iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IX **Month of publication:** September 2025

DOI: https://doi.org/10.22214/ijraset.2025.74191

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IX Sep 2025- Available at www.ijraset.com

Expense Edge, a Necessary Tool to Improve Financial Health

P. Sreekar Reddy[1], G. Praveen Babu[2]
[1]*Post Graduate Student, MCA, Department of Information Technology, Jawaharlal Nehru Technological University, Hyderabad, India*
[2] *Associate Professor, Department of Information Technology, Jawaharlal Nehru Technological University, Hyderabad, India*

Abstract - This project presents Expense Edge, an advanced web-based solution designed to simplify and automate personal financial management through the application of machine learning. The system addresses the common challenges of manual expense tracking, lack of financial foresight, and the absence of personalized budgeting advice. Users can register and log in to a secure platform where they can add their financial transactions (both credits and debits). The backend, developed using the Python Flask framework with MongoDB as the database, ensures secure user authentication and flexible, document-oriented data storage.

For the user, the platform provides an intuitive dashboard that offers an at-a-glance summary of their recent financial health, including income, spending, and a list of transactions. The core intelligence is delivered by a dual machine learning engine. A Naive Bayes classifier, trained on a rich, persona-driven synthetic dataset, automatically categorizes user expenses from text descriptions. A Prophet time-series model analyzes the user's entire spending history to generate a sophisticated, 30-day spending forecast.

The system's primary innovation is its hybrid recommendation engine, which synthesizes the raw statistical forecast with pragmatic financial rules, such as the user's current savings and recent income. This produces a transparent, multi-scenario financial summary and, if a savings goal is at risk, generates specific, proportional recommendations for budget cuts. Through its combination of real-time data entry, predictive modeling, and intelligent, personalized advice, the system improves financial literacy and empowers users to make informed decisions. Future developments may include mobile application integration and real-time bank account synchronization.

Keywords: Intelligent financial wellness, Predictive budgeting, Expense tracking, Machine learning, Naive Bayes classifier, Prophet time-series model, Full-stack web application, Python Flask, MongoDB, Personalized financial advice, Spending forecast, Budget recommendation, Secure user authentication, Single-Page Application (SPA), Data-driven finance

I. INTRODUCTION

In an era of increasing financial complexity, effective personal finance management has become a critical skill, yet many individuals, particularly young professionals, lack the tools for proactive planning. Existing digital solutions often function as passive ledgers, requiring tedious manual data entry and offering little more than a historical record of spending. They can show a user where their money went, but they fail to provide intelligent, forward-looking guidance on where it should go. This creates a significant gap between data collection and actionable financial wisdom, leaving users in a reactive cycle of budgeting.

This system, Expense Edge, offers a focused, data-driven, and scalable web application to address these challenges. Developed using the Flask framework for the backend and MongoDB for data persistence, it enables users to seamlessly track their income and expenses. The platform is architected as a secure, Single-Page Application (SPA) that provides a smooth and modern user experience. The platform incorporates machine learning to transform raw data into intelligent advice. A Naive Bayes classifier automates the tedious task of expense categorization from simple text descriptions. The core innovation is a Prophet time-series model that forecasts future spending patterns. This predictive insight feeds into a hybrid recommendation engine that synthesizes the forecast with the user's current savings and recent income to provide a transparent financial summary and specific, quantified advice for achieving savings goals.

While potential future developments include a mobile application and real-time bank integration, the current version delivers a robust foundation for intelligent, data-driven financial wellness. By combining user-centric design with predictive machine learning, the platform enables proactive planning, greater financial transparency, and empowers users to take confident control of their financial future.

©IJRASET: All Rights are Reserved

A. Objective

The *Expense Edge* project aims to create an intelligent web-based platform for efficient and proactive personal finance management. Built using Flask and MongoDB, the system allows users to securely track their income and expenses, while leveraging machine learning models like Naive Bayes and Prophet to gain personalized, forward-looking insights. The goal is to deliver a data-driven solution that automates manual tasks, improves financial planning, and enhances user literacy through the following key objectives:

1) Automate expense categorization using a Naive Bayes classifier.
2) Forecast future spending patterns using a Prophet time-series model.
3) Generate personalized, actionable budget recommendations based on user goals, income, and forecasted spending.
4) Provide a secure, multi-user environment with robust authentication and session management.
5) Deliver an intuitive dashboard for at-a-glance financial summaries and data visualization.
6) Ensure a scalable and robust system architecture using modern full-stack development practices.

II. LITERATURE SURVEY

Existing personal finance platforms like Mint, YNAB (You Need A Budget), and Spendee offer valuable but incomplete solutions, each exhibiting clear limitations that Expense Edge is designed to address.

Mint, a market leader, excels at data aggregation by linking various financial accounts. However, it functions primarily as a retrospective tool. Its automated categorization is often inaccurate, requiring significant manual correction by the user. Most critically, its budgeting features are reactive; they can alert a user *after* they have exceeded a budget but lack the predictive capability to warn them that their current spending trajectory *will lead* to a future deficit.

On the other hand, YNAB is built on a strong, proactive budgeting philosophy but is entirely manual. It requires immense user discipline and a steep learning curve to master its zero-based budgeting methodology. It contains no machine learning or AI-driven forecasting; it cannot analyze past spending to automatically suggest a budget or predict future expenses. All the financial intelligence and foresight must come from the user, not the system.

Spendee offers a clean user interface and good manual tracking features but, like the others, is fundamentally a digital ledger. It provides simple visualizations of past spending but lacks a predictive engine to forecast future outcomes or generate personalized, data-driven recommendations on how to achieve a financial goal.

To address these gaps, Expense Edge proposes a dedicated, intelligent platform focused on automating the most tedious tasks while providing the proactive, forward-looking insights that existing systems lack. By integrating a Naive Bayes classifier for automated categorization and a Prophet model for predictive forecasting, the platform eliminates the manual burden while empowering users with actionable intelligence. The recommendation engine's ability to synthesize a user's entire financial picture—past, present, and future—to generate specific, quantified advice is a key innovation that sets it apart from the purely retrospective tools currently available.

III.METHODOLOGY OF PROPOSED SYSTEM

A. Proposed System

The proposed system, Expense Edge, is an intelligent web-based platform developed using Python Flask and MongoDB that allows users to securely manage their personal finances. After a secure registration and login process supported by Werkzeug password hashing and Flask-Login session management, users can add their financial transactions (both credits and debits) through an intuitive interface. The system's intelligence is powered by a dual machine learning engine. A Naive Bayes classifier automatically categorizes expenses based on their text descriptions, removing the need for manual data entry. A Prophet time-series model, using a logistic growth function constrained by the user's income, generates a realistic 30-day spending forecast.

This forecast is the primary input for the system's core feature: the Intelligent Recommendation Engine. This engine synthesizes the user's stated savings goal, their current savings balance, their recent income, and the constrained forecast to produce a transparent, multi-scenario financial summary. If the user is not on track to meet their goal, the engine calculates the precise shortfall and generates specific, proportional recommendations for budget cuts across discretionary spending categories. This platform streamlines financial tracking and equips users with powerful, data-driven tools for proactive planning and smarter financial decision-making.

B. Dataset Description for Machine Learning Models

This project utilizes a high-fidelity, programmatically generated synthetic dataset to train its machine learning models. This approach was chosen to ensure user privacy and to create a rich, logically consistent dataset tailored to the project's requirements. The dataset is generated in-memory for each new user during the onboarding process.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Key Dataset Features and Generation Process:

- Structure and Content: The dataset consists of over 2,200 transaction records simulating a 36-month (3-year) financial history. Each record includes a date, description, amount (in INR), category, and type (credit or debit).
- Persona-Driven Logic: The data is generated based on a realistic persona of a young professional in India. This includes a starting salary of ₹30,000 with a ₹10,000 increment every 12 months.
- Realistic Financial Behavior: The generation script ensures the persona is solvent by setting a monthly expense budget between 60% and 90% of that month's income, guaranteeing a positive savings rate. Spending is probabilistically weighted towards categories like "Dining Out" and "Partying/Friends" to reflect the persona's lifestyle.
- Chronological Integrity: After generation, the entire dataset is explicitly sorted by date, a critical pre-processing step for the time-series forecasting model.
- Role in Modeling: The debit transactions from this dataset are used to train the Naive Bayes classifier. The entire 3-year history is used to provide an immediate and robust foundation for the Prophet forecasting model for new users.

C. System Architecture

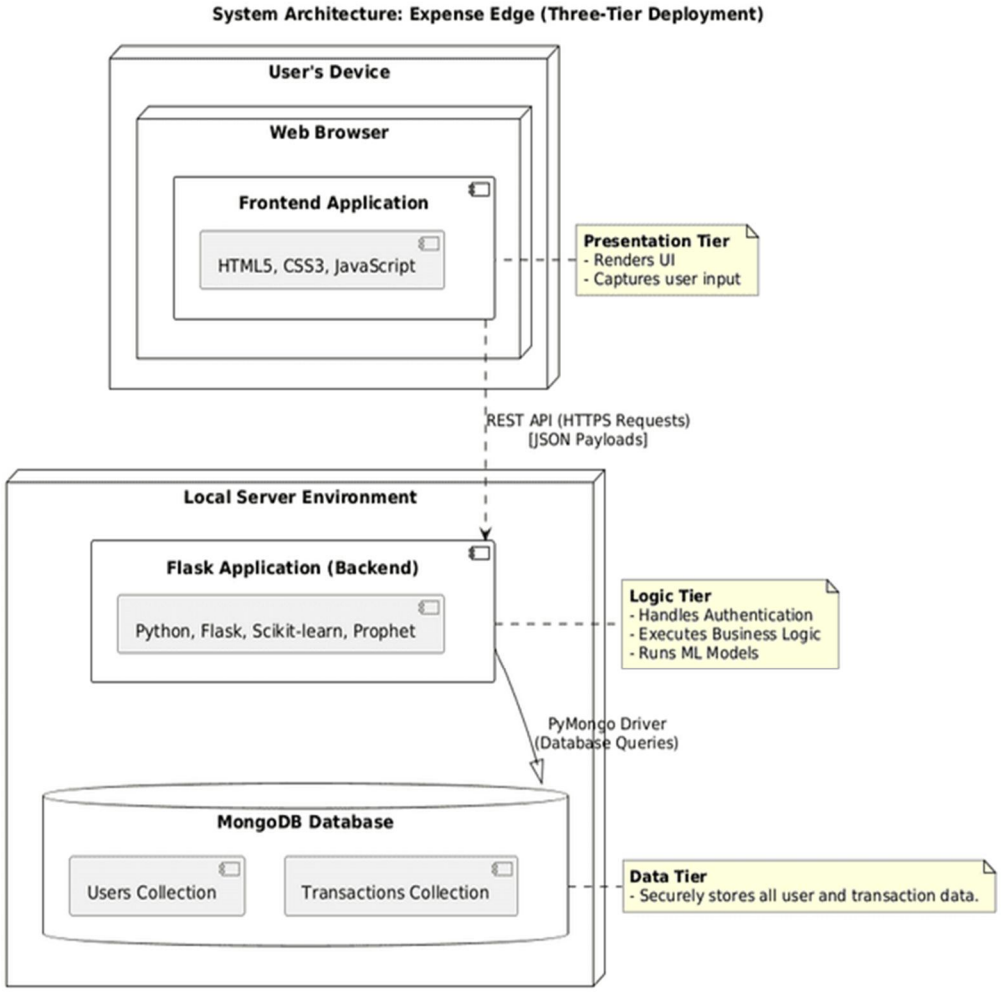


Figure-1: System Architecture of Expense Edge

The system is architecturally designed to facilitate a seamless user experience for financial management and to provide intelligent, predictive insights through an integrated machine learning engine. The workflow is logically divided into two interconnected streams: the User Interaction and Data Flow and the Machine Learning Model Lifecycles.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IX Sep 2025- Available at www.ijraset.com

D. User Interaction and Data Flow

This workflow describes the complete sequence of events from the perspective of a user interacting with the application.

1) User Registration and Onboarding: The process begins when a new user signs up via the registration form. Upon successful creation of an account in the users collection, the system automatically logs them in and redirects them to a welcome page. Here, the user chooses their onboarding path: either uploading their own JSON data or triggering the in-memory generation of a rich, synthetic dataset. This initial data seeding is a critical step that populates the transactions collection and enables the immediate use of the application's data-driven features.
2) Data Storage: All user and transaction data is securely stored in a MongoDB database. The users collection holds credentials with hashed passwords, while the transactions collection stores all financial records. Each transaction document is linked to a user via a user_id, ensuring strict data isolation and privacy.
3) Dashboard Interaction: Once logged in, the user is presented with the main dashboard. This view is populated by an asynchronous call to the /api/dashboard-summary endpoint. The backend fetches the user's transactions from the last 30 days, calculates key metrics like total income and spending for that period, and returns a consolidated JSON object to the frontend for rendering.
4) Adding a New Transaction: The user can add new credit or debit transactions via a dedicated form. Upon submission, the data is sent to the /api/transactions endpoint. If the transaction is a debit, the backend first passes its text description to the pre-trained Transaction Classifier model to get an automated category before saving the complete, categorized document to the database.
5) Requesting a Budget Recommendation: This is the core intelligent interaction. The user enters a savings goal and submits the request. The backend then initiates the full machine learning pipeline: it fetches the user's entire history, generates a 30-day spending forecast, and synthesizes this prediction with the user's current financial snapshot to generate a detailed, transparent recommendation.

E. Machine Learning Model Lifecycles

1) *Lifecycle A: Transaction Classifier (Naive Bayes)*

- Dataset Sourcing: The training data for this model is sourced from the _generate_starter_transactions function. It uses the debit transactions from this high-fidelity synthetic dataset, which contains over 2,200 records with clear descriptions and categories.
- Preprocessing: The raw text from the description field undergoes a Natural Language Processing (NLP) pipeline. It is passed through a TF-IDF Vectorizer from the scikit-learn library, which cleans the text and converts it into a numerical matrix that represents the statistical importance of each word.
- Model Training: A Multinomial Naive Bayes classifier is trained on the TF-IDF matrix (the features) and the corresponding transaction categories (the labels). This training process is encapsulated in a scikit-learn Pipeline and is executed only once when the server starts and a pre-trained model is not found.
- Model Persistence: After training, the optimized model object is serialized and saved to a file named transaction_classifier.joblib using the joblib library. This allows the application to load the model into memory instantly on subsequent startups, avoiding the need for costly retraining.
- Real-time Prediction: The loaded model is used in real-time. When a user adds a new expense, its description is passed to the model's .predict() method, which returns the most probable category in milliseconds.

2) *Lifecycle B: Spending Forecaster (Prophet)*

- Data Loading: This model's lifecycle is initiated on-demand whenever a user requests a forecast or recommendation. The system fetches the entire transaction history for that specific user from the MongoDB database.
- Data Cleaning & Preprocessing: The loaded data undergoes a rigorous preparation sequence using the pandas library.
    a) The data is filtered to include only debit transactions.
    b) Any records missing a date are dropped.
    c) All date values are converted to a consistent, timezone-naive format.
    d) The transactions are then aggregated by day, summing all expenses to create a clean time series with columns renamed to ds (date) and y (value), as required by the Prophet model.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

- Model Training (Fitting): A new Prophet model is instantiated and configured to use a logistic growth model. A crucial "capacity cap" is calculated from the user's average monthly income and added as a cap column to the data. The model is then fitted to this user-specific, capped data. This on-demand training ensures the forecast is always personalized and up-to-date.
- Prediction: The newly trained model is used to predict the spending for the next 30 days. The prediction is a rich DataFrame containing the most likely forecast (yhat) as well as the 80% confidence interval (the yhat_lower and yhat_upper bounds).
- Post-processing & Integration: The raw forecast values are extracted and passed to the Intelligent Recommendation Engine. A final, rule-based pragmatic constraint is applied (capping the forecast at the user's income) to produce a "Realistic Spend" value. This final, refined number is then used to calculate the user's projected savings and generate actionable advice.

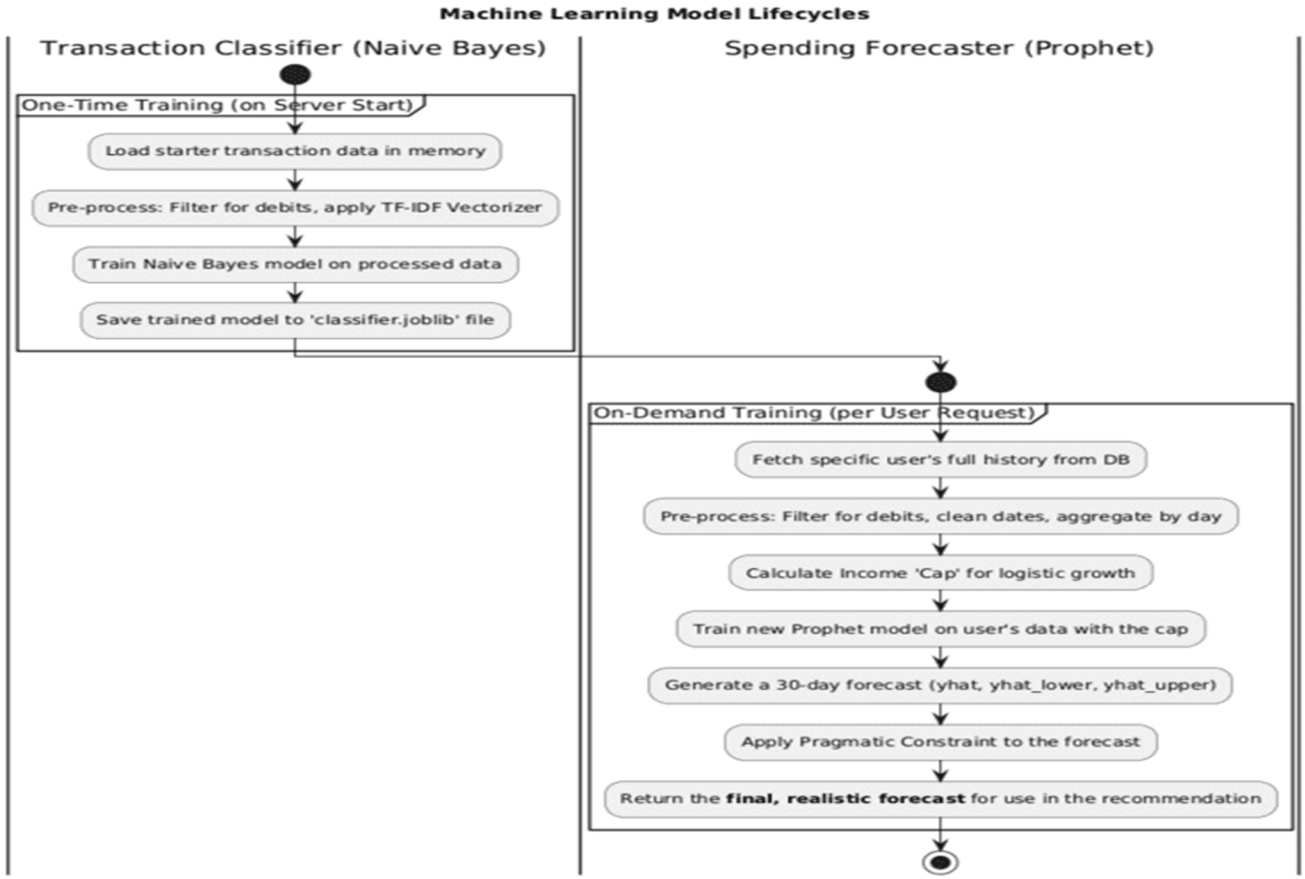


Figure-2: Naïve Bayes to Classify and Prophet to Forecast Spending

3) Methodology

The implementation follows a structured pipeline for integrating the machine learning intelligence into the application.

a) Data Preparation and Onboarding: When a new user registers, the FinancialService module programmatically generates the synthetic dataset in memory. This data is immediately formatted into BSON documents and bulk-inserted into the MongoDB transactions collection, linked to the new user's unique ID.

b) Model Development and Training:

- Naive Bayes Classifier: This model is trained once when the server starts (if a saved model.joblib file is not found). It uses the debit portion of the generated dataset. The text descriptions are vectorized using TF-IDF, and the scikit-learn pipeline is fitted to the data. The trained model is then serialized and saved to disk for instant use.
- Prophet Forecaster: This model is trained on-demand for each user. When a forecast is requested, the user's entire spending history is fetched, pre-processed into a time-series format, and used to fit a new Prophet model. This ensures the forecast is always personalized and up-to-date. The model is configured to use a logistic growth function, with the user's average income passed as a "capacity cap" to ensure financial realism.

c) Application Integration: The trained models are integrated as classes within the backend service layer. The Naive Bayes model is called in real-time during transaction creation. The Prophet model is called by the recommendation engine. The backend endpoints then expose this intelligence to the frontend, which is responsible for all visualization and user interaction.

F. Database design (MySQL Schema)

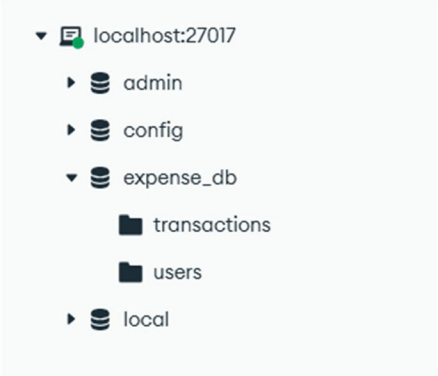


Figure-3: MangoDB Database Tables List

The project utilizes a MongoDB database named expense_db with two core collections.

1) Users collection: Stores a document for each registered user.
- _id: ObjectId (Primary Key)
- email: String (Unique identifier for login)
- password_hash: String (Securely hashed password)

2) Transactions collection: Stores a document for every financial transaction.
- _id: ObjectId (Primary Key)
- user_id: String (Foreign key linking to the _id of a document in the users collection)
- date: Datetime (Timestamp of the transaction, stored in UTC)
- description: String (e.g., "Zomato", "Monthly Salary")
- amount: Float (Value in INR)
- category: String (e.g., "Dining Out", "Income")
- type: String ("credit" or "debit")

IV. RESULTS

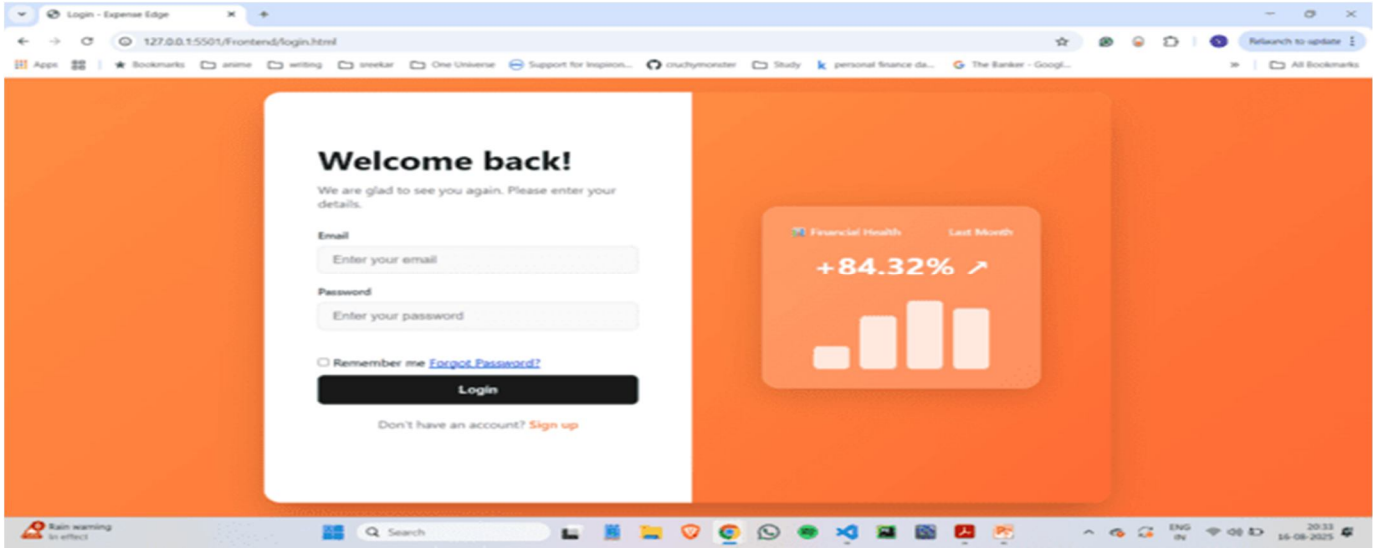


Figure-4: Login page for Users

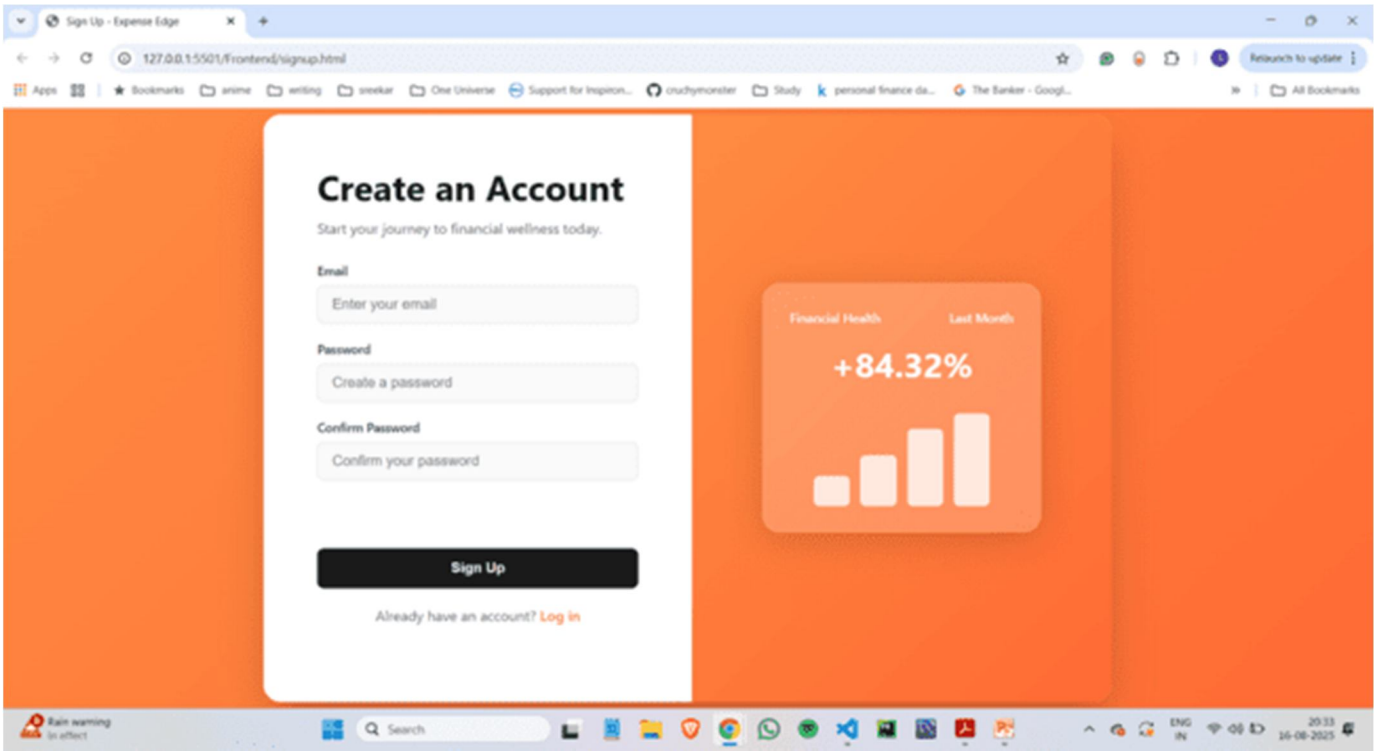


Figure-5: Sign-Up Page for New Users.

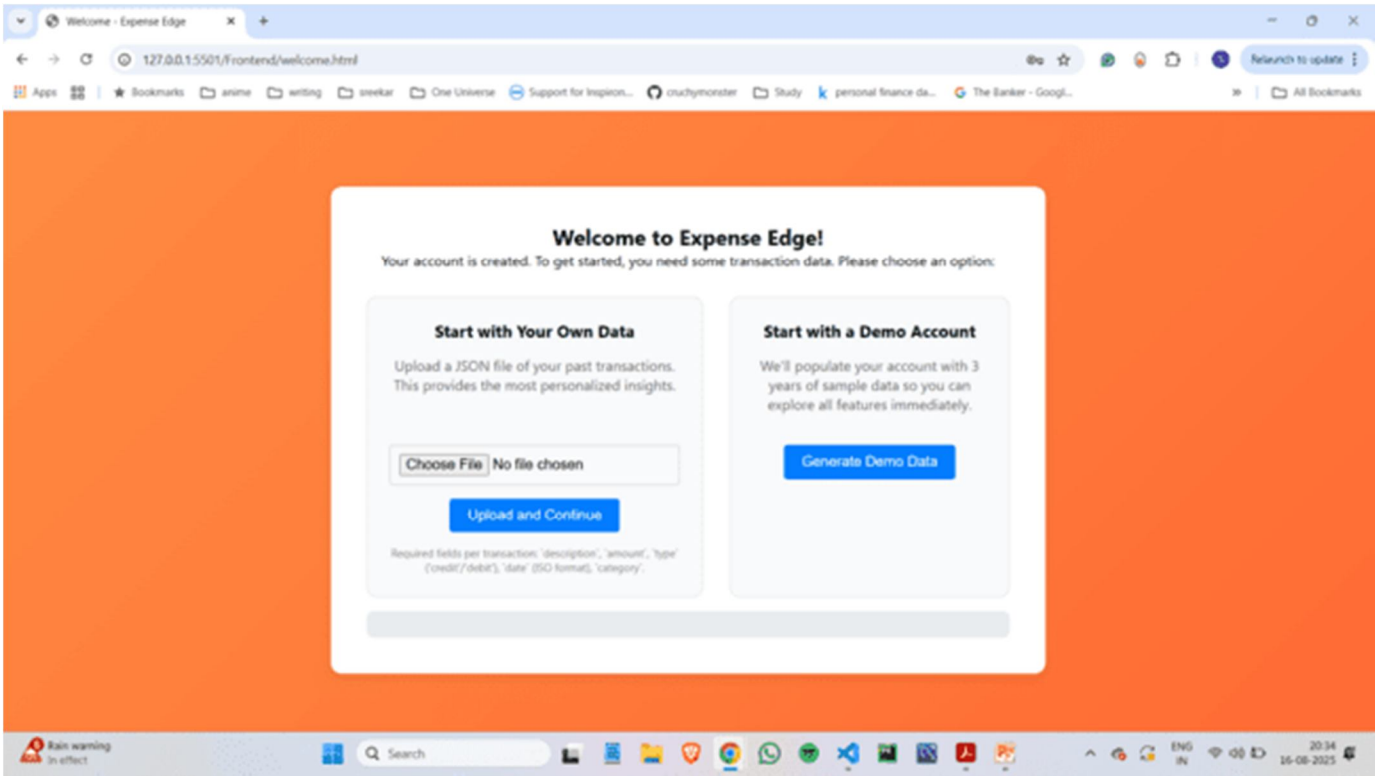


Figure-6: New Users On boarding Options Page

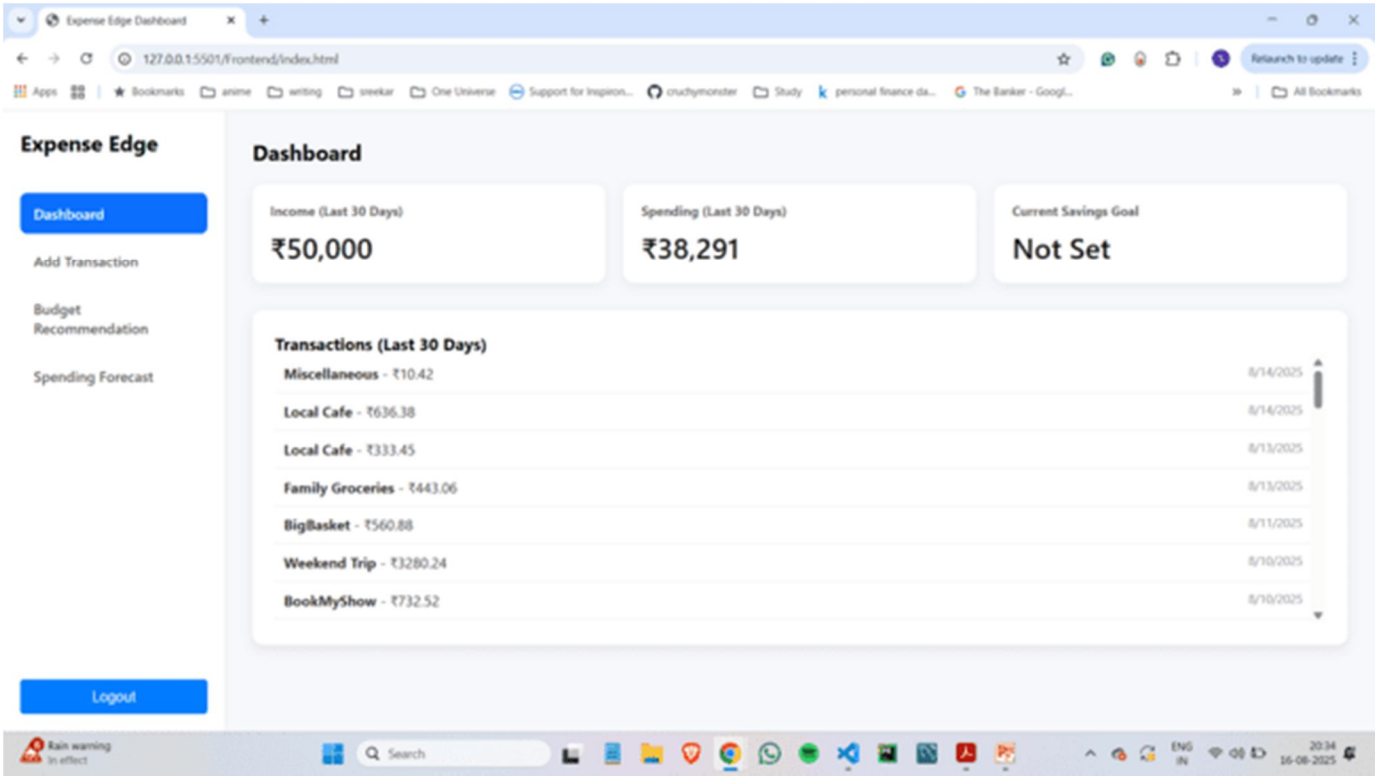


Figure-7: Main Dashboard Interface

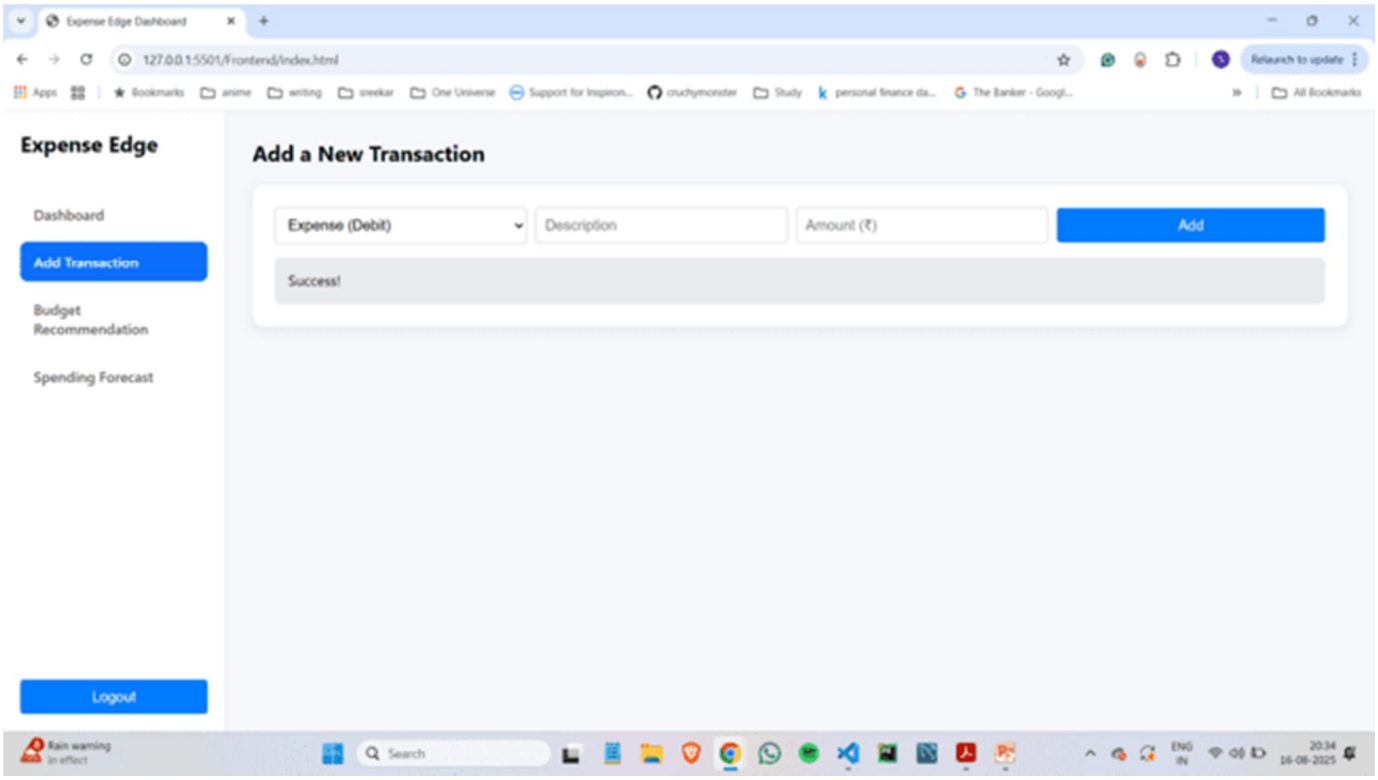


Figure-8: Add a New Transaction Module

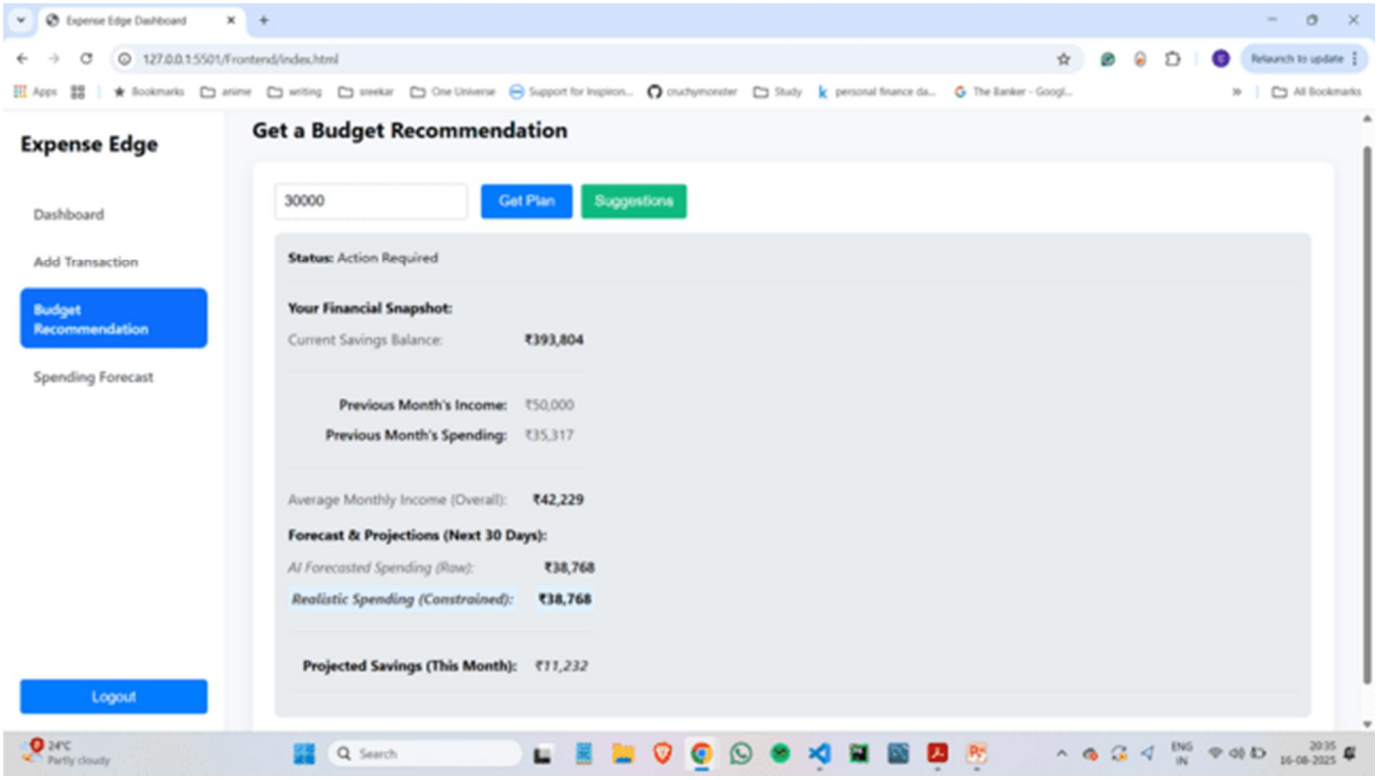


Figure-9: Budget Recommendation – Financial Summary

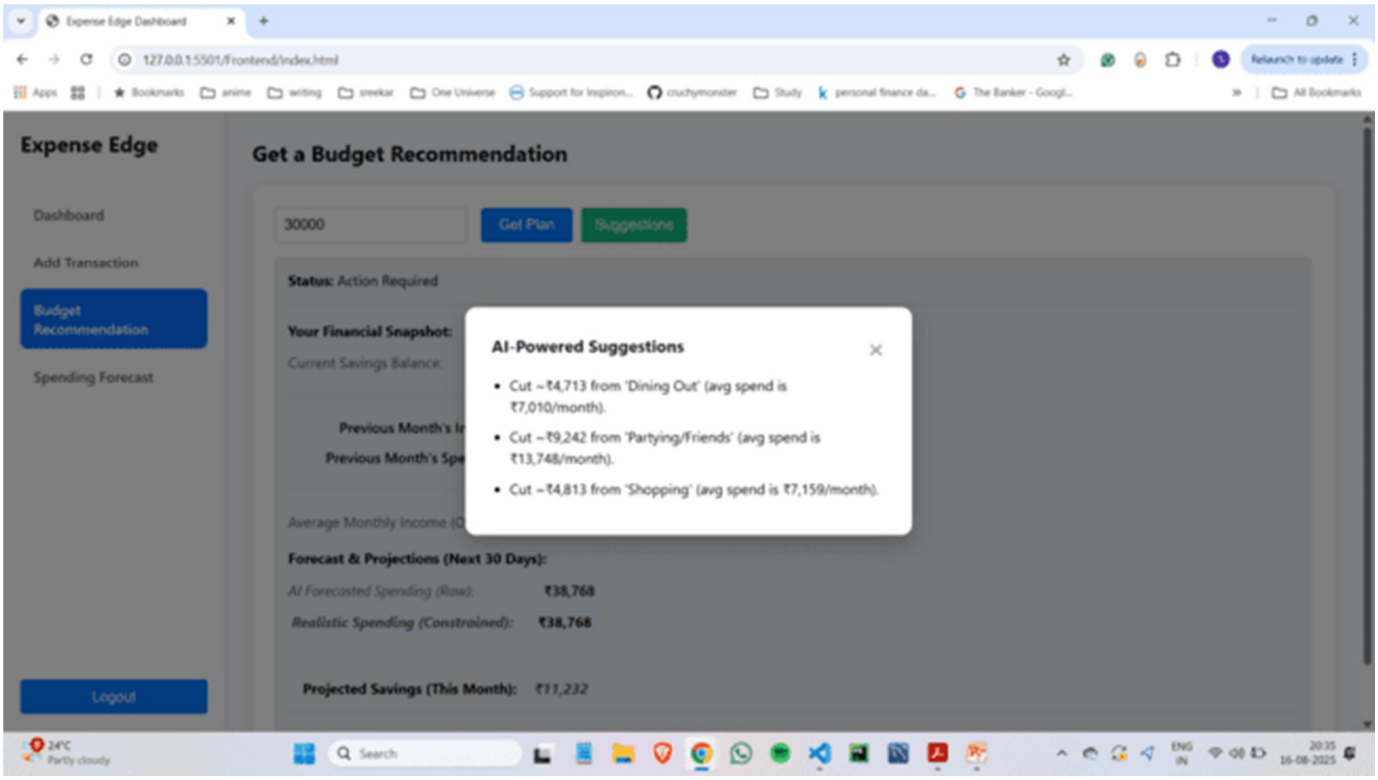


Figure-10: Actionable Suggestions Modal

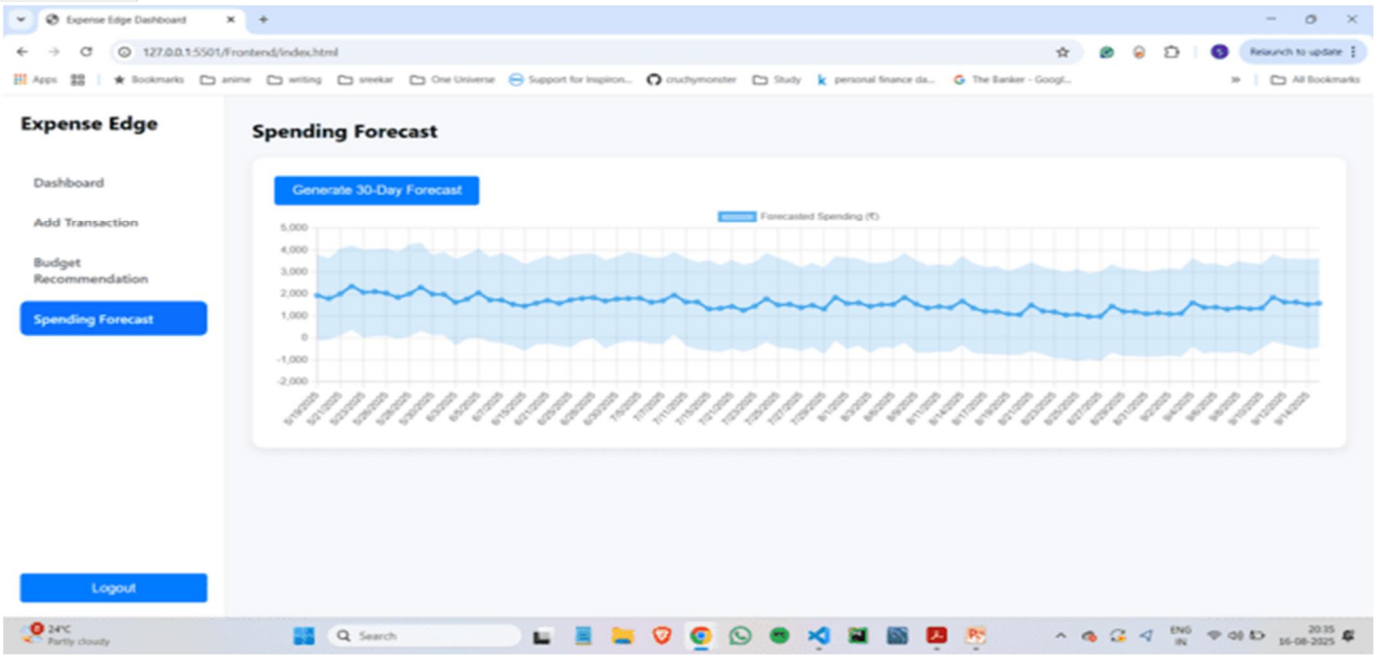


Figure-11: Spending Forecast Visualization

A. Description About the Results

The system is accessed via a professional and secure Login Page (Figure-4). New users can create an account through the Sign-Up Page (Figure-5). Upon their first login, users are directed to the unique Onboarding Page (Figure-6), where they can choose to either upload their own transaction history or start with a rich, pre-populated demo account.

Once onboarded, the user is presented with the main Dashboard (Figure-7), which functions as a Single-Page Application. This central hub displays at-a-glance financial metrics for the last 30 days, such as total income and spending, and a list of recent transactions. Users can navigate via the sidebar to the Add a New Transaction page (Figure-8), where new expenses are automatically categorized by the Naive Bayes model. The core intelligence is demonstrated in the Budget Recommendation feature (Figure-9). After a user sets a savings goal, the system displays a transparent financial summary, including their current savings, recent performance, and a multi-scenario forecast. If action is needed, a button appears, which triggers the Actionable Suggestions Modal (Figure-10), providing specific, quantified budget cuts. Finally, the Spending Forecast Visualization (Figure-11) shows a clear Chart.js graph of the Prophet model's prediction, complete with a "cone of uncertainty" representing the model's confidence.

V. LIMITATIONS AND FUTURE SCOPE

While the Expense Edge project successfully delivers a robust and intelligent proof-of-concept, its current implementation as a prototype has several inherent limitations. These constraints, however, serve to define a clear and exciting roadmap for future enhancements that could elevate the application to a production-grade financial wellness platform.

The primary limitations of the current system are concentrated in the areas of deployment, data sourcing, and model evolution. The application is currently designed to run in a local development environment, which is not suitable for a public, multi-user deployment. The system's intelligence is bootstrapped using a synthetic dataset, and the machine learning models would benefit greatly from a user feedback loop for continuous improvement. The reliance on manual transaction entry is also a significant point of friction for long-term user engagement.

The future scope for Expense Edge is extensive. Key enhancements would include:

1) Cloud Deployment: Migrating the entire application stack to a cloud platform like AWS or Heroku and containerizing it with Docker to create a scalable, production-ready service.

2) Real-Time Bank Integration: Integrating with a financial data aggregation API (like Plaid) to automate transaction import, which would be the single most significant feature for user adoption.

3) Model Retraining Pipeline: Implementing a feature for users to correct miscategorized transactions. These corrections would be used to periodically retrain the Naive Bayes classifier, creating a self-improving AI that becomes more personalized over time.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

4) Advanced Anomaly Detection: Integrating an unsupervised learning model, such as an Isolation Forest, to flag statistically unusual transactions and alert the user to potential fraud or impulsive spending.

VI. CONCLUSION

The project titled "Expense Edge: An Intelligent Financial Wellness Platform with Predictive Budgeting" successfully delivers on its objective to create an intelligent, user-centric application that simplifies personal finance management while empowering users with predictive insights and data-driven visual analytics.

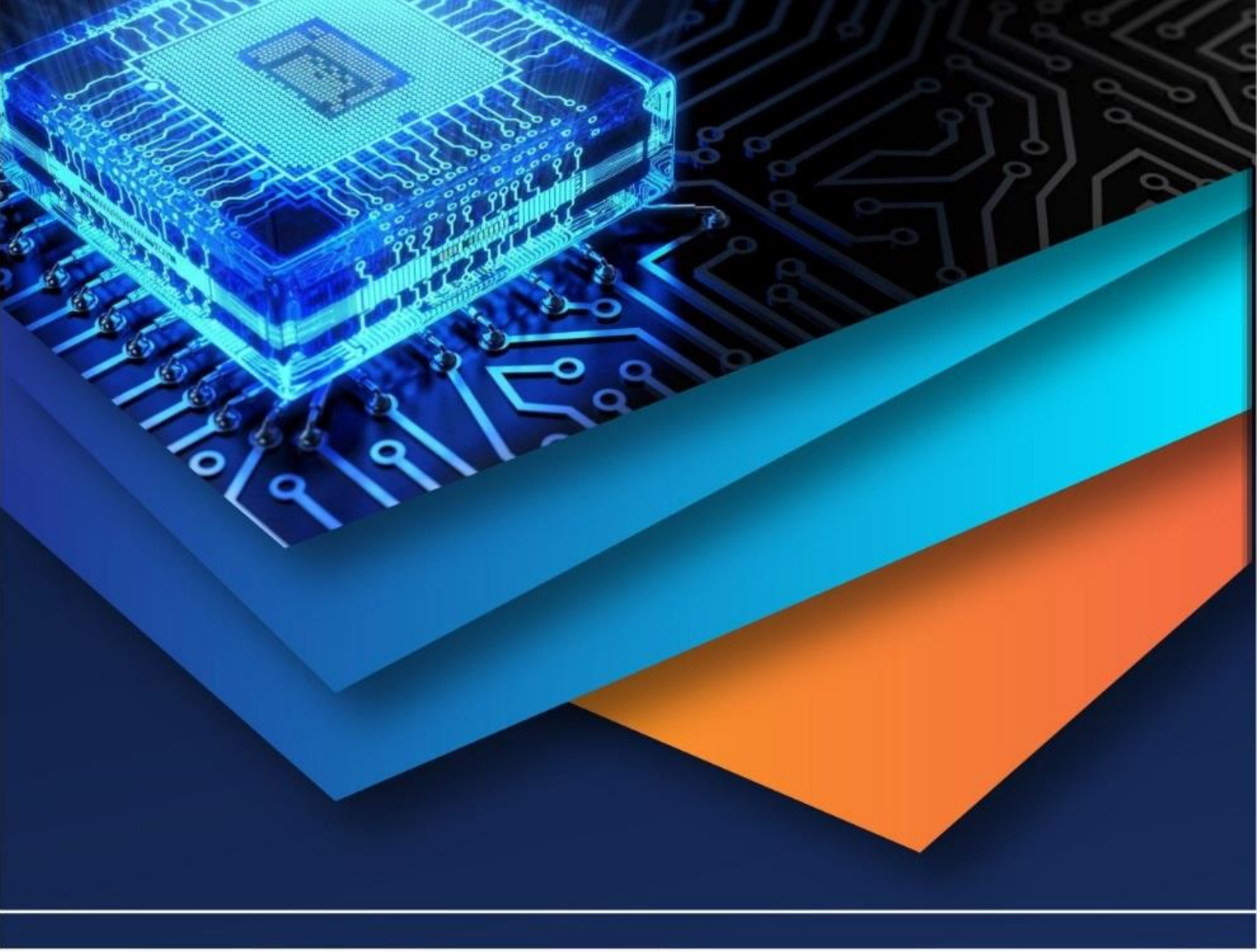
Through the seamless integration of Python Flask, MongoDB, and a dual machine learning engine (Naive Bayes and Prophet), the system automates tedious tasks like expense categorization and provides sophisticated, forward-looking spending forecasts. The core innovation—the hybrid recommendation engine—synthesizes statistical predictions with pragmatic financial rules to produce advice that is transparent, personalized, and actionable.

The platform is simple to use, architected for scalability, and designed with modularity and security in mind. It directly addresses the limitations of traditional, retrospective expense trackers by offering a focused, proactive, and intelligent solution for modern financial planning. Expense Edge stands as a robust proof-of-concept and a strong foundation for a future-ready, data-driven financial wellness tool.

REFERENCES

[1] Chart.js Team. (2024). Chart.js Documentation. Available at: https://www.chartjs.org/docs/latest/
[2] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585, 357-362. Official documentation available at: https://numpy.org/doc/
[3] The Joblib Development Team. (2024). Joblib Documentation (for Model Persistence). Available at: https://joblib.readthedocs.io/
[4] Ling, M. (2024). Flask-Login Documentation. Available at: https://flask-login.readthedocs.io/
[5] MongoDB, Inc. (2024). MongoDB Compass Documentation. Available at: https://www.mongodb.com/docs/compass/current/
[6] The Pallets Projects. (2024). Werkzeug Documentation (for Password Security). Available at: https://werkzeug.palletsprojects.com/
[7] The Pandas Development Team. (2024). pandas-dev/pandas: Pandas. Zenodo. Official documentation available at: https://pandas.pydata.org/docs/
[8] Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. Official documentation available at: https://scikit-learn.org/stable/documentation.html
[9] PlantUML Team. (2024). PlantUML Language Reference Guide. Available at: https://plantuml.com/
[10] Python Software Foundation. (2024). Python Language Reference, version 3.8. Available at: https://docs.python.org/3/
[11] Richardson, C. (2024). Flask-CORS Documentation. Available at: https://flask-cors.readthedocs.io/
[12] Ronacher, A., & The Pallets Projects. (2024). Flask Documentation. Available at: https://flask.palletsprojects.com/
[13] Taylor, S. J., & Letham, B. (2017). Prophet: Forecasting at Scale. Meta Research. Official documentation available at: https://facebook.github.io/prophet/docs/quick_start.html
[14] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer. (This is a foundational academic text that provides the theoretical basis for probabilistic classifiers like Naive Bayes).
[15] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and Practice. OTexts. Available at: https://otexts.com/fpp2/ (This is a widely respected open-source textbook on time-series analysis, providing the context for using forecasting models like Prophet).

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)