement Advanced Recommendation Techniques – Utilize different recommendation techniques such as collaborative filtering, content-based filtering, and hybrid models to provide more accurate and efficient book suggestions. By leveraging machine learning models, the system will be able to learn from user behavior and make better predictions over time.

Enhance User Experience and Engagement – Create an intuitive and user-friendly system that makes book discovery faster, easier, and more enjoyable. Instead of users spending hours searching for the right book, the recommender system will instantly suggest books that are highly rated and relevant to their interests.

Apply Machine Learning and Data Science Methods – Use real-world datasets containing information about books, users, and ratings to develop an effective recommendation system. This project will demonstrate the practical application of data science, data preprocessing, feature engineering, and machine learning algorithms in solving real-world problems.

Improve the Accuracy of Recommendations – Continuously refine the recommendation algorithm by testing different models, optimizing parameters, and validating results against user feedback. Ensuring high accuracy will increase user trust in the system and make it more effective in providing relevant book suggestions.

1.4 ORGANIZATION OF REPORT

The report is divided into 3 parts explaining the project.

- The first chapter gives the introduction of the project as well as our motivation behind selecting the project as well as the problem statement of the project.
- The second chapter gives the overview on the literature survey conducted by the team.
- Survey of existing system, limitations of literature survey and the implementation plan of the project are also explained in the chapter
- Final chapter explains the framework on which the project is made as well as shows outputs of our project.
- Finally, we have the page elaborating the conclusion and future scope of the project.

1.5 ABBREVIATIONS

- **ISBN** (International Standard Book Number) – A unique identifier assigned to books, helping to track and differentiate them in databases. It is used to merge book data from multiple sources.
- **ML** (Machine Learning) – A branch of AI that allows computers to learn patterns from user ratings, reviews, and preferences to improve book recommendations.
- **AI** (Artificial Intelligence) – The technology used to enhance recommendation systems by making them smarter and more adaptive to user behavior.
- **CF** (Collaborative Filtering) – A recommendation technique that suggests books based on what similar users have liked or rated highly, without considering the content of the books.
- **CBF** (Content-Based Filtering) – A recommendation approach that suggests books by analyzing their content, such as genre, author, and descriptions, and matching them with user preferences.
- **UI/UX** – User Interface / User Experience (Improves user interaction with the system)
- **API** – Application Programming Interface (Used to fetch data or integrate external services)
- **CSV** – Comma-Separated Values (Format for storing book, user, and rating data)
- **NLP** – Natural Language Processing (Helps analyze book descriptions or reviews for better recommendations)
- **SVD** – Singular Value Decomposition (A matrix factorization technique for recommendations)

2. LITERATURE SURVEY

Year	Title of the paper	Methodology Used	Advantages	Limitations and challenges
2023	Comparison Of Android-Based Personal Financial Management Applications With Variative Financial Conditions by L.Dewi	<p>Financial planning puts individuals in a better position to understand, manage and make financial decisions. Currently, there are various Android-based applications available to help manage personal finances. This study aims to make a comparison of Android-based personal financial management applications. Recording of 12 types of financial transactions is carried out using three financial management applications, namely Money Lover, Keuangan Pribadi Ku, and Catatan Keuangan Harian. Revenue, expenditure, and realization budget data have surplus, balanced, and deficit conditions prepared for input and processing in the three applications, with a comparison using ten parameters. The results of this study indicate that the three applications have standard</p>	<p>Enhanced Financial Awareness: The study highlights that personal finance management applications like Money Lover and Catatan Keuangan Harian help users manage their finances effectively, increasing awareness of their financial health.</p> <p>Variety of Features: All three apps offer a standard set of features to assist users in managing personal finances, allowing for flexibility in handling different types of transactions.</p> <p>Support for Various Financial Conditions: Both Money Lover and Catatan Keuangan</p>	<p>Limited Deficit Handling in Keuangan Pribadi Ku: The study reveals that the Keuangan Pribadi Ku app lacks features for managing financial records in deficit conditions, limiting its utility for users facing such situations.</p> <p>Inconsistency Across Platforms: The apps are Android-based, which may exclude users on other platforms like iOS, reducing accessibility.</p> <p>Feature Gaps: While the</p>

		<p>features that can be used as tools in personal financial management. However, the Keuangan Pribadi Ku application does not provide features to handle recording when the financial condition is in a deficit and can only run well in balanced and surplus financial conditions. Meanwhile, the Money Lover dan Catatan Keuangan Harian provides features to record surplus, balanced, and deficit financial conditions. This research has contributed to increasing understanding and awareness of the importance of utilizing supporting technology in personal financial management and providing consideration for users in choosing a personal finance management application that suits their needs.</p>	<p>Harian are capable of recording financial transactions under surplus, balanced, and deficit conditions, making them more versatile for comprehensive financial tracking.</p>	<p>applications provide basic tools for personal finance management, there may be gaps in advanced financial planning features or integrations with external financial services.</p>
2018	Development of Spreadsheet-Based Applications for Learning of Financial Management	<p>Spreadsheet based applications are needed by students and lecturers to expedite the financial management learning process. The aim of the research was to produce a spreadsheet-based</p>	<p>Expedited Learning Process: The spreadsheet-based application accelerates the learning process for students and lecturers</p>	<p>Technological Familiarity: Users need to be proficient with spreadsheet tools like Excel, which</p>

	I. Jaya, N. Masih, N. Wahyuni, I. Sugiarta	<p>application for financial management that can be used for real learning. Spreadsheet-based application for financial management includes cash, accounts receivable, inventory, fixed assets, liabilities, equity, financial ratios, and time value of money management. his study uses a qualitative approach with descriptive methods. The research model used was adapted from the research and development model with stages: 1) analysis of the needs of teaching materials; 2) development of a draft spreadsheet-based application for financial management; 3) testing draft spreadsheet-based applications for financial management. The results of the study is a spreadsheet-based application for financial management can be used in real learning. The test results show that spreadsheet-based applications have given the same results as the manual calculation results.</p>	<p>by simplifying and streamlining financial management tasks such as cash flow, accounts receivable, and inventory management.</p> <p>Real Learning</p> <p>Application: The study shows that the spreadsheet-based tool can be effectively used in practical, real-world learning scenarios, enhancing the understanding of financial concepts.</p> <p>Accurate</p> <p>Calculations: The application has been proven to deliver the same results as manual calculations, ensuring accuracy and reliability in teaching and learning financial management.</p> <p>Comprehensive</p> <p>Financial</p> <p>Management: It covers key financial areas like fixed assets, liabilities,</p>	<p>may pose a challenge for those with limited experience in this area.</p> <p>Software Compatibility: Depending on the version of spreadsheet software available (e.g., Excel or Google Sheets), compatibility issues or differences in features may arise, affecting the usability of the application.</p> <p>Scalability for Larger Data Sets: Spreadsheet-based systems may struggle with handling larger or more complex datasets compared to dedicated financial management</p>
--	--	---	--	---

			equity, financial ratios, and time value of money, providing a holistic tool for financial education.	software, potentially limiting its use for larger-scale educational projects.
2015	Online Personal Finance Management Applications Viera Gáfriková, W. Szczesny, Z. Odrzygóźdż · 2015	The paper is devoted to internet applications supporting personal finance management (PFM). The main aim of the paper is to describe presumptions for development of PFM applications and to compare – from the user's point of view – selected PFM websites in Polish, taking into consideration two categories: credibility and functionality. Within each category several factors were considered. The factors' assessment has resulted in the creation of websites positioning.	Credibility and Trustworthiness: The study highlights the importance of credibility in personal finance management (PFM) websites, ensuring that users can trust the platforms with their sensitive financial information. Functionality: The research focuses on comparing the functionality of various PFM websites, indicating that well-developed applications offer a range of useful features for managing personal finances efficiently.	Limited Scope of Study: The comparison focuses on Polish PFM websites, which may limit its relevance or applicability to users in other countries or regions with different financial needs and systems. Variability in Functionality: The study suggests that the functionality of PFM websites can vary greatly, potentially leading to inconsistencies in the quality of

		<p>User-Centered Development: The paper outlines key presumptions for the development of PFM applications, emphasizing the need to consider user needs and experiences, which leads to more user-friendly and effective financial tools.</p>	<p>financial management support available to users.</p> <p>Potential Gaps in Features: Despite comparing websites based on functionality, the study may imply that not all PFM applications provide comprehensive features for advanced personal finance management, leaving room for improvement in certain platforms.</p>
--	--	---	--

2.1 SURVEY OF EXISTING SYSTEM

While surveying existing personal finance management systems, the following key aspects were examined:

- **Traditional Book Recommendation Methods** – Readers often rely on bestseller lists, word-of-mouth, or manual searches on platforms like Amazon and Goodreads, which can be time-consuming and may not always align with personal preferences.
- **Rule-Based Recommendation Systems** – Some early systems used predefined rules, such as recommending books from the same genre or by the same author. However, these systems lack personalization and fail to adapt to individual user preferences.
- **Collaborative Filtering-Based Systems** – Many modern recommendation systems, such as those used by Goodreads and Amazon, utilize collaborative filtering. These systems analyze user behavior and suggest books based on similar readers' preferences. However, they suffer from the cold start problem (lack of recommendations for new users or books).
- **Content-Based Filtering Approaches** – Platforms like Google Books sometimes use content-based filtering, where books are recommended based on their metadata (genre, author, summary). However, this method can be limited in diversity, as it only suggests books similar to what the user has already read.
- **Hybrid Recommendation Systems** – Some advanced systems, like those used by Netflix-style book recommendation platforms, combine collaborative and content-based filtering to improve accuracy. These systems provide better recommendations but can be computationally expensive and require large datasets.
- **Machine Learning & Deep Learning Models** – Recent advancements have introduced AI-powered recommendation engines that analyze complex user interactions, review sentiments, and even reading patterns. While effective, these methods require significant computational resources and large amounts of data.
- **Online Bookstore Recommenders** – E-commerce platforms like Amazon and Flipkart use purchase history, user ratings, and browsing behavior to recommend books. However, these systems are often biased toward promoting best-selling or sponsored books rather than genuinely personalized suggestions.

2.2 LIMITATIONS OF EXISTING SYSTEM

1. **Lack of Personalization** – Traditional recommendation methods, such as bestseller lists and manual searches, fail to cater to individual user preferences. These systems do not analyze user interests, resulting in generic suggestions that may not align with a reader's taste.
2. **Cold Start Problem** – Many collaborative filtering-based systems struggle with new users who have not provided enough ratings or interactions. Similarly, newly published books with few reviews often go unnoticed because the system lacks enough data to recommend them effectively.
3. **Over-Specialization in Content-Based Filtering** – Content-based filtering tends to recommend books that are very similar to what a user has already read. This lack of

diversity prevents users from discovering books outside their comfort zone, limiting their exposure to new genres and authors.

4. **High Computational Complexity** – AI-driven recommendation systems, especially those using deep learning and hybrid models, require significant computational power and vast datasets. This makes them difficult to deploy for smaller platforms or personal recommendation engines with limited resources.
5. **Data Sparsity** – The number of users actively rating or reviewing books is often low compared to the total number of books available. Sparse datasets make it difficult for machine learning models to generate relevant recommendations, leading to lower accuracy and effectiveness.
6. **Popularity Bias** – Many recommendation systems prioritize books that are already highly rated or widely read, often promoting bestsellers or trending books. This makes it harder for less popular but high-quality books to gain visibility, restricting diversity in recommendations.
7. **Influence of Fake Reviews & Biased Ratings** – Online book recommendation platforms often suffer from spam reviews, bot-generated ratings, and biased feedback. Books with artificially inflated ratings may be recommended frequently, leading to misleading suggestions for users.
8. **Limited Adaptability to Changing Preferences** – Many existing systems do not dynamically update recommendations based on a user's evolving interests. If a user starts exploring a new genre, the system may still suggest books based on past preferences rather than adapting in real-time.

2.3 MINI PROJECT IMPLEMENTATION PLAN

The *SmartSpend* project was designed to address the shortcomings identified in the survey of existing systems. The implementation plan focused on creating a secure, scalable, and user-friendly platform for comprehensive personal finance management.

Key Improvements:

- **Automated Data Tracking:** Integrate APIs for automatic expense and transaction tracking, minimizing the need for manual input.
- **Enhanced Security:** Use JWT for secure authentication, along with secure cookies for session management, ensuring user data is protected at all times.
- **Comprehensive Financial Overview:** Provide tools to track loans, EMIs, investments, and expenses in a unified dashboard for a complete financial picture.
- **Intuitive UI:** Develop a clean, intuitive user interface with React, ensuring ease of navigation and use for all types of users.

- **Scalability and Performance:** Design a scalable backend with MongoDB and Express to handle a growing user base and large financial datasets, ensuring optimal performance.
- **Detailed Documentation:** Provide clear documentation for building, deploying, and scaling the application, ensuring developers can easily contribute to and extend the platform.

Team Roles:

- **Yash Patil:** Database schema design, frontend and backend development, encryption, and deployment.
- **Sairaj Pai:** Frontend design, backend development, form validation, and quality assurance.
- **Nandini Nichite:** Documentation, security features, testing, and frontend validation.

2.4 ADVANTAGES

1. **Personalized Book Suggestions** – Unlike traditional methods like bestseller lists, a recommendation system tailors book suggestions based on individual user preferences, making it easier to discover books that match their interests.
2. **Efficient and Time-Saving** – Instead of manually searching through thousands of books, users receive relevant recommendations instantly, reducing the time spent looking for the right book.
3. **Advanced Machine Learning for Accuracy** – AI-driven recommendation models, such as collaborative filtering and content-based filtering, analyze user behavior to suggest books with high accuracy, improving the overall reading experience.
4. **Continuous Learning and Adaptability** – The system improves over time by learning from user ratings, reviews, and interactions, ensuring that recommendations remain relevant as user preferences evolve.
5. **Enhanced Discoverability for Lesser-Known Books** – Unlike traditional recommendation methods that focus on bestsellers, the system can introduce users to hidden gems and niche books, helping authors gain recognition.
6. **Hybrid Models for Better Recommendations** – Combining collaborative filtering and content-based filtering allows for a more diverse and balanced set of recommendations, catering to different types of readers.
7. **Increases User Engagement** – A well-designed book recommender system enhances user engagement by encouraging readers to explore new books, leave reviews, and interact with the platform.

8. **Multi-Language and Genre Support** – Modern systems can recommend books across different languages and genres, making them accessible to a wider audience and promoting cultural diversity in reading.

2.5 DISADVANTAGES

1. **Cold Start Problem** – New users and books often lack enough interaction data (ratings or reviews), making it difficult for the system to generate meaningful recommendations for them.
2. **Data Sparsity Issue** – Many users do not actively rate or review books, resulting in a lack of sufficient data for the system to make accurate recommendations, which can lower the quality of suggestions.
3. **Over-Specialization in Content-Based Filtering** – Content-based filtering tends to recommend books that are very similar to what a user has already read, reducing diversity and limiting the exploration of new genres and authors.
4. **Popularity Bias** – The system may prioritize highly-rated or frequently reviewed books, making it difficult for lesser-known but high-quality books to gain visibility. This can create an imbalance in recommendations.
5. **High Computational Requirements** – Advanced AI-based recommendation models require large datasets, extensive processing power, and complex algorithms, which can be resource-intensive and expensive to maintain.
6. **Potential for Fake Reviews and Manipulated Ratings** – Some online platforms suffer from spam reviews, fake ratings, or promotional content, which can distort recommendations and mislead users.
7. **Limited Adaptability to Changing Interests** – Some systems fail to quickly adapt when a user's reading preferences change, continuing to recommend books based on outdated behavior instead of new interests.
8. **Language and Regional Limitations** – Many recommendation systems are primarily trained on English-language books, making it challenging for non-English readers to receive quality recommendations tailored to their language or culture.
9. **Privacy Concerns** – Collecting and analyzing user data for recommendations may raise privacy and security issues, as users might be uncomfortable sharing their reading habits and preferences.
10. **Lack of Human Intuition** – Unlike human recommendations, an AI-driven system lacks emotional intelligence and contextual understanding, which sometimes leads to inappropriate or less meaningful book suggestions.

3. PROPOSED SYSTEM

3.1 INTRODUCTION

In today's digital era, the vast availability of books has made it increasingly challenging for readers to discover the right books that match their interests. A Book Recommender System addresses this issue by providing personalized suggestions based on user preferences, ratings, and reviews. These systems leverage data analysis and machine learning techniques to identify patterns in user behavior and recommend books that align with their reading habits.

This project utilizes a dataset containing information about books, users, and their ratings to build an efficient recommendation system. The system applies data filtering techniques to analyze user interactions and generate meaningful recommendations. By considering key factors such as the number of reviews and average ratings, the model highlights the most popular and highly-rated books. This ensures that users receive recommendations that are both relevant and well-regarded by the reading community.

The goal of this project is to create a data-driven book recommendation system that enhances the user experience by making book discovery more engaging and effortless. Whether a reader is looking for bestsellers or hidden literary gems, this system aims to provide accurate and insightful recommendations based on real user feedback.

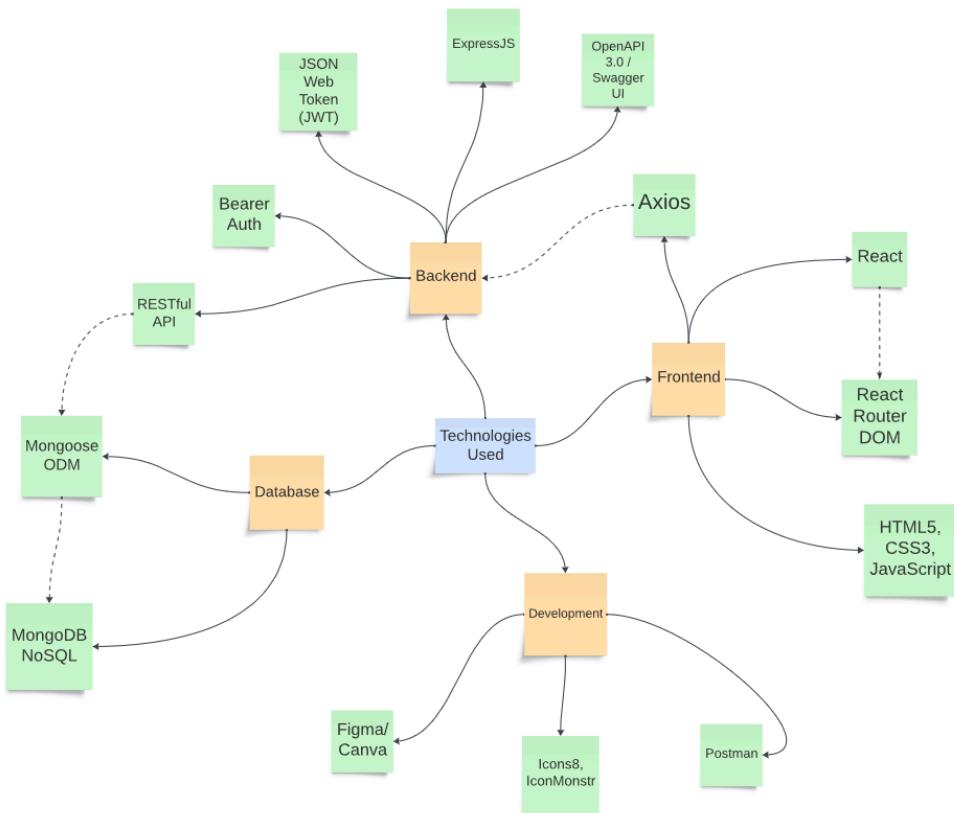
Key Features of Book Recommender System :

1. **Personalized Book Suggestions** – Uses user preferences, reading history, and ratings to recommend books tailored to individual interests.
2. **Search and Filter Options** – Allows users to search books by title, author, genre, publication year, or language, making it easy to find specific books.
3. **AI-Powered Recommendation Engine** – Utilizes machine learning algorithms such as collaborative filtering, content-based filtering, and hybrid models to improve accuracy in recommendations.
4. **User Ratings & Reviews Integration** – Considers user ratings and reviews to enhance recommendations and provide a reliable assessment of book popularity and quality.
5. **Adaptive Learning System** – Continuously updates recommendations based on changing user behavior and new book additions to ensure up-to-date suggestions.
6. **Cold Start Handling Mechanism** – Implements strategies like popularity-based recommendations or new arrival suggestions to assist new users and books with limited data.
7. **Popular & Trending Books Section** – Displays bestsellers, trending books, and top-rated recommendations to help users explore books with high engagement.

8. **User-Friendly UI/UX** – Provides an interactive and intuitive interface with features like book previews, summaries, author details, and cover images for a seamless browsing experience.

3.2 ARCHITECTURE/FRAMEWORK

Comprehensive Mind Map of Technologies used to build the project:



3.2.1 MindMap Of Technologies Used

Backend:

1. **Technology Stack** – The backend is built using frameworks like **Django (Python)**, **Flask**, or **Node.js** to handle API requests, process data, and serve recommendations efficiently.

2. **Database Management** – Stores book details, user profiles, ratings, and reviews using databases like **MySQL, PostgreSQL, or MongoDB**, ensuring efficient data retrieval.
3. **Machine Learning Model Integration** – Implements **collaborative filtering, content-based filtering, or hybrid models** using **Scikit-Learn, TensorFlow, or PyTorch** to generate recommendations.
4. **Data Processing & Filtration** – Cleans and processes datasets from sources like **books.csv, users.csv, and ratings.csv**, filtering books based on criteria like minimum reviews and average ratings.
5. **API Development** – Exposes RESTful APIs using **Django REST Framework (DRF) or FastAPI** to serve book recommendations, fetch user data, and handle interactions between the frontend and backend.
6. **User Authentication & Security** – Manages user authentication with **JWT tokens, OAuth, or session-based authentication** to protect user data and ensure secure access.
7. **Performance Optimization** – Implements **caching (Redis, Memcached)** and indexing techniques to enhance query speed and improve system efficiency.

Frontend:

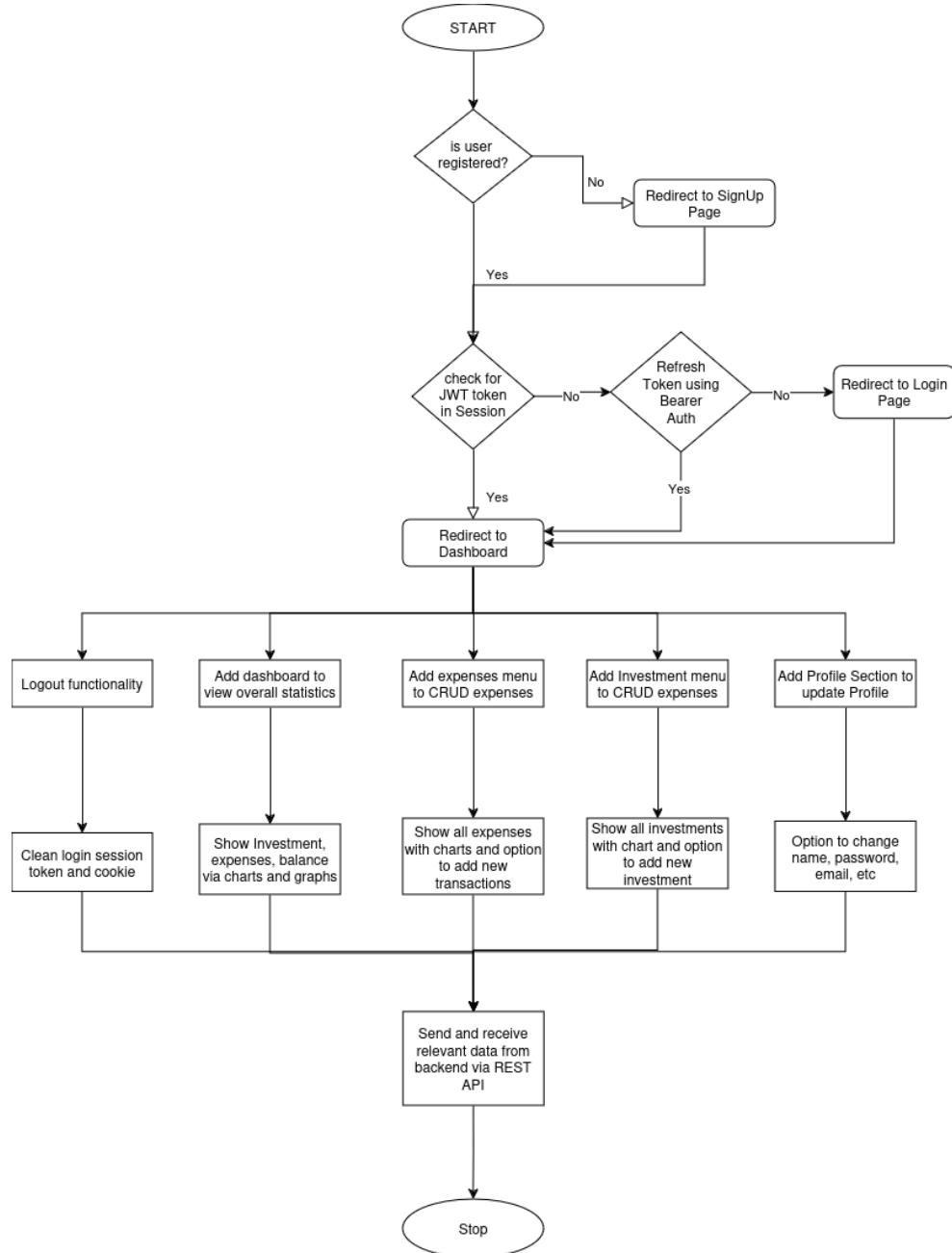
1. **User-Friendly Interface (UI/UX)** – Designed with a clean and intuitive layout using **HTML, CSS, JavaScript, and frameworks like React.js or Vue.js** for a seamless user experience.
2. **Homepage with Book Categories** – Displays sections like **Top Rated Books, Trending Books, Recommended for You**, and genre-based book collections.
3. **Search & Filter Functionality** – Allows users to search books by **title, author, genre, year of publication**, and apply filters for refined results.
4. **Book Details Page** – Shows **book cover image, author, average rating, number of reviews, summary, and user reviews** for better decision-making.
5. **Personalized Book Recommendations** – Users receive AI-based **recommended books** based on their reading history and preferences.
6. **User Ratings & Reviews Section** – Users can **rate books, write reviews, and view community feedback**, helping others make informed choices.

7. **Login & User Profile Management** – Implements **user authentication (signup, login, logout)** and profile sections where users can track **saved books, reading history, and preferences**.
 8. **Responsive & Mobile-Friendly Design** – Ensures compatibility across **desktop, tablets, and smartphones**, providing an optimized browsing experience.
 9. **API Integration for Data Fetching** – Fetches book details, ratings, and recommendations from the backend using **RESTful APIs**.
-

3.3 FLOW CHART & ALGORITHM

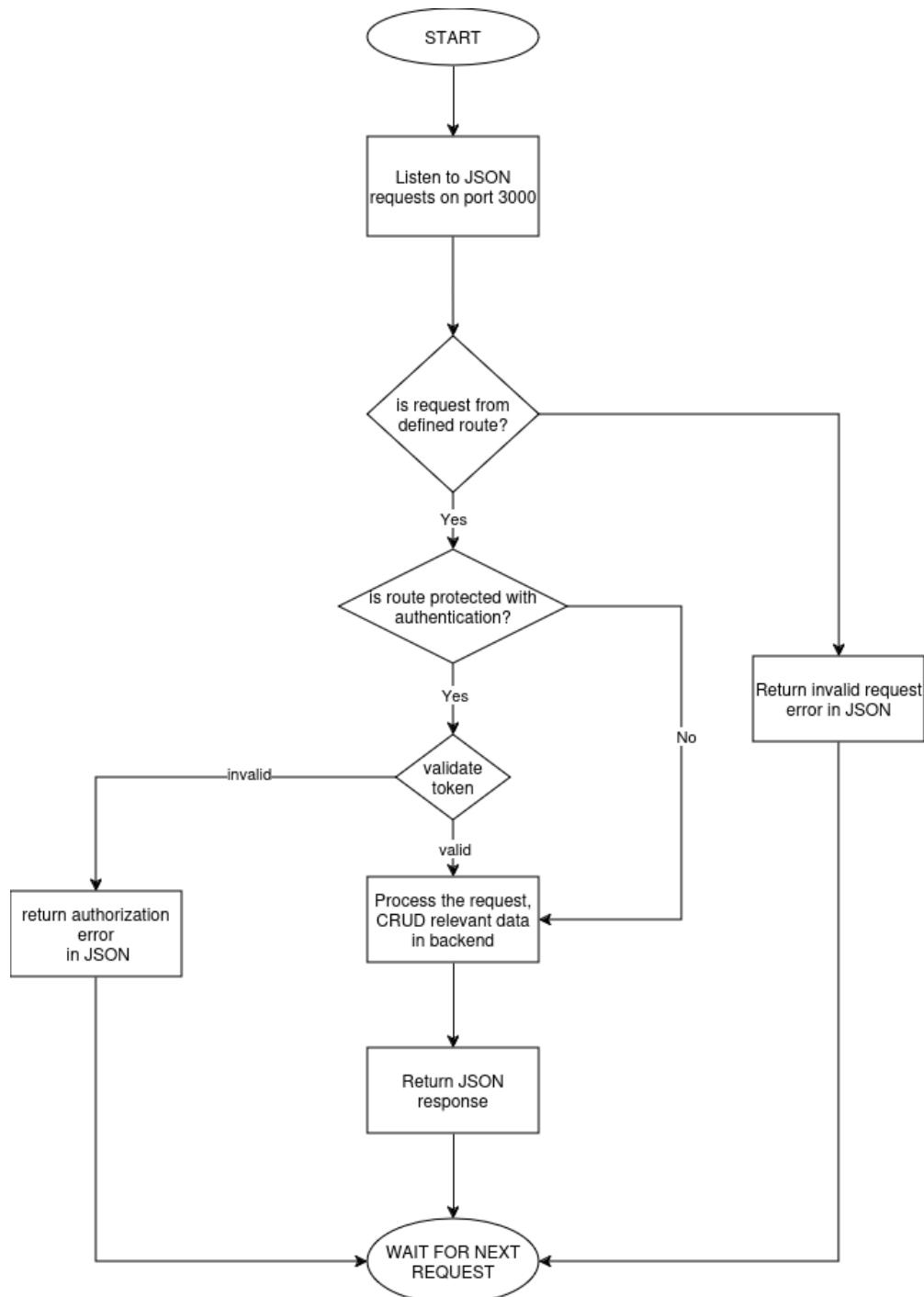
The project can be understood by various visual representations and diagrams given below:

1. Frontend Flowchart:



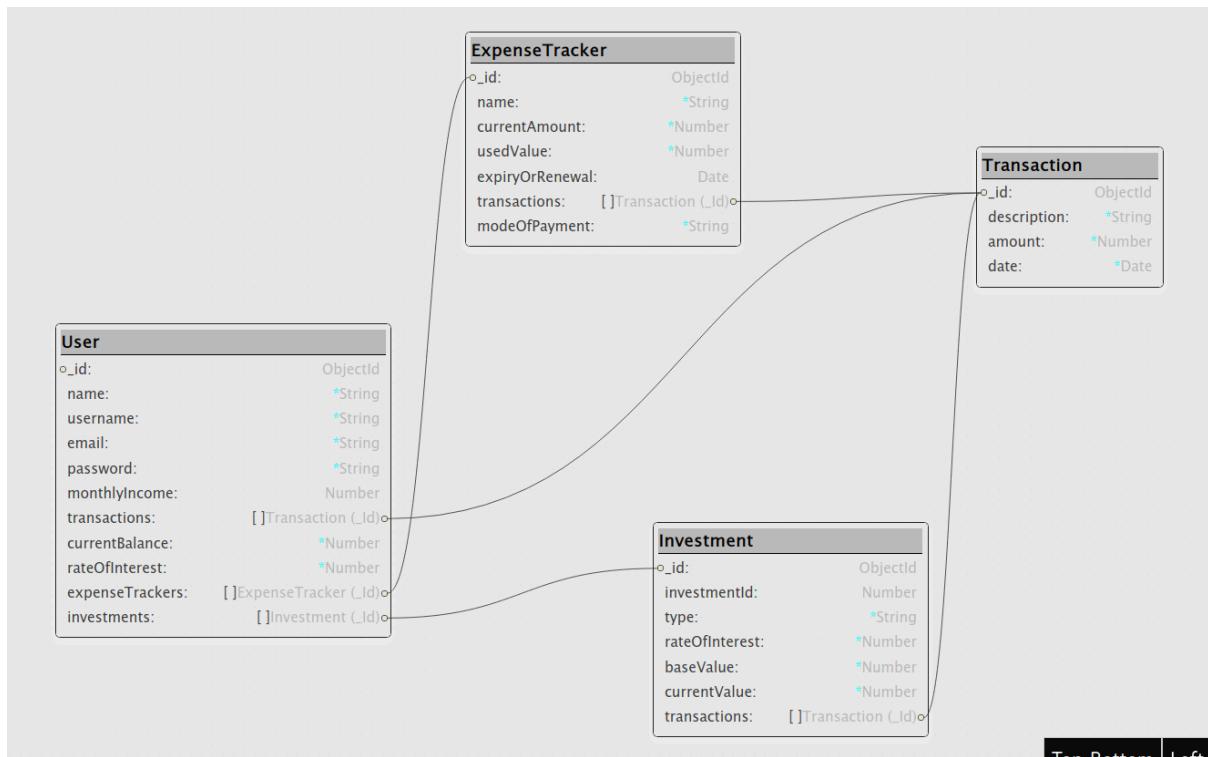
3.3.1 Frontend Flowchart

2. Backend Flowchart



3.3.2 Backend Flowchart

3. Database Design Schema Visual Representation:



3.3.3 Database Schema

3.4 OPERATION ENVIRONMENT

Hardware Requirements:

- Processor: Intel Core i5 (or AMD equivalent) and above
- RAM: Minimum 8GB (16GB recommended for better performance)
- Storage: 256GB SSD (or higher for faster data processing)
- Graphics Card: Not required but a basic GPU (NVIDIA GTX 1050 or equivalent) can help with ML models
- Operating System: Windows, macOS, or Linux

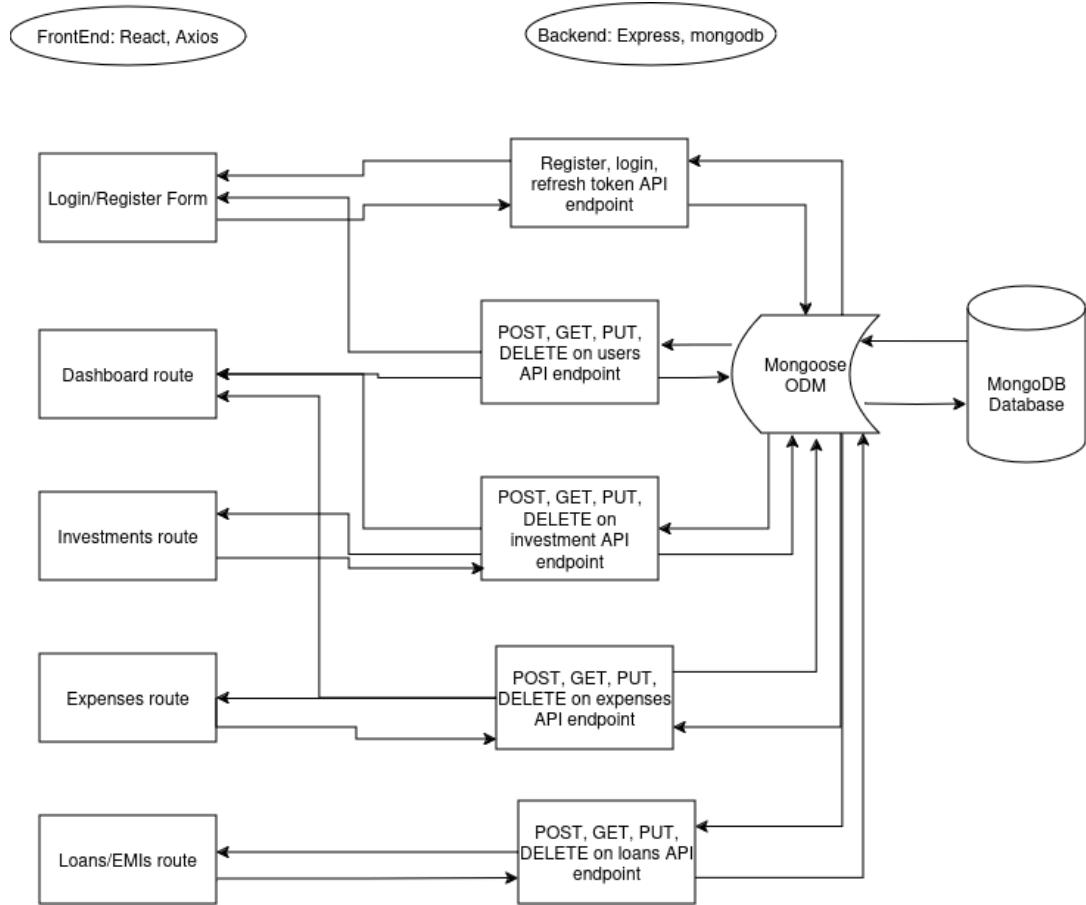
Software Requirements:

- Operating System: Windows 10/11, macOS, or Linux (Ubuntu recommended)
- Programming Language: Python 3.x (for backend & ML models)

- IDE & Code Editors: VS Code, Jupyter Notebook, or PyCharm
- Version Control: Git & GitHub/GitLab for code management

3.5 EXPERIMENT AND RESULTS

DataFlow Diagram (DFD):



3.5.1 DataFlow Diagram (DFD)

The data flow in a book recommendation system begins with data collection from various sources, including user interactions, book details, and rating datasets. This raw data, stored in structured formats such as CSV files or databases (MySQL, PostgreSQL, or MongoDB), is preprocessed to remove inconsistencies, missing values, and duplicates. Once cleaned, the data is fed into a machine learning-based recommendation model, which employs techniques like collaborative filtering, content-based filtering, or hybrid models to generate personalized book suggestions.

When a user interacts with the system, such as searching for a book or providing ratings, the frontend (built using React.js, Vue.js, or another framework) sends API requests to the backend (Django/Flask). The backend processes these requests, fetches relevant book

recommendations from the database, and returns them as a response. The recommendation logic is dynamically updated based on new user preferences and interactions, ensuring continuously improved suggestions.

Additionally, the system includes user authentication, where login credentials are validated to enable personalized experiences. The recommended books, along with metadata such as title, author, average rating, and book cover images, are displayed on the frontend interface. The data flow continues as users interact with recommendations by providing feedback, adding books to their wishlist, or leaving reviews, which are stored back into the database to refine future suggestions.

Finally, the system can be deployed on cloud platforms like AWS, Google Cloud, or Heroku, ensuring scalability and smooth user experiences. Continuous monitoring and performance tracking help optimize the recommendation algorithms, enhancing accuracy over time.

CODE SNIPPETS

SmartSpend is designed with a modular code structure, ensuring that functions are divided into individual files for easier readability and maintenance. Below are some key code snippets showcasing various functionalities of the application.

Backend root app.js code snippet:

The screenshot shows the Visual Studio Code interface with the title bar "app.js - FinancePlanner - Visual Studio Code". The left sidebar is the Explorer view, showing a tree structure of files and folders under the "FINANCEPLANNER" root. The "routes" folder is expanded, showing files like auth.js, index.js, login.js, register.js, transactions.js, user.js, and validators.js. Other visible files include middleware/auth.js, node_modules, public, and various JSON configuration files. The main editor area displays the app.js code, which is a Node.js script using Express.js to set up a web application. It includes imports for http-errors, express, path, cookie-parser, morgan, cors, and swagger-ui-express. It defines routes for index, user, login, register, investments, expenseTracker, and transactions. It also sets up CORS and Swagger documentation. The status bar at the bottom shows the file is 7 lines long, 30 columns wide, in UTF-8 encoding, and uses Babel JavaScript.

```
JS auth.js JS transactions.js JS expensetrackerjs JS index.js JS investments.js JS app.js ...  
JS opp.js > ...  
1 var createError = require('http-errors');  
2 var express = require('express');  
3 var path = require('path');  
4 var cookieParser = require('cookie-parser');  
5 var logger = require('morgan');  
6 const swaggerUi = require('swagger-ui-express');  
7 const cors = require('cors');  
8  
9 var indexRouter = require('../routes/index');  
10 var userRouter = require('../routes/user');  
11 var loginRouter = require('../routes/login');  
12 var registerRouter = require('../routes/register');  
13 var investmentRouter = require('../routes/investments');  
14 var expenseTrackerRouter = require('../routes/expensetracker');  
15 var transactionRouter = require('../routes/transactions');  
16  
17 var app = express();  
18 const swaggerDocument = require('../swagger-output.json');  
19 const corsOptions = {  
20   origin: 'http://localhost:5173',  
21   methods: ['GET', 'POST', 'PUT', 'PATCH', 'DELETE', 'OPTIONS'],  
22   credentials: true,  
23   optionsSuccessStatus: 200,  
24 }  
25  
26 // view engine setup  
27 app.set('views', path.join(__dirname, 'views'));
```

3.5.2 Backend Code Snippet

This screenshot is identical to the one above, showing the Visual Studio Code interface with the title bar "app.js - FinancePlanner - Visual Studio Code". The Explorer view on the left shows the same file structure. The main editor area displays the same Node.js code for setting up the Express.js application, including routes for index, user, login, register, investments, expenseTracker, and transactions, along with CORS and Swagger configuration. The status bar at the bottom shows the file is 7 lines long, 30 columns wide, in UTF-8 encoding, and uses Babel JavaScript.

```
JS auth.js JS transactions.js JS expensetrackerjs JS index.js JS investments.js JS app.js ...  
JS opp.js > ...  
1 var createError = require('http-errors');  
2 var express = require('express');  
3 var path = require('path');  
4 var cookieParser = require('cookie-parser');  
5 var logger = require('morgan');  
6 const swaggerUi = require('swagger-ui-express');  
7 const cors = require('cors');  
8  
9 var indexRouter = require('../routes/index');  
10 var userRouter = require('../routes/user');  
11 var loginRouter = require('../routes/login');  
12 var registerRouter = require('../routes/register');  
13 var investmentRouter = require('../routes/investments');  
14 var expenseTrackerRouter = require('../routes/expensetracker');  
15 var transactionRouter = require('../routes/transactions');  
16  
17 var app = express();  
18 const swaggerDocument = require('../swagger-output.json');  
19 const corsOptions = {  
20   origin: 'http://localhost:5173',  
21   methods: ['GET', 'POST', 'PUT', 'PATCH', 'DELETE', 'OPTIONS'],  
22   credentials: true,  
23   optionsSuccessStatus: 200,  
24 }  
25  
26 // view engine setup  
27 app.set('views', path.join(__dirname, 'views'));
```

```

14 router.post(
15   '/:username/investments',
16   [
17     body('type').custom(investmentValidator),
18     body('rateOfInterest').isFloat().withMessage('Invalid rate of interest'),
19     body('baseValue').isDecimal().withMessage('Invalid base value'),
20   ],
21   async (req, res) => {
22
23     /* #swagger.security = [{ "bearerAuth": [] } */
24
25     const { username } = req.params;
26
27     const errors = validationResult(req);
28     if (!errors.isEmpty()) {
29       return res.status(400).json({ errors: errors.array() });
30     }
31
32     const { type, rateOfInterest, baseValue } = req.body;
33
34     const transaction = createNewTransaction("Initial Investment", baseValue);
35
36     const investment = await Investment.create({
37       type,
38       rateOfInterest,
39       ...
40     });

```

3.5.3 Frontend Code Snippet

Mongoose Database schema code snippet:

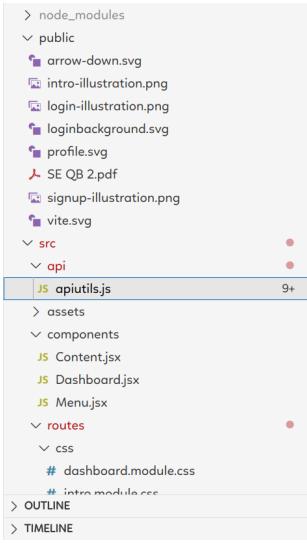
```

24 const expenseTrackerSchema = new mongoose.Schema({
25   name: {
26     type: String,
27     required: true,
28   },
29   currentAmount: {
30     type: Number,
31     required: true,
32   },
33   usedValue: {
34     type: Number,
35     required: true,
36     default: 0,
37   },
38   expiryOrRenewal: {
39     type: Date,
40     required: false,
41   },
42   transactions: [
43     {
44       type: mongoose.Schema.Types.ObjectId,
45       ref: 'Transaction',
46     },
47   ],
48 }

```

3.5.4 Database Model Code Snippet

Frontend Axios API Call functions code snippet:



```

1 import axios from "axios";
2 import { redirect } from "react-router-dom";
3
4 const apiURL = 'http://localhost:3000/api/';
5 const instance = axios.create({
6   withCredentials: true,
7   baseURL: apiURL,
8   timeout: 1000,
9   headers: {
10     'Content-Type': 'application/json',
11     'Access-Control-Allow-Origin': '*',
12   },
13 });
14
15 function setToken(token){
16   instance.defaults.headers.common['Authorization'] = `Bearer ${token}`;
17 }
18
19 export function getLocalCredentials() {
20   const userName = localStorage.getItem("username");
21   const token = sessionStorage.getItem("token");
22   return {userName, token};
23 }
24
25 async function getAuthToken() {
26   console.log("Refreshed token");
27   try {
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
217
218
219
219
220
221
222
223
224
225
226
227
227
228
229
229
230
231
232
233
234
235
235
236
237
237
238
239
239
240
241
242
243
244
244
245
246
246
247
247
248
248
249
249
250
251
251
252
252
253
253
254
254
255
255
256
256
257
257
258
258
259
259
260
260
261
261
262
262
263
263
264
264
265
265
266
266
267
267
268
268
269
269
270
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401

```

```

1 import { useLoaderData } from 'react-router-dom';
2 import Intro from './Intro';
3 import Dashboard from '../components/Dashboard';
4
5
6 function DashboardLoader() {
7   const { userData } = useLoaderData();
8   return (
9     <>
10    {userData ? <Dashboard userData={userData}> : <Intro /> }
11   </>
12 )
13 }
14
15 export default DashboardLoader

```

3.5.7 HomePage Code Snippet

Output/Snippets:

Backend API Docs using OpenAPI v3.0 Snippet:

The screenshot shows the "Personal Finance Manager API" documentation page. At the top, there is a header with the title "Personal Finance Manager API" and a version "1.0.0 OAS 3.0". Below the header, there is a "Servers" dropdown set to "http://localhost:3000/" and an "Authorize" button. The main content area is titled "default" and lists several API endpoints:

- GET /**
- GET /api/user/**
- PUT /api/user/** (highlighted in orange)
- POST /api/login/**
- POST /api/login/refreshToken**
- POST /api/login/logout**
- POST /api/register/**

3.5.8 Swagger UI API Docs Snippet

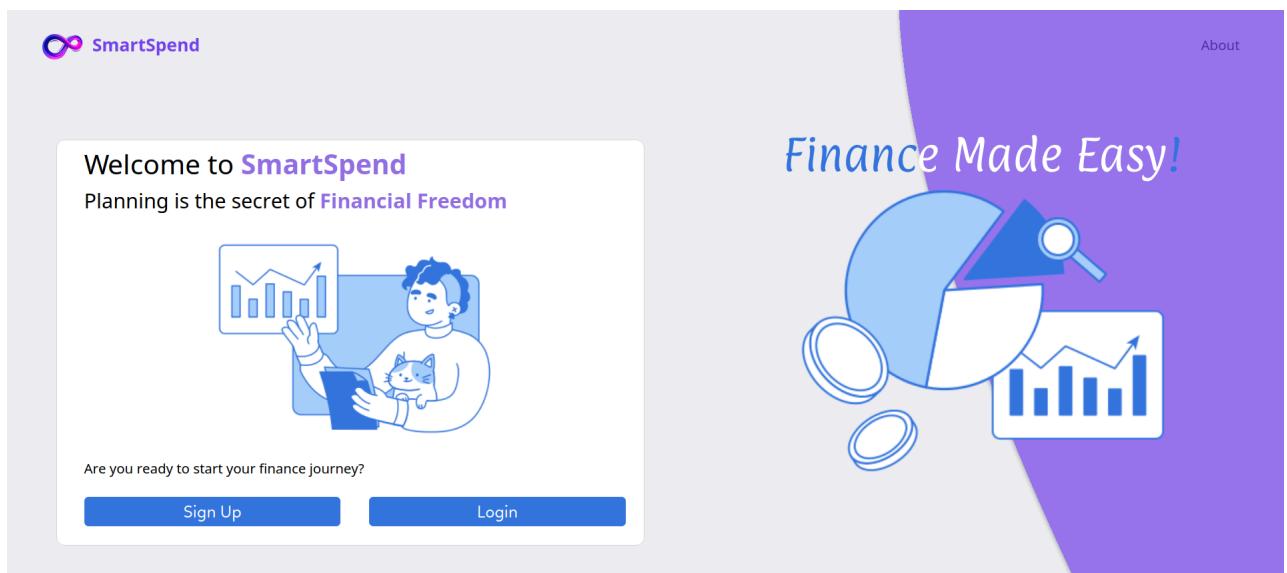
Example API call for logging in:

The screenshot shows a terminal window with the following content:

```
Request URL
http://localhost:3000/api/login/
Server response
Code Details
200 Response body
{
  "message": "Login successful",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJlc2VybmtZSI6InlhcgILCjpxYQiojE3Mjg4NzQ2MDUsInV4cCI6HTcyODg5NjIwNX0.T3eALBx9MaApBU3owL8MPc_GhVuuoU86WDNxzj8Hl"
}
Response headers
access-control-allow-credentials: true
access-control-allow-origin: http://localhost:5173
connection: keep-alive
content-length: 193
content-type: application/json; charset=utf-8
date: Mon, 14 Oct 2024 02:56:45 GMT
etag: "dclm7Qq00vhmpriAZCpM3dzLM"
keep-alive: timeout=5
vary: Origin
x-powered-by: Express
```

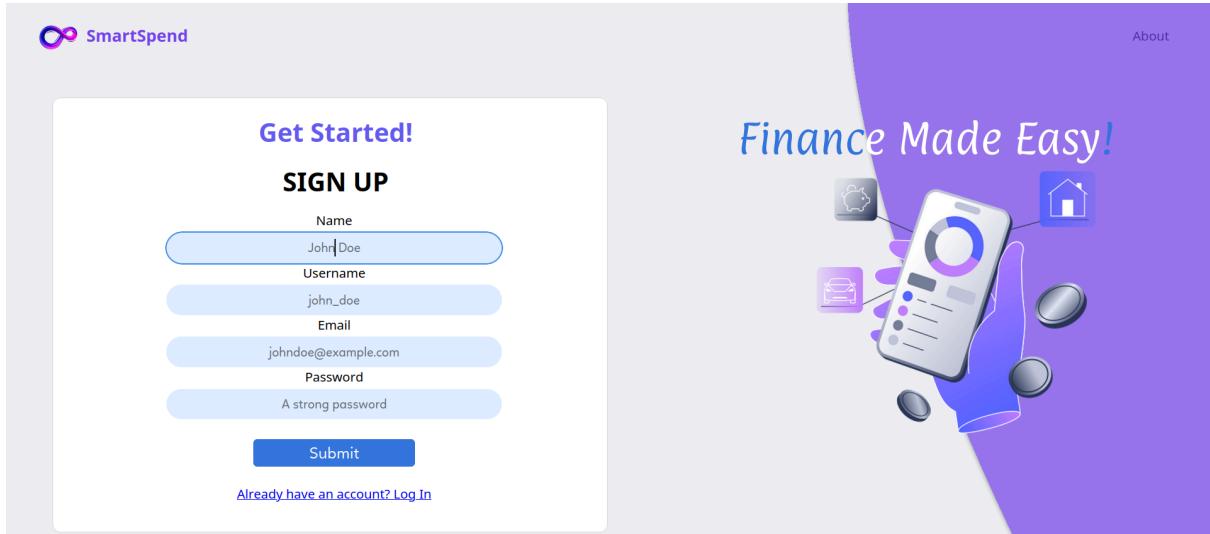
3.5.9 API Call Snippet

Frontend Intro Landing Page:



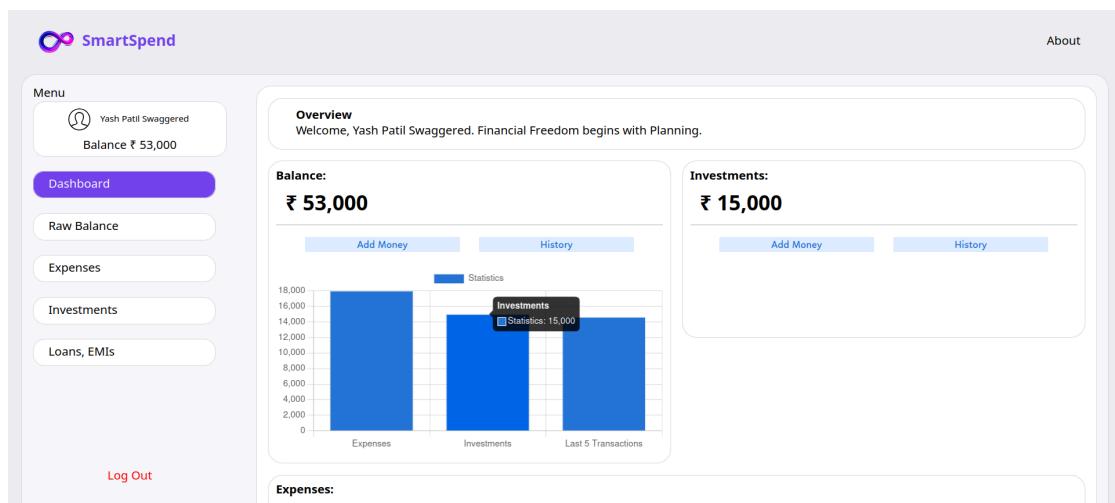
3.5.10 Landing Page Snippet

Frontend Sign Up Page:



3.5.11 SignUp Page Snippet

Dashboard Page after successful Login:



3.5.12 Dashboard Snippet

Raw Balance Page:

SmartSpend

About

Menu

- Yash Patil Swaggered
- Balance ₹ 53,000
- [Dashboard](#)
- [Raw Balance](#)
- [Expenses](#)
- [Investments](#)
- [Loans, EMIs](#)

[Log Out](#)

Balance
Check your account balance, transaction history and add new transactions.

Balance Data:

Current Balance: ₹ 53,000
 Monthly Income: ₹ 12,433
 Amount spent this month: ₹ calc
 Remaining from this month: ₹ calc
 Days remaining till next income: ₹ calc

Add/Remove money

Transaction Description

 Transaction Amount

 Transaction type:

Transactions:

Description	Amount	Date
Add Money	3000	12/10/2024, 9:24:12 pm
Test	1333	12/10/2024, 9:56:56 pm
aDD MORE MONEY	3434	12/10/2024, 9:58:12 pm
aDD MORE MONEY	3434	12/10/2024, 9:59:30 pm
aDD MORE MONEY	3434	12/10/2024, 9:59:31 pm

3.5.13 User Balance Info Snippet

SmartSpend

About

Menu

- yash4
- Balance ₹ 50,000
- [Dashboard](#)
- [Raw Balance](#)
- [Expenses](#)
- [Investments](#)
- [Settings](#)

[Log Out](#)

Investments
Check your investments, add new expenses and update existing ones.

Investment Data:

Total Investments: ₹ 1,10,000
 Investments + Account Balance: ₹ 1,60,000
 Number of Investments: 3
 Types of investments: FD, Gold, Real Estate
 Amount invested this month: ₹ 1,10,000
 Income to investment ratio: 220 %

Investment Graph:

Legend: FD (Red), Gold (Blue), Real Estate (Yellow)

3.5.14 Investment Info

The screenshot shows a user interface for managing expenses. At the top, there's a form for adding a new transaction with fields for 'New Transaction Description' (Buy Oil), 'Transaction Amount' (200), 'Transaction type' (Add spent money), and a 'Submit' button. A success message 'Added transaction of ₹ 200' is displayed in a blue box. To the left, 'EXPENSE INFO' details are shown: Expense Name: Groceries, Total Value: 10000, Used Value: 1200, Payment Method: Cash, and a progress bar at 12%. Below this is a table of 'Transactions' with columns for Description, Amount, and Date.

Description	Amount	Date
Buy Oil	200	22/10/2024, 6:56:19 pm
Buy Cereals	1000	22/10/2024, 6:56:09 pm
Create Expense	10000	22/10/2024, 6:55:46 pm

3.5.15 Expenses Snippet

The screenshot shows a user interface for account settings. On the left, a sidebar menu includes 'Dashboard', 'Raw Balance', 'Expenses', 'Investments', 'Settings' (which is highlighted in purple), and 'Log Out'. The main area has a 'Settings' section with account details: Name: Yash Patil, Username: yash4, Email: ahshf@hsdfh.co, and Monthly Income: ₹ 50,000. To the right, there's a 'Update Account Info:' form with fields for New Name (Yash Patil), Monthly Income (200), New Password, Confirm Password, and a 'Submit' button. A success message 'Updated Account Information Successfully' is shown in a blue box.

3.5.16 Account Settings Snippet

3.6 CONCLUSION

The book recommendation system leverages machine learning and data filtering techniques to provide users with personalized book suggestions based on their reading preferences, ratings, and interactions. By utilizing collaborative filtering, content-based filtering, or hybrid approaches, the system enhances the user experience by recommending books that align with their interests while also introducing them to new genres and authors.

The implementation of a scalable backend (Django/Flask) with a responsive frontend (React.js/Vue.js) ensures smooth interaction between users and the recommendation engine. Additionally, real-time data updates and user feedback loops allow continuous improvement in recommendation accuracy.

Despite challenges like cold start problems, data sparsity, and popularity bias, the system significantly enhances book discovery for readers, making it a valuable tool for online bookstores, libraries, and digital reading platforms. Future improvements, such as deep learning-based recommendations and enhanced NLP techniques, could further refine suggestions and provide an even more intuitive reading experience.

FUTURE SCOPE

1. **Advanced AI-Based Recommendations** – Implementing **deep learning techniques** such as **neural networks and NLP (Natural Language Processing)** can improve the accuracy of recommendations by better understanding user preferences and book content.
2. **Real-Time Dynamic Recommendations** – Enhancing the system to provide **real-time suggestions** by continuously analyzing user interactions, reading behavior, and current trends.
3. **Voice and Chatbot Integration** – Developing **AI-powered chatbots or voice assistants** that allow users to get book recommendations through voice commands or chat interactions.
4. **Multi-Language Support** – Expanding the system to support **multiple languages**, making it accessible to a wider audience across different regions.
5. **Social Media & Community Integration** – Allowing users to share their book recommendations, ratings, and reviews on **social media platforms** and interact with other readers in **community forums**.
6. **Sentiment Analysis for Reviews** – Using **NLP-based sentiment analysis** to analyze user reviews and provide better recommendations based on book popularity and emotional impact.
7. **Improved Search & Filter Mechanism** – Introducing **AI-driven smart search**, allowing users to find books based on specific themes, emotions, or topics rather than just keywords.
8. **Integration with Online Bookstores & Libraries** – Connecting the recommendation system with **e-commerce platforms (Amazon, Goodreads, Google Books)** and **library systems** for direct book purchases or borrowing.

REFERENCES

- [1] Gáfriková, V., Szczesny, W., & Odrzygóźdż, Z. (2015). Online Personal Finance Management Applications. *Information Systems Management*, 4, 39-52.
- [2] Jaya, I., Masih, N., Wahyuni, N., & Sugiarta, I. (2018). Development of Spreadsheet-Based Applications for Learning of Financial Management. .
<https://doi.org/10.2991/icss-18.2018.108>.
- [3] Dewi, L. (2023). Comparison Of Android-Based Personal Financial Management Applications With Variative Financial Conditions. *JAS (Jurnal Akuntansi Syariah)*.
<https://doi.org/10.46367/jas.v7i1.1098>.
- [4] React Documentation: <https://react.dev/reference/react>
- [5] ExpressJS Documentation: <https://expressjs.com/en/guide/routing.html>
- [6] React Router DOM Reference playlist:
https://www.youtube.com/watch?v=VpzeeBeVWeg&list=PL4cUxeGkcC9iNnY07bh_UPaRIQZcJfARY
- [7] React Router DOM Docs: <https://reactrouter.com/en/main/start/concepts>
- [8] Chart.js Reference: <https://www.chartjs.org/docs/latest/samples/information.html>
- [9] Axios and Authentication Reference:
 - <https://axios-http.com/docs/intro>
 - <https://deadsimplechat.com/blog/setting-headers-with-axios-in-nodejs>
- [10] JsonWebToken Reference: <https://jwt.io/introduction>

