 Personal Finance Tracker

Submitted to Manipal University Jaipur

Towards the partial fulfilment for the Award of the Degree of

**BACHELOR OF TECHNOLOGY**

In Computers Science and Engineering

2025-2026

By

**Piyush Prakash Sharma  
2427030222**



Under the Guidance of

**Ms.Harshika Mathur**

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

Sign of Supervisor

**Department of Computer Science and Engineering**

**School of Computer Science and Engineering**

**Manipal University Jaipur**

**Jaipur, Rajasthan**

## I. Introduction to Problem (What, Why, How)

### The "What": The Problem of Personal Financial Management

When a person tries to manage their finances with the help of some web or mobile app they are met with possibilities of data leaks and enormous paywalls and NO normal Person wants to go through the hassle of transforming raw, unstructured financial transaction data provided by the banks usually in form of CSV files into useful, categorized ,,high-level financial insights .Information which would provide invaluable information on their spending habits ,financial spending and flow of their earnings etc.

### The "Why": The Failure of Manual Methods

A significant gap exists in the current software market.

1. **Commercial SaaS (e.g., YNAB, Quicken):** These tools are easy to use but require recurring subscription fees and significant data privacy compromises (linking live bank accounts to the cloud).
2. **Open-Source (e.g., Firefly III):** These solutions are free and private but are built for technical users, requiring skills in server self-hosting, database management, and scripting.

### The "How": Fragmentation of the Modern PFM Market

The proposed solution is the **Finance Tracker**, an **Application based** tool. It is a lightweight, offline Python application that operates *only* on a local, user-provided CSV file. It automates the most laborious tasks (data cleansing, categorization, and report generation) and outputs the final analysis into a universally familiar Microsoft Excel file.

**II. Literature Survey**

A survey of 10 existing solutions reveals the specific trade-offs in the current market.

### A. Cloud-Based Subscription Platforms

* **YNAB (You Need A Budget)**
  + **Pros:** Excellent zero-based budgeting system.
  + **Cons:** High subscription cost ($109/yr) and steep learning curve.
* **Monarch Money**
  + **Pros:** Comprehensive dashboard for net worth and investments.
  + **Cons:** Expensive ($99/yr) and fully cloud-dependent.
* **Quicken Simplifi**
  + **Pros:** User-friendly interface and good reporting.
  + **Cons:** Requires recurring subscription; less powerful than desktop predecessors.
* **Copilot Money**
  + **Pros:** Uses Machine Learning for auto-categorization; excellent UI.
  + **Cons:** Exclusive to Apple (iOS/Mac) ecosystem and US banks.
* **Rocket Money**
  + **Pros:** Specializes in identifying and cancelling subscriptions.
  + **Cons:** Privacy concerns regarding data access permissions.

### B. Freemium Cloud Services

* **Empower Personal Dashboard**
  + **Pros:** Best-in-class for investment and net worth tracking (free).
  + **Cons:** Budgeting tools are rudimentary; primarily a lead-gen tool for advisors.
* **Wallet by BudgetBakers**
  + **Pros:** Extensive international bank support.
  + **Cons:** Bank synchronization is often unstable; free version is heavily restricted.

### C. Spreadsheet & Local Alternatives

* **Tiller Money**
  + **Pros:** Pipes data directly into Google Sheets for maximum customization.
  + **Cons:** Requires high spreadsheet proficiency; paid subscription.
* **GnuCash**
  + **Pros:** Free, open-source, local-first privacy, professional double-entry accounting.
  + **Cons:** Extremely complex UI; strictly manual data entry.
* **HomeBank**
  + **Pros:** Free, lightweight, and simpler than GnuCash.
  + **Cons:** Lacks automation; relies entirely on manual file imports.

## III. Comparative Study

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Software** | **Hosting** | **Cost** | **Data Import** | **Key Feature** | **Key Weakness** |
| **YNAB** | Cloud | High ($109/yr) | API Sync | Zero-Based Budgeting | Expensive |
| **Monarch** | Cloud | High ($99/yr) | API Sync | All-in-One Dashboard | Cloud Privacy Risk |
| **Simplifi** | Cloud | Medium | API Sync | Reporting | Subscription Model |
| **Empower** | Cloud | Free | API Sync | Wealth Tracking | Weak Budgeting |
| **Copilot** | Cloud | High | API Sync | ML Categorization | Mac/iOS Only |
| **Rocket** | Cloud | Freemium | API Sync | Sub. Management | Privacy Concerns |
| **Tiller** | Cloud | Medium | API Sync | Custom Sheets | High Difficulty |
| **GnuCash** | Local | Free | Manual | Double-Entry Books | Steep Learning Curve |
| **HomeBank** | Local | Free | Manual | Lightweight | No Automation |
| **LocalFinTracker** | **Local** | **Free** | **CSV Import** | **Private Automation** | **Static Categories** |

## IV. Problem Statement

To track their expenses, Users currently face only 2 choices:

Sacrifice privacy for the convenience of cloud-based automation.

OR

Accept the tedious process of manual entry for the security of offline software.

There is no streamlined, free tool that automates financial analysis from bank statements while guaranteeing 100% local data sovereignty.

## V. Objective

1. **Develop an Automated Program/Software:** Create a Python application using pandas to clean and categorize raw bank CSV data into meaningful insights.
2. **Generate Visual Reports:** Utilize openpyxl to auto-generate Excel dashboards with summary tables and pie charts for immediate user feedback.
3. **Ensure Data Sovereignty:** Validate a local-execution model where all processing occurs on the client machine, eliminating third-party data risks.

## VI. Planning of Work / Proposed Solution

### 6.1 Current Implementation (Methodology)

The current system is a functional Python script executing a linear data pipeline:

1. **Input :** The script accepts a raw CSV bank statement.
2. **Cleaning:** The clean\_amount\_column function sanitizes currency strings (removing '₹', commas) and standardizes data types.
3. **Categorization:** A keyword-matching algorithm maps transaction descriptions to categories (e.g., "Swiggy" → "Food") using a predefined dictionary.
4. **Reporting:** The openpyxl library generates a structured .xlsx file containing a transaction log, a category summary table, and an embedded Pie Chart.

### Future Enhancements

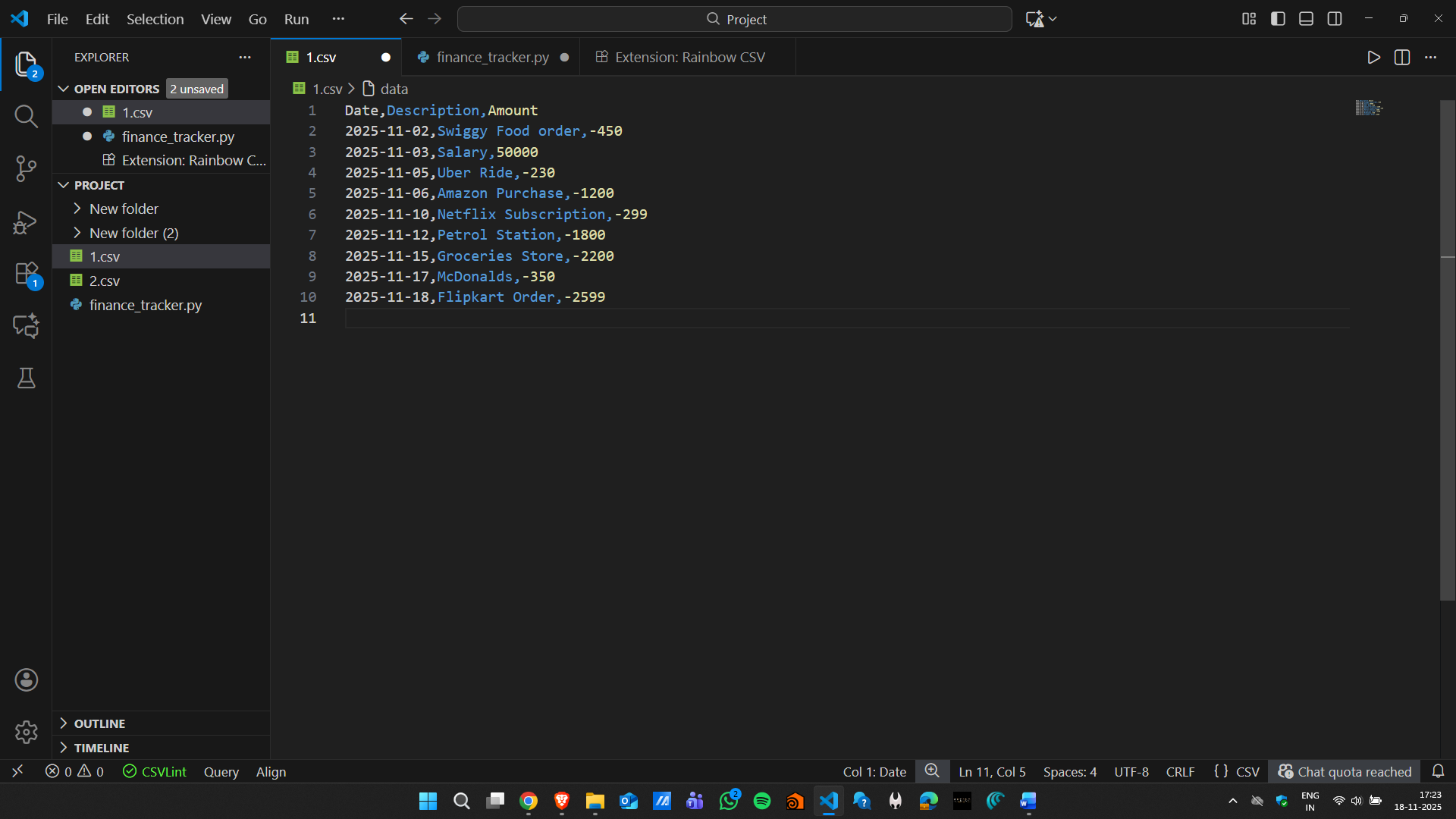
To expand beyond the current prototype, the project will integrate API and ML capabilities:

* **Phase 1: API Integration (Plaid):** Replace manual CSV downloads with the Plaid API. The application will fetch transactions directly using a locally stored access token, maintaining the "local-first" architecture while adding sync convenience.
* **Phase 2: Intelligent ML Categorization:** Replace static keywords with a scikit-learn Naive Bayes classifier. The system will learn from user corrections, creating a personalized model that improves over time without sharing data.
* **Phase 3: Predictive Forecasting:** Implement time-series forecasting to predict future spending trends based on historical data, alerting users to potential budget overruns.

## VII. Screenshots

*(Placeholder for visual evidence)*

* **Figure 1:** Raw CSV Input File.
* **Figure 2:** Python Execution Console.
* **Figure 3:** Generated Excel Report (Summary & Pie Chart).



A computer screen shot of a computer screen

AI-generated content may be incorrect.A screen shot of a computer

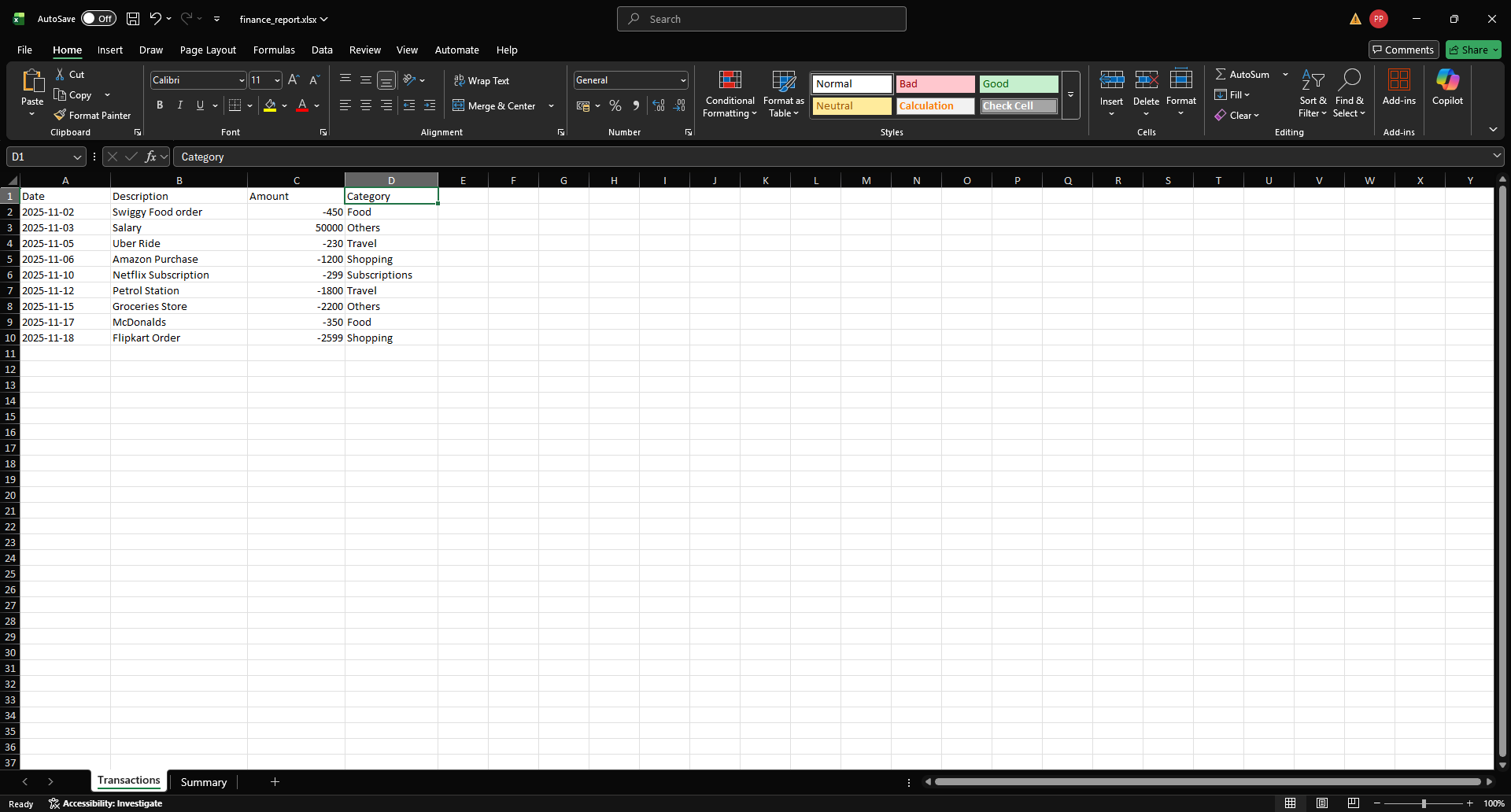
AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

## A screen shot of a computer AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## VIII. Bibliography

1. **Pandas Development Team.** (2024). *pandas: powerful Python data analysis toolkit*. Available: https://pandas.pydata.org
2. **Gazoni, E. & Clark, C.** (2024). *openpyxl: A Python library to read/write Excel 2010 xlsx/xlsm files*. Available: https://openpyxl.readthedocs.io
3. **Pedregosa, F. et al.** (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
4. **Plaid Inc.** (2024). *Plaid API Documentation*. Available: https://plaid.com/docs/
5. **You Need A Budget (YNAB).** (2024). *The Four Rules of Budgeting*. Available: https://www.ynab.com/the-four-rules