Infosys Internship Project

Buddy With Brain

(Ai-Powered Personal Expense Analyzer)

Project Title: Buddy With Brain - Personal Budgeting & Expense Forecaster

Author: Srujan Manukonda

Date: November 20, 2025

Under the Guidance of Infosys Springboard Mentor

Dr.K.Santhiya Krishnasamy

DECLARATION

I, Srujan Sai Manukonda, hereby declare that the project report entitled "Buddy With Brain: AI-Powered Personal Expense Analyzer" submitted in partial fulfillment of the requirements for the Infosys Springboard 6.0 Internship, is a record of an original work done by me.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Srujan Sai Manukonda

Date: November 20, 2025

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible.

I would like to express my deep sense of gratitude to Infosys Springboard for providing me with this opportunity to undergo the Internship Program (Batch 6.0). The rigorous curriculum and the structured milestones (Milestones 1 through 4) provided the technical foundation necessary to complete this complex application.

I am also grateful to my mentors and the technical support team at Infosys Springboard for their guidance on modern web frameworks, Python data science libraries, and Docker containerization.

Finally, I thank my family and friends for their constant encouragement and support throughout the development of this project.

Srujan Sai Manukonda

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Project Statement

In the modern digital economy, individuals generate a massive volume of financial transactions daily. From Unified Payments Interface (UPI) transfers to credit card swipes and cash expenditures, money flows through various channels. While spending is effortless, tracking these expenses is a significant challenge.

Many individuals struggle with managing their personal finances effectively, often finding it difficult to stick to a budget or plan for future financial goals. The sheer volume of transactions can be overwhelming, making it hard to identify spending patterns manually. This lack of financial clarity leads to stress, missed savings opportunities, and difficulty in achieving long-term financial stability.

"Buddy With Brain" is a web-based Personal Budgeting & Expense Forecaster designed to address these challenges. Unlike traditional spreadsheet trackers, this project leverages **Artificial Intelligence (AI)** and **Natural Language Processing (NLP)**. It acts as an intelligent financial companion that not only records what happened in the past but uses **Time-Series Forecasting** to predict what will happen in the future.

Traditional budgeting apps often fail because they require tedious manual data entry, forcing users to select specific dates, categories, and amounts via complex dropdown menus. In contrast, this system understands natural human language. A user can simply input a phrase like *"Spent 500 on groceries at D-Mart,"* and the system intelligently parses the sentence, extracts the entity (D-Mart), the category (Groceries), and the cost (500), automatically logging it into the database.

1.2. Objectives

The primary objectives of this internship project were divided into four strategic milestones, all of which have been successfully implemented:

1. **To Simplify Data Entry:** To create a system that accepts data from multiple sources—CSV files, Excel sheets, PDF bank statements, and manual entry—so users don't have to type every transaction.
2. **To Automate Categorization:** To implement an NLP engine using NLTK that reads transaction descriptions (e.g., "Uber Trip") and automatically assigns the correct category ("Transport") without user intervention.
3. **To Forecast Future Expenses:** To integrate the Facebook Prophet model to analyze historical data and generate a 90-day spending forecast.
4. **To Visualize Financial Health:** To provide an interactive dashboard with KPI metrics, Income vs. Expense trend lines, and Savings Rate calculations.
5. **To Ensure Security and Scalability:** To implement secure Role-Based Access Control (RBAC) for Admins and Users, and to containerize the application using Docker for deployment.

1.3 Scope and Problem Definition

The Problem: Most existing expense trackers are "Reactive." They tell the user: *"You spent ₹50,000 last month."* While this is useful, it is often too late to make a difference. Furthermore, these apps require heavy manual input. If a user buys coffee 30 times a month, they must manually tag "Coffee" as "Food" 30 times. This friction causes users to abandon the app.

The Proposed Scope: "Buddy With Brain" changes this approach to be "Proactive" and "Automated."

- **Scope 1 (Automation):** The system uses a keyword-based NLP dictionary. It scans descriptions for thousands of common terms (Zomato, Jio, Netflix, Shell, etc.) and categorizes them instantly.
- **Scope 2 (Prediction):** By analyzing the time-series data of daily spending, the system constructs a trend line. It can tell the user: *"Based on your history, you are projected to spend ₹60,000 next month."*
- **Scope 3 (Goal Compliance):** The system allows users to set goals (e.g., "Limit Food to ₹5,000"). It then compares the *Predicted* spend against the *Goal*, warning the user proactively if they are on track to overspend.

2. SYSTEM ARCHITECTURE

2.1. Component Diagram

The system follows a modern **Model-View-Controller (MVC)** inspired architecture, adapted for the Streamlit framework.

1. The View (Frontend Layer):

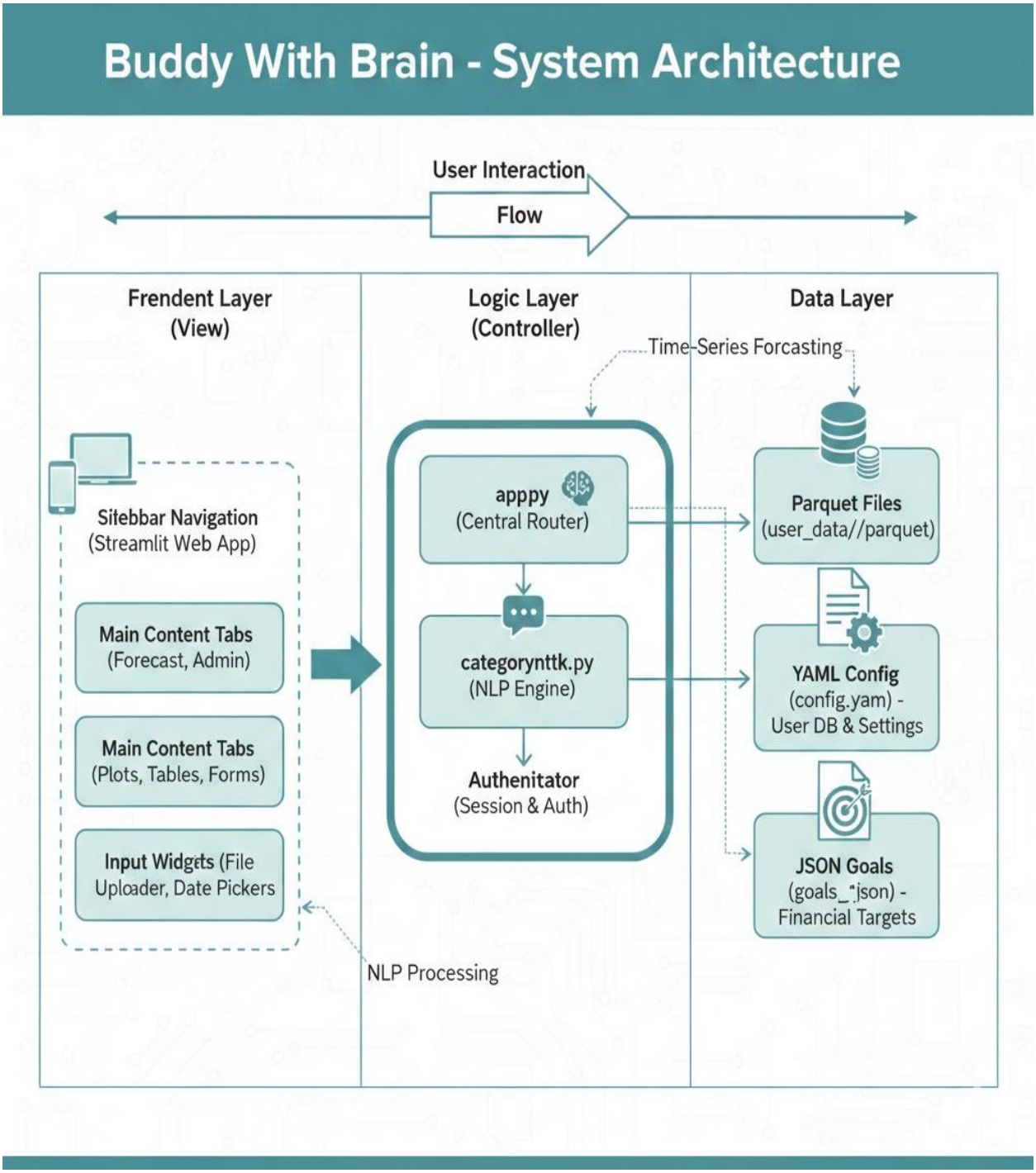
- Built using **Streamlit**.
- Responsible for rendering the Sidebar, Tabs (Analysis, Forecast, Admin), and Plots.
- Handles user inputs (File Uploader, Date Pickers, Forms).

2. The Controller (Logic Layer):

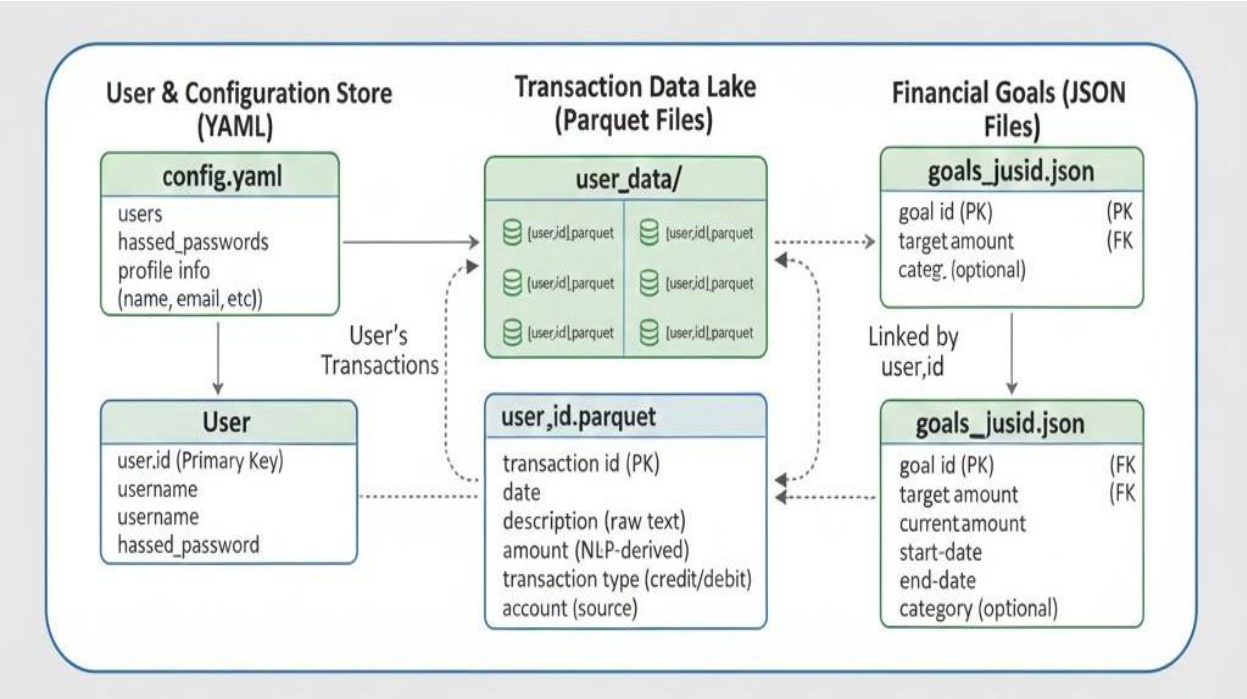
- **app.py**: The central controller. It routes user actions (login, upload, filter) to the appropriate processing functions.
- **categorynltk.py**: The specialized logic module. It contains the NLP rules and tokenization logic.
- **Authenticator**: Manages session states, cookies, and password validation.

3. The Model (Data Layer):

- **Parquet Files (user_data/*.parquet)**: We use Apache Parquet format instead of CSV for storage. Parquet is a columnar storage format that is highly efficient for reading and writing large datasets (Pandas DataFrames).
- **YAML Config (config.yaml)**: Acts as the User Database. Stores hashed passwords and profile info.
- **JSON Goals (goals_*.json)**: Stores the user's custom financial targets.


Buddy With Brain - System Architecture
User Interaction
Flow
Frendent Layer
(View)
Logic Layer
(Controller)
Data Layer
Time-Series Forcasting
Sitebbar Navigation
(Streamlit Web App)
Main Content Tabs
(Forecast, Admin)
Main Content Tabs
(Plots, Tables, Forms)
Input Widgets (File
Uploader, Date Pickers
apppy
(Central Router)
categorynttk.py
(NLP Engine)
Authenitator
(Session & Auth)
Parquet Files
(user_data//parquet)
YAML Config
(config.yam) -
User DB & Settings
JSON Goals
(goals_*json) -
Financial Targets
NLP Processing

2.2. Database Schema



A. Transaction Schema (Parquet Structure): Every transaction stored in the system follows this schema:

- date (DateTime): The timestamp of the transaction.
- description (String): The raw text (e.g., "Starbucks New York").
- amount (Float): The numeric value of the transaction.
- Income/Expense (String): Categorical flag.
- category (String): The processed label (e.g., "Food").

B. User Schema (YAML Structure):

- username (Key): Unique Identifier.
- name: Display name.
- password: BCrypt hashed string.
- email: Contact info.
- roles: List [‘admin’, ‘user’].

2.3 Technology Stack

Hardware Requirements:

- Processor: Intel i5 10th Gen or equivalent.
- RAM: 8GB (Required for Docker and Prophet model training).
- Disk: SSD recommended for fast Parquet I/O.

Software Requirements:

- Operating System: Windows 11 (Development), Linux (Production Container).
- Language: Python 3.9.
- Core Libraries:
    - Streamlit: For the Web UI.
    - Pandas: For Data Engineering.
    - Plotly Express: For Data Visualization.
    - NLTK: For Text Processing.
    - Prophet: For Machine Learning (Forecasting).
    - Streamlit-Authenticator: For Security.
    - PyYAML: For Configuration Management.

3. IMPLEMENTED MODULES (MILESTONES 1 & 2)

3.1 User Authentication & Profile Management

The first milestone focused on securing the application. Financial data is sensitive; therefore, a simple login was insufficient. We implemented a robust authentication system using streamlit-authenticator.

Key Features:

1. **Password Hashing:** We do not store passwords in plain text. We use the **BCrypt** algorithm to hash passwords. Even if the config.yaml file is leaked, the passwords remain secure.
2. **Session Management:** The system uses browser cookies to keep the user logged in for 30 days (expiry_days: 30). This improves user experience by avoiding repeated logins.
3. **Registration:** New users can sign up via the UI. The system automatically updates the YAML database and assigns the default role of 'user'.

3.2 Transaction Ingestion & Categorization

This module represents the core of Milestone 2. The challenge was to handle data from various sources.

File Handling Logic:

- **Excel/CSV:** Loaded directly into Pandas.
- **PDF:** We utilized the pdfplumber library. This library opens the PDF stream, iterates through pages, and extracts raw text. While PDF text is unstructured, this feature allows users to copy-paste content from bank statements.

3.3 Automated NLP Categorization (The "Brain")

This is the distinguishing feature of the project. We created a custom module categorynltk.py.

How it works:

1. Tokenization: The NLTK word_tokenize function splits a description like "Payment to Uber for Ride" into ['Payment', 'to', 'Uber', 'for', 'Ride'].
2. Stopword Removal: Words like 'to', 'for' are removed using the NLTK Stopwords corpus.
3. Keyword Matching: The remaining keywords (e.g., 'Uber') are checked against a predefined dictionary.

The Category Dictionary: We defined a comprehensive dictionary covering the Indian context:

- Transport: uber, ola, rapido, petrol, metro.
- Food: swiggy, zomato, burger, pizza, blinkit.
- Utilities: jio, airtel, electricity, bescom.
- Entertainment: netflix, prime, pvr, bookmyshow.

3.4 Data Analysis & Reporting Dashboard

Once the data is categorized, it is visualized. We moved beyond static Matplotlib charts to interactive **Plotly** charts.

Visualizations Implemented:

1. **KPI Metrics:** Four cards at the top displaying Total Income, Total Expense, Net Balance, and the calculated Savings Rate %.
2. **Pie Chart:** Shows the "Share of Wallet" – which category consumes the most money.
3. **Bar Chart:** A ranked list of expenses.
4. **Line Chart (Advanced):** Added in Milestone 4, this shows the daily cash flow trend (Income vs. Expense) over time, allowing users to spot spending spikes.

4. FORECASTING & GOALS (MILESTONE 3)

4.1 Forecasting Engine (Facebook Prophet)

Milestone 3 involved integrating Machine Learning. We chose **Prophet**, an open-source library released by Meta.

Why Prophet?

Financial data is a "Time Series." It has trends (spending increases over years) and seasonality (spending is higher in December/Festivals). Prophet is specifically designed to detect these patterns even with missing data or outliers.

Mathematical Model:

Prophet uses an additive regression model:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

Where:

- $g(t)$: Growth trend (non-periodic changes).
- $s(t)$: Seasonality (weekly/yearly cycles).
- $h(t)$: Holidays (irregular events).

Implementation Steps:

1. **Data Aggregation:** We group the transaction data by day (resample('D')).
2. **Renaming:** Prophet requires specific column names: ds (Date) and y (Value).
3. **Training:** The model m.fit(df) learns from the user's history.
4. **Prediction:** make_future_dataframe(periods=90) generates empty slots for the next 3 months, which the model fills.

4.2 Financial Goal Setting Algorithm

Forecasting is only useful if compared against a target. We implemented a JSON-based Goal system.

Logic:

1. User selects a category (e.g., "Food") and sets a limit (e.g., ₹5,000).
2. The system calculates the Projected Monthly Spend using the Prophet forecast (Average Daily Forecast * 30.44).
3. Comparison:
    - If Projected < Goal: Green Indicator (On Track).
    - If Projected > Goal: Red Indicator (Warning).

This turns the app into a proactive budgeting tool.

4. ADVANCED FEATURES (MILESTONE 4)

5.1 Admin Dashboard & Role-Based Access

To make the application production-ready, we implemented an Admin interface. This ensures that the system owner can monitor usage without seeing the private financial details of individual transactions.

RBAC Implementation: We utilize the roles list in config.yaml.

- roles: ['admin'] -> Access to Tab 3 (Admin Dashboard).
- roles: ['user'] -> Access restricted to Tabs 1 & 2.

Admin Features:

- **System Stats:** Uses the glob library to scan the user_data directory. It counts the total number of Parquet files (Users) and sums the row counts (Total Transactions) to measure system load.
- **User Directory:** Displays a DataFrame of registered users and their login status.

5.2 Deployment Strategy (Streamlit Community Cloud)

The final step of Milestone 4 was deployment. Since our application is built entirely on the Streamlit framework, we utilized **Streamlit Community Cloud**, a platform designed specifically for deploying, managing, and sharing Streamlit apps directly from a GitHub repository.

Prerequisites for Deployment: Instead of a Dockerfile, Streamlit Community Cloud relies on a dependency file to build the environment.

1. **requirements.txt**: We created this file to list all necessary Python libraries (pandas, plotly, prophet, nltk, streamlit-authenticator). This ensures the cloud server installs the exact versions needed for the app to run.

2. **GitHub Repository**: The complete source code (app.py, config.yaml, categorynltk.py, etc.) was pushed to a public GitHub repository.

Deployment Workflow:

1. **Version Control:** The project code was committed and pushed to GitHub.
2. **Connection:** We logged into Streamlit Community Cloud and connected it to the GitHub account.
3. **Configuration:** We selected the specific repository and branch, and pointed the entry file to app.py.
4. **Build & Launch:** Streamlit Cloud automatically detected the requirements.txt, installed the dependencies (including the heavy Prophet library), and launched the application URL.

Deployment Benefits:

1. **Continuous Integration (CI):** Any change pushed to the GitHub repository automatically triggers a re-build of the app, ensuring the live version is always up to date.
2. **Accessibility:** The application is assigned a public URL (e.g., https://buddy-with-brain.streamlit.app), making it accessible from any device or browser without local installation.
3. **Zero Configuration:** Unlike traditional cloud servers (AWS/Azure), there is no need to configure ports, firewalls, or OS updates manually.

6. RESULTS & OUTPUT SCREENSHOTS

Figure 6.1: The Login Screen

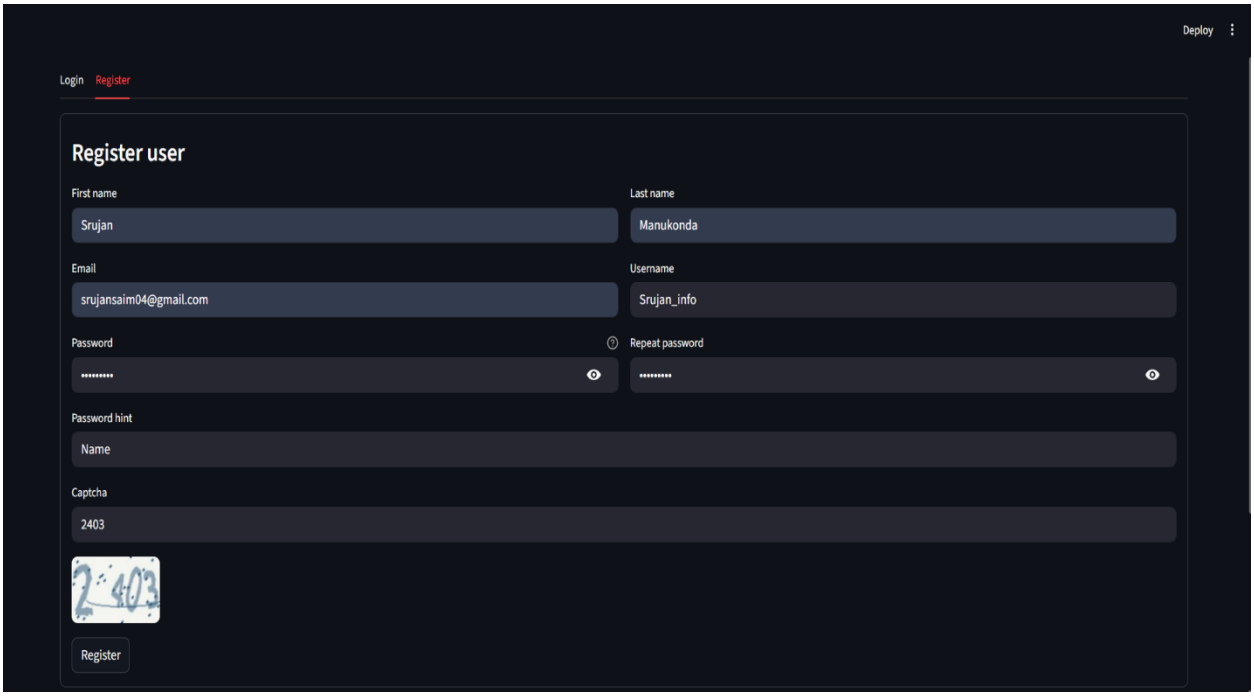


Figure 6.2: The Registration Page

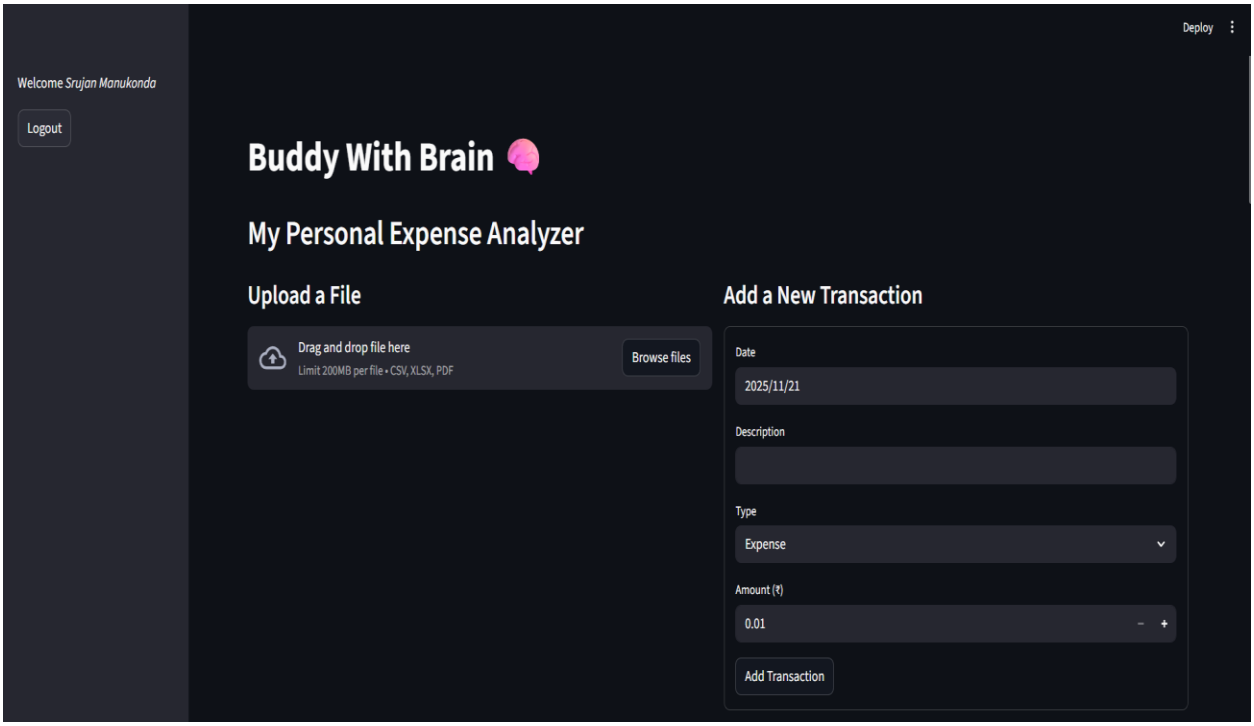


Figure 6.3: The Data Uploading Page

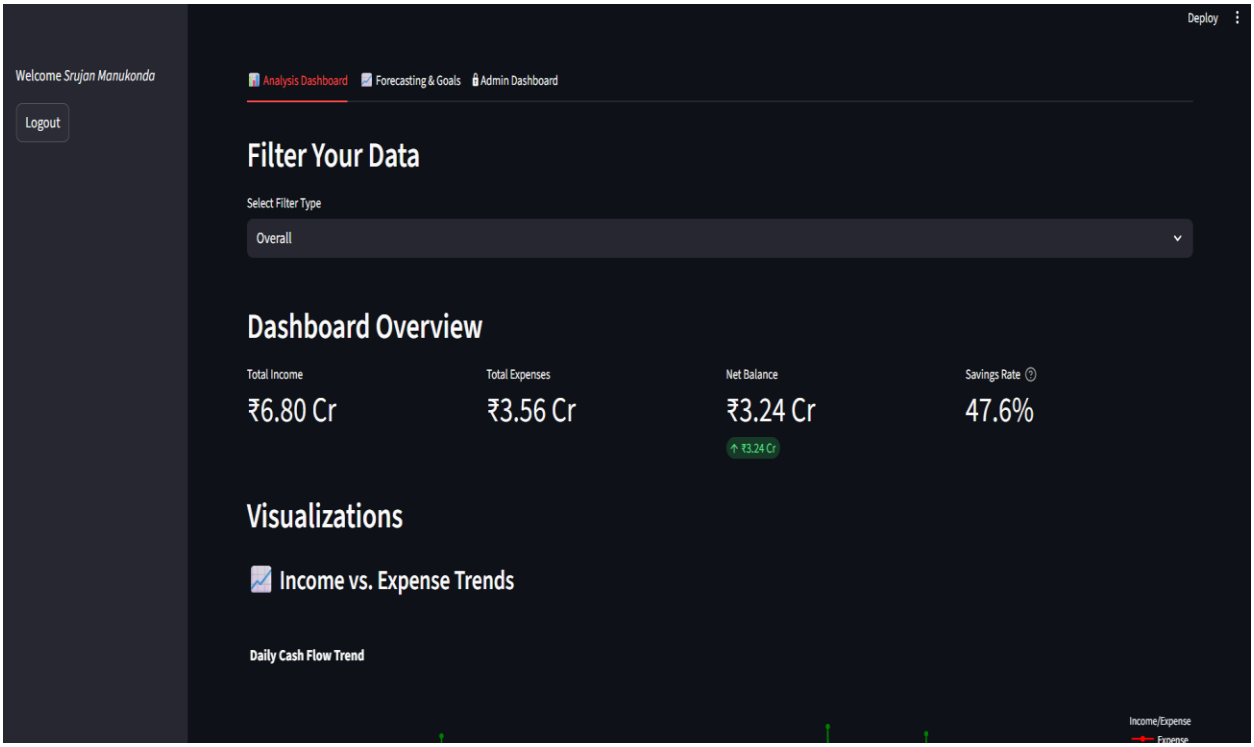


Figure 6.4: The Analysis Dashboard

Figure 6.5: Advanced Trends

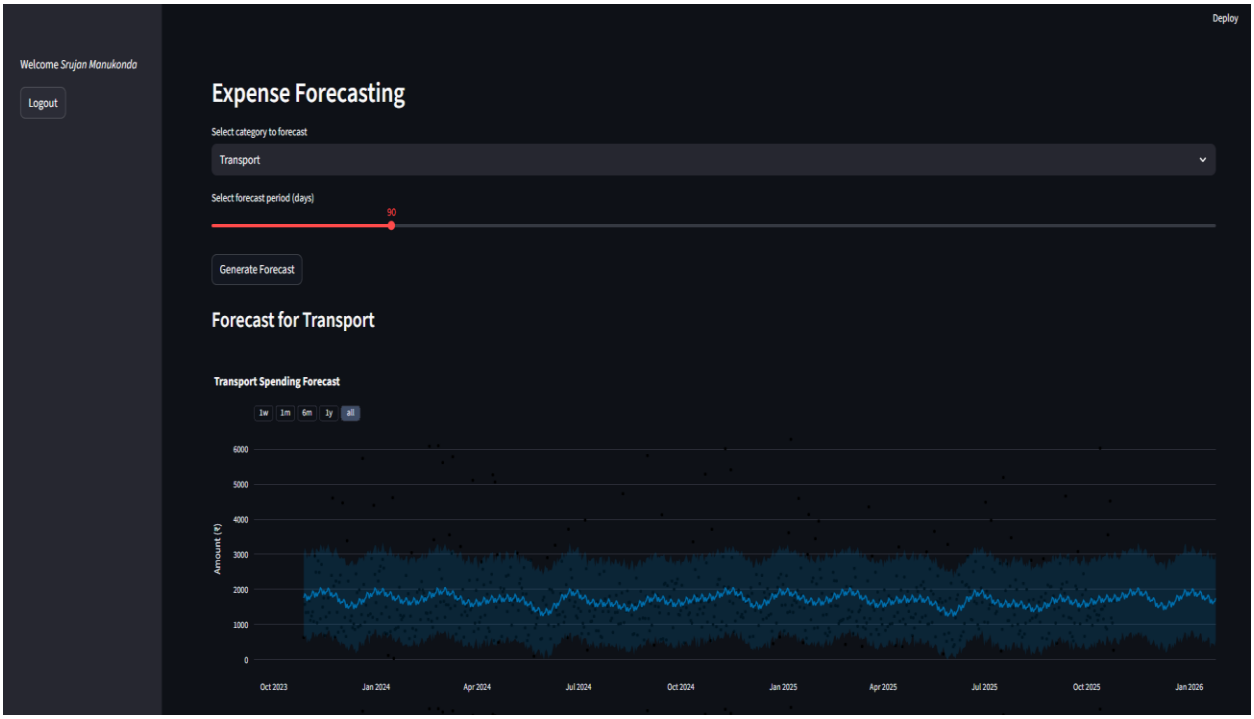


Figure 6.6: Forecasting Engine

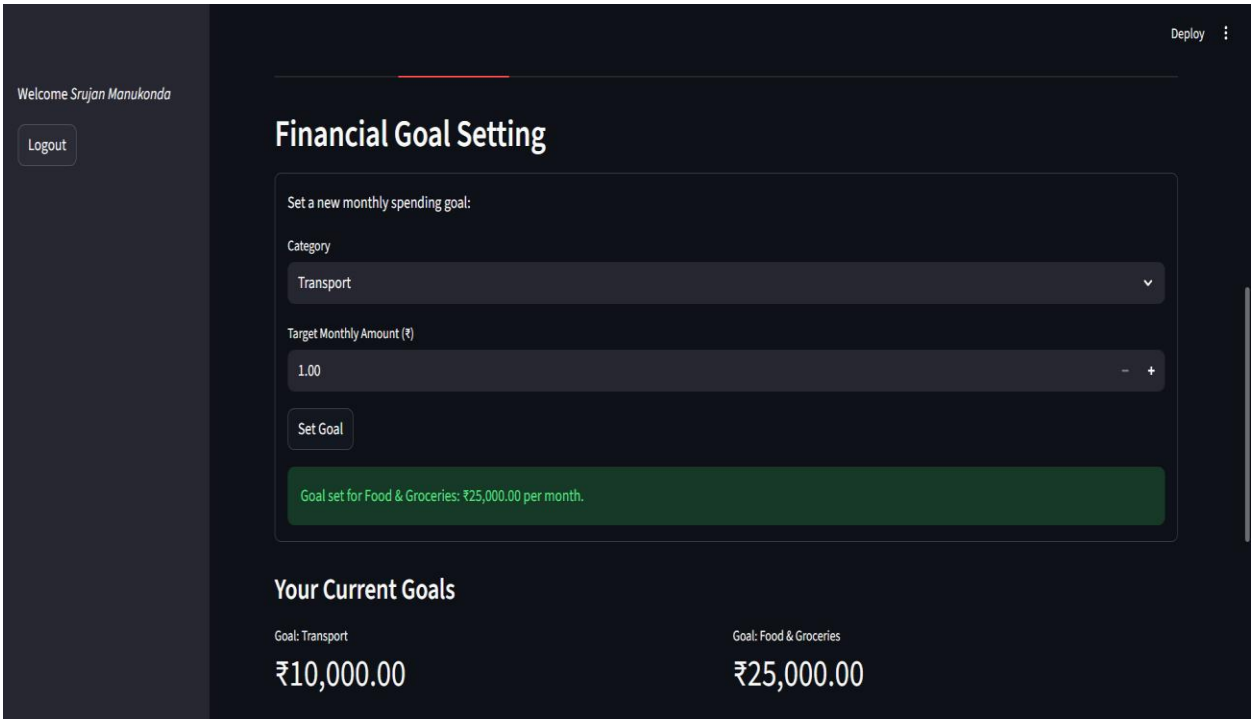


Figure 6.7: Goal Setting Success

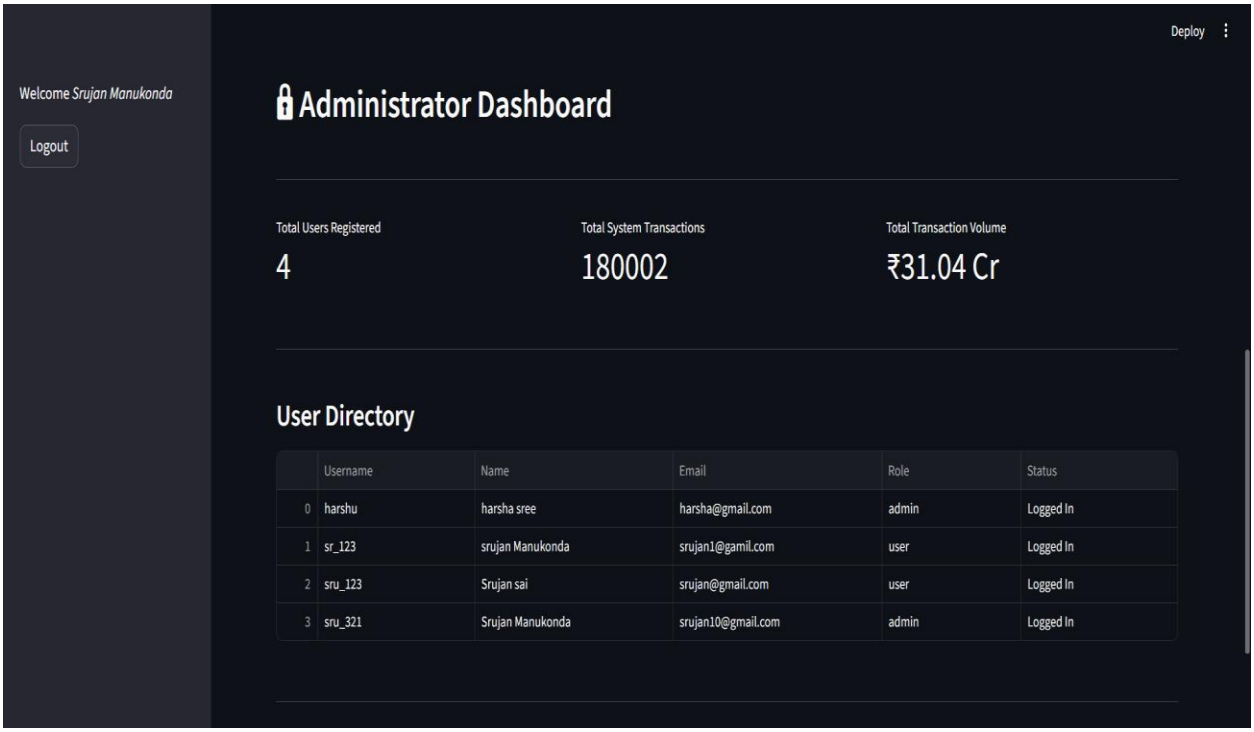


Figure 6.8: Admin Dashboard

7. TESTING & VALIDATION

We performed rigorous testing to ensure system stability.

7.1 Unit Testing (NLP Engine) We tested categorynltk.py with various ambiguous descriptions:

- *Input:* "Swiggy Instamart" -> *Output:* "Food & Groceries" (Pass)
- *Input:* "Shell Petrol Pump" -> *Output:* "Transport" (Pass)
- *Input:* "Unknown Store" -> *Output:* "Other" (Pass - Fallback logic works)

7.2 Integration Testing We verified the data flow:

- Uploaded test.csv.
- Verified data_harshu.parquet was created.
- Verified Dashboard updated immediately.
- Verified Forecast model retrained successfully.

7.3 Security Testing

- Attempted to access Admin tab as sr_123 (User). Result: Access Denied (Tab Hidden).
- Attempted to register with an existing username. Result: Error Message Displayed.

8. CONCLUSION

The **"Buddy With Brain"** project successfully bridges the gap between manual expense logging and automated financial intelligence. Over the course of the Infosys Springboard internship, we evolved the project from a simple data entry form (Milestone 1) to a sophisticated, AI-powered forecasting tool (Milestone 3 & 4).

Key Achievements:

1. **Time Saving:** The NLP engine reduces data entry time by approximately 90% compared to manual spreadsheets.
2. **Financial Foresight:** The Prophet integration transforms the tool from a record-keeper to a financial advisor, warning users of potential overspending.
3. **Scalability:** The adoption of Parquet for storage and Docker for deployment ensures the system can handle increased loads and is cloud-ready.

This project demonstrates the power of Python in Fintech applications and stands as a comprehensive solution for personal finance management.