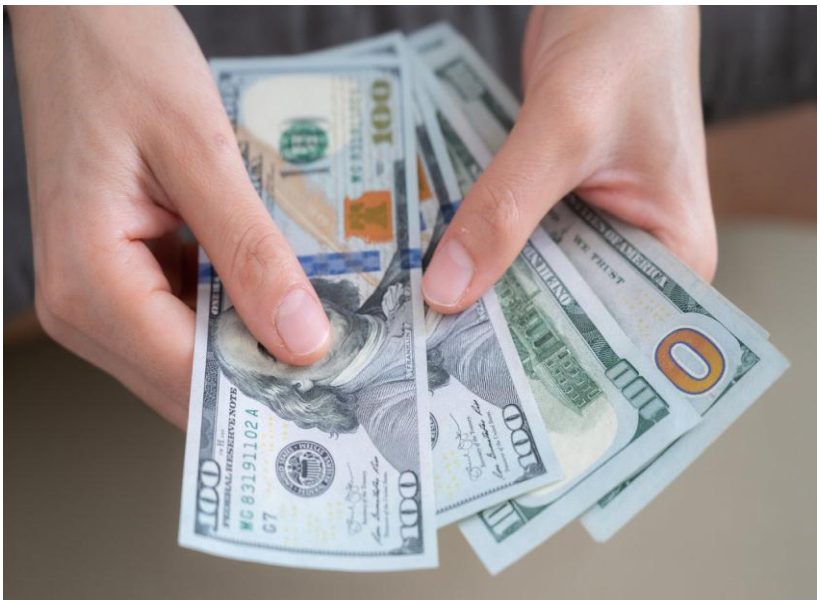


PERSONAL EXPENSE TRACKER

Manage your money smartly

ABSTRACT

A detailed project report on the program made to manage your expenses efficiently

Akshat Singh

BTECH(CSE CORE)-- (25BCE10857)

2. Introduction

Managing day-to-day expenses is an important part of personal finance. Many people, especially students like me, either track their spending manually or rely on heavy mobile applications that provide more features than they actually need. This often leads to irregular tracking and poor visibility into where their money is going.

This project presents a command-line based Expense Tracker built using Python. It enables users to record their expenses, organize them into categories, view historical data, generate statistics, and visualize spending through charts. All data is stored locally in a CSV file, making the application lightweight, portable, and easy to understand from a programming perspective.

The project also demonstrates core programming concepts such as file handling, data structures, modular design, input validation, and basic data visualization using Matplotlib.

3. Problem Statement

Users, especially students and individuals with limited budgets, often lack a simple offline tool to track and analyze their expenses. Manual methods (notebooks, loose notes, basic calculators) become difficult to maintain and do not provide any analysis or visual feedback. Full-featured finance apps may be too complex, require internet access, or raise privacy concerns.

There is a need for a **simple, local, command-line application** that:

- Records expenses with essential details
- Stores data permanently in an accessible and simple format
- Provides summary statistics and basic analytics
- Visualizes spending in an easy-to-understand way that everybody can refer to

4. Functional Requirements

The system provides the following functional modules:

1. **Add_Expense()**
    - This function Accepts date, category, amount, and note from the user.
    - Validates date format and numeric amount.
    - Appends the record to the in-memory list and updates the CSV file.
2. **View_Expenses_with_index()**
    - This function Displays all expenses with an index number for the ease of user.

- It also Shows date, category, amount, and note in a formatted list.

3. **Remove_Expense()**
    - Takes index as input from the user which Allows the user to delete an expense by index.
    - Updates the in-memory list and rewrites the CSV file.
4. **View_by_Category()**
    - Filters expenses for a given category (case-insensitive).
    - Displays only matching entries.
    - This function makes the user able to see how much he is spending on a particular category.
5. **View_by_Date_Range()**
    - Accepts start and end dates from the user i.e, a range.
    - Then it Displays all expenses whose date lies within the range.This gives an idea of spendings in like a month , 2 months or even a year.
6. **Statistics Module**
    - Calculates total amount spent by the user.
    - Calculates average spending of the user per entry.
    - Computes amount spent per category and also number of expenses per category.
    - Identifies the highest spending category so the user can know where most of his money goes.
    - Also shows monthly totals i.e. ; amount of money spent by the user per month
7. **Bar Chart Visualization**
    - Generates a bar chart of category-wise total spending using Matplotlib . Helps visualize the spendings of the user with the help of chart.
8. **Persistent Storage**
    - Reads and writes data from/to data/expenses.csv.
    - Creates the file with correct headers if it does not exist.

9. There are other modules as well such as the get_category_stats() , get_monthly_totals().These modules are being used in other modules for computing the necessary requirements.

---

5. Non-Functional Requirements

1. **Usability**
    - Menu-driven interface with clear options.A user friendly interface
    - Meaningful prompts and error messages, So that the user can know then the inputs are invalid.
2. **Reliability**
    - Input validation for dates and amounts to avoid crashes.
    - Safe handling of missing or empty CSV files.
3. **Maintainability**
    - Code organized into functions (load, save, add, delete, filter, stats, plot).
    - Variables and functions use descriptive names.
4. **Performance**
    - Suitable for hundreds to a few thousand records since data is kept in memory.
    - CSV reading and writing kept efficient and minimal.
5. **Portability**
    - Runs on any system with Python 3 and Matplotlib installed.
    - Uses standard libraries for file handling.
6. **Data Integrity**
    - CSV file always written with a header row.
    - Overwrites the file only with the updated in-memory list.

6. System Architecture

The architecture is simple and modular:

- **User Interface Layer**
    - Command-line menu in main()

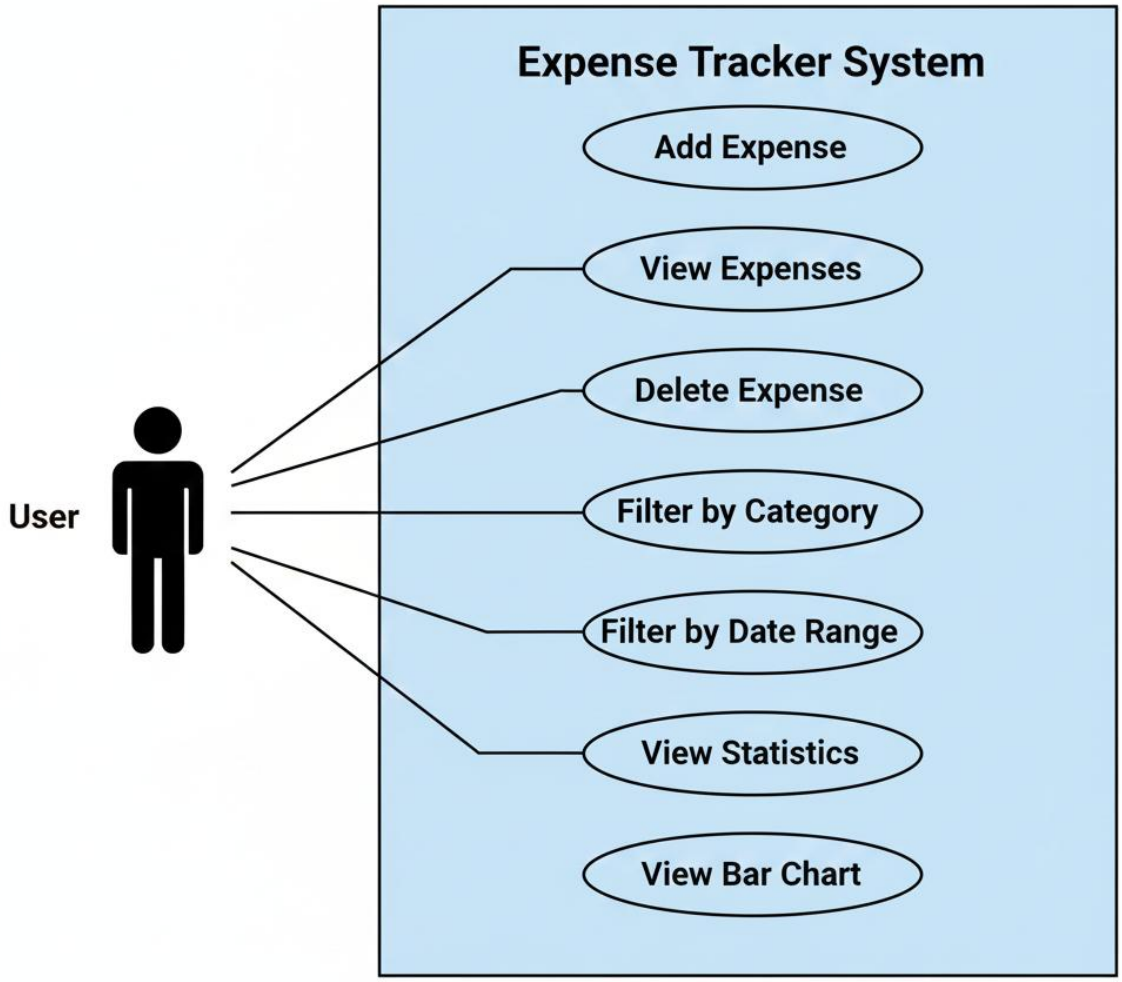
- Prompts the user and reads input

- **Application Logic Layer**
  - Functions: add_expenses(), remove_expense(), view_by_category(), view_by_date_range(), show_statistics(), plot_category_bar_chart()
  - Implements all operations on the expense list
- **Data Access Layer**
  - load_expenses() reads from data/expenses.csv using csv.DictReader
  - save_expenses() writes to data/expenses.csv using csv.DictWriter
- **Visualization Layer**
  - Uses Matplotlib to generate bar charts from processed statistics.

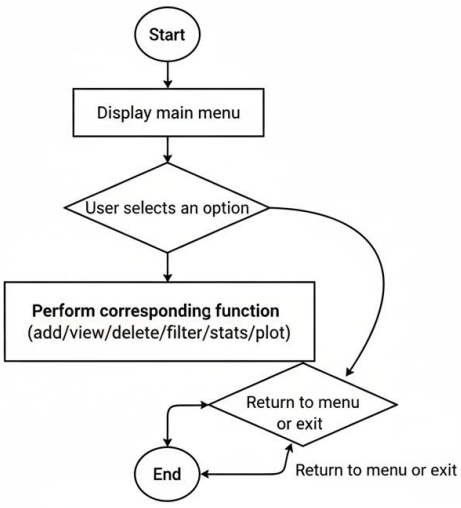
---

7. Design Diagrams

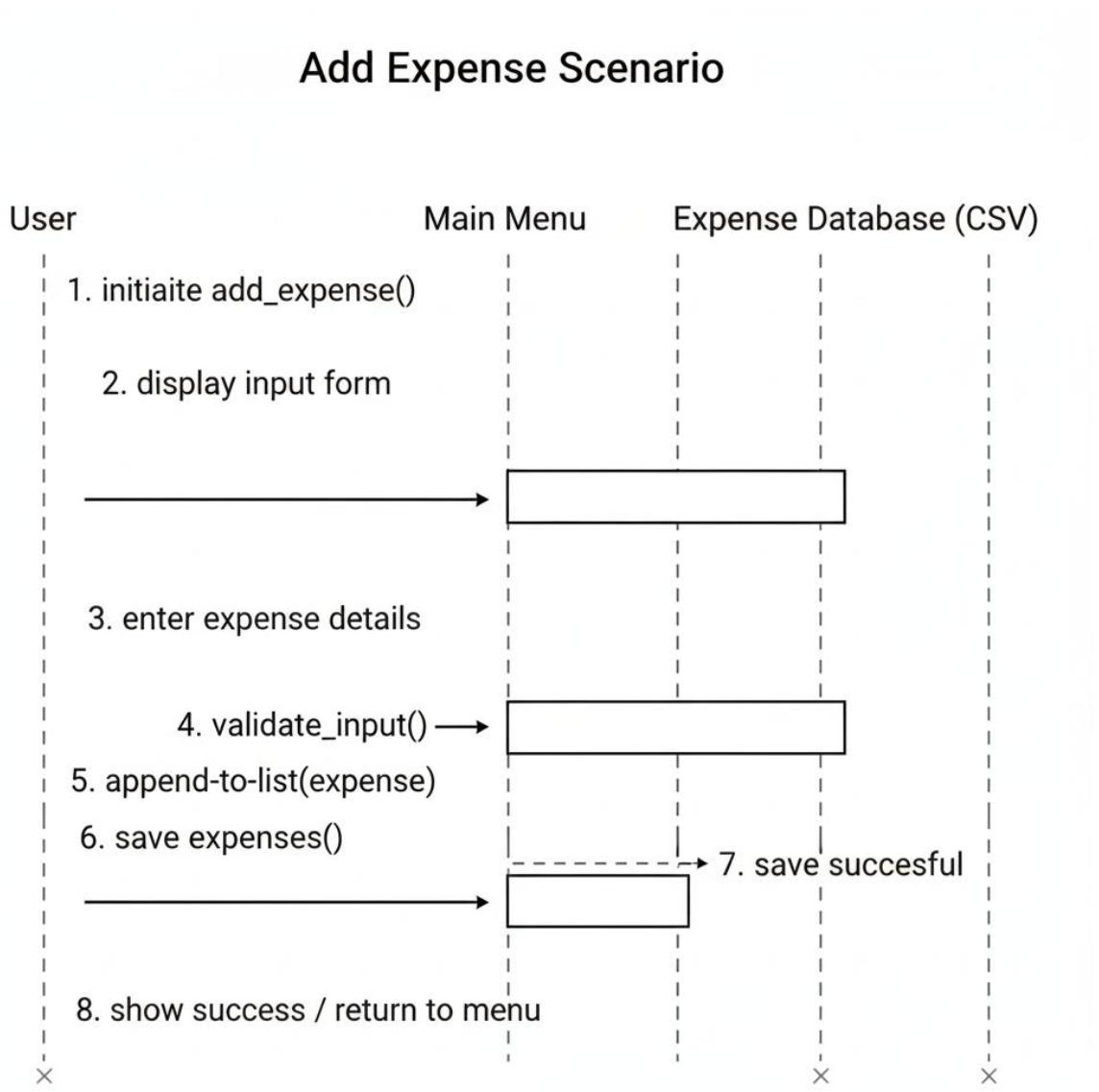
7.1 Use Case Diagram



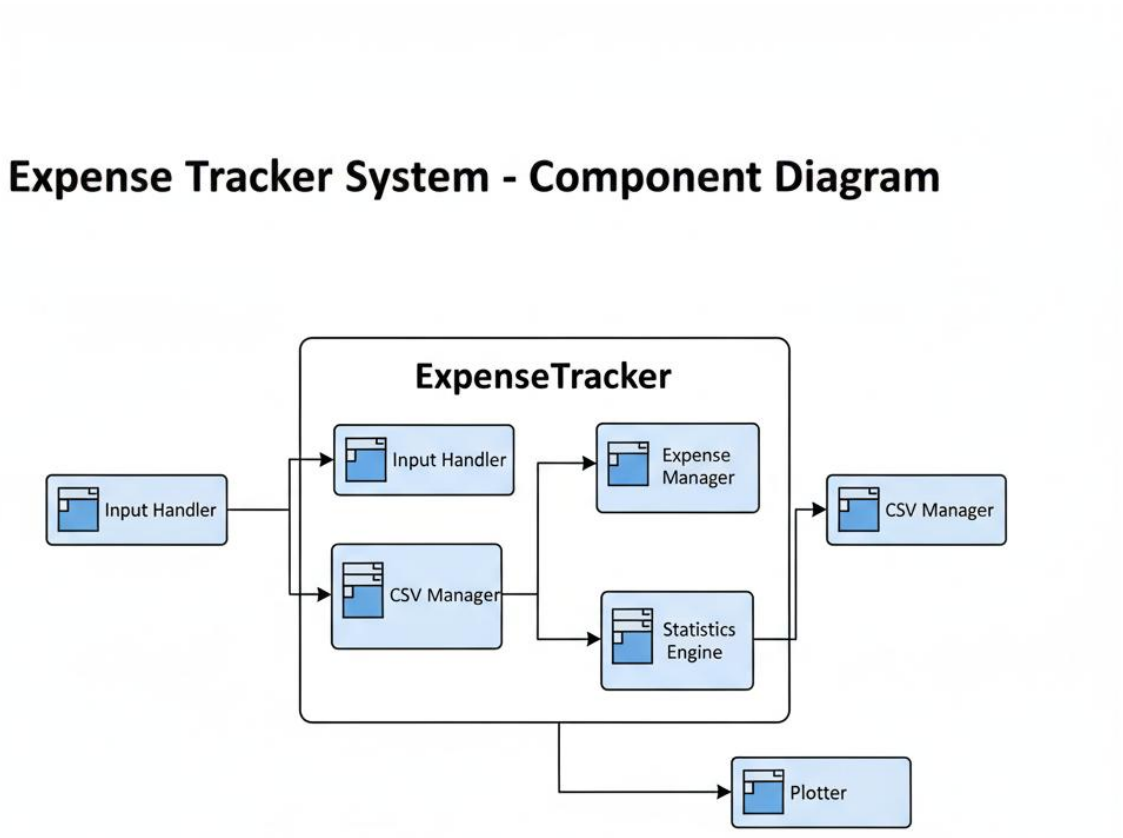
7.2 Workflow Diagram



7.3 Sequence Diagram

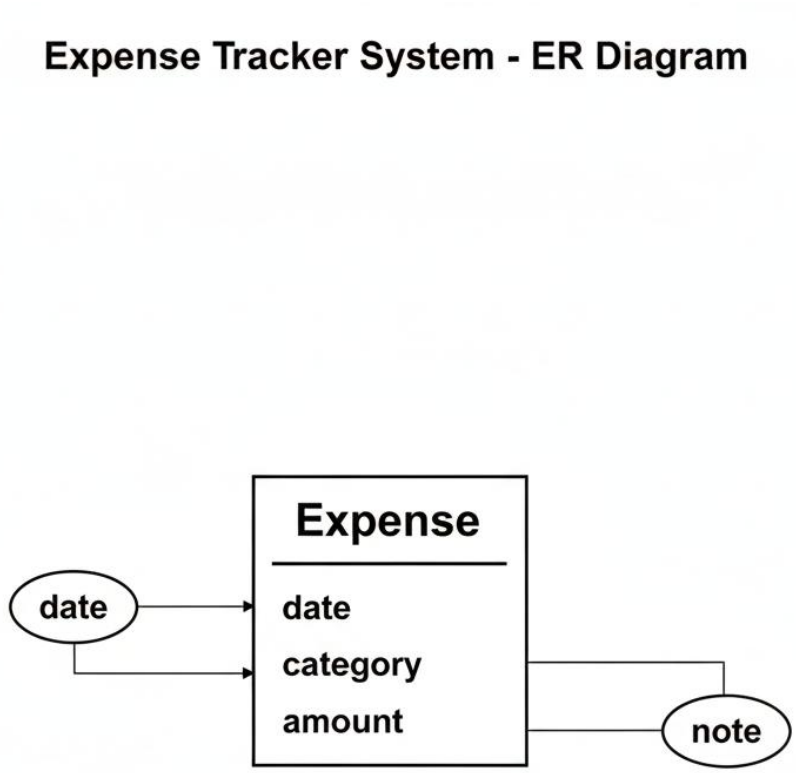


7.4 Class / Component Diagram



7.4 Class / Component Diagram

7.5 ER Diagram (if storage used)



7.5 ER Diagram

8. Design Decisions & Rationale

Python as the Language:

Chosen due to its simplicity, rich standard library, and suitability for scripting and data processing.

CSV as Storage Format:

→Human-readable and easy to inspect.

→Can be opened in Excel or Google Sheets.

→imple to handle with csv module.

→Command-Line Interface:

→Lightweight and portable.

→No need for complex frontend or GUI.

→Matches course focus on core programming concepts.

Modular Function Design:

Each operation (add, delete, filter, stats, plotting) has its own function.

Which makes it easier to maintain, test, and extend.

Matplotlib for Visualization:

Standard plotting library in Python.

Provides quick visual summaries without additional infrastructure.

9. Implementation Details→

Key modules and functions:

Data Handling:

~load_expenses(): Reads CSV file into a list of dictionaries named 'expenses' which is used to make changes in the file.

~save_expenses(expenses): Writes the full list back to CSV.

Core Operations:

~add_expenses(expenses): Takes inputs from the user in categories like date,category,amount,note.Also makes sure the user doesn't enter invalid entries

~view_expenses_with_index(expenses): Highlights each expense with an index making it easier for the user to keep track of the expenses

~remove_expense(expenses): Takes input from the user about the index that is going to be removed

~view_by_category(expenses): Takes input of category from the user and returns the total expenditure on that particular category

~view_by_date_range(expenses):Takes a date range from the user as input and return expenses in between that time frame

Analytics:

~get_category_stats(expenses) – returns totals and counts per category.

~get_monthly_totals(expenses) – aggregates by YYYY-MM.

~show_statistics(expenses) – orchestrates all calculations and prints output.

Visualization:

~plot_category_bar_chart(expenses) – generates a bar chart using category totals.

Main Function:

~main() – loads data, shows menu, dispatches to corresponding functions based on user choice.

10. Screenshots / Results→

Here are some screenshots attached that show output of the program and how the interface works for the user

Main menu in the terminal.

```
********** EXPENSE TRACKER **********
Welcome to my program.This is the Menu and here are your Options:
1. Add expense
2. View expenses
3. Delete expense
4. View by category
5. View by date range
6. Show statistics
7. Plot category bar chart
8. Exit
Enter your choice:
```

Output of “Add expense”

```
Enter your choice: 1
Enter date (YYYY-MM-DD): 2025-12-13
Enter category: (e.g.; {Food,Shopping,Date,Party,etc})Food
Enter amount: 450
Enter note: Mc.donalds
Expense Successfully Added
```

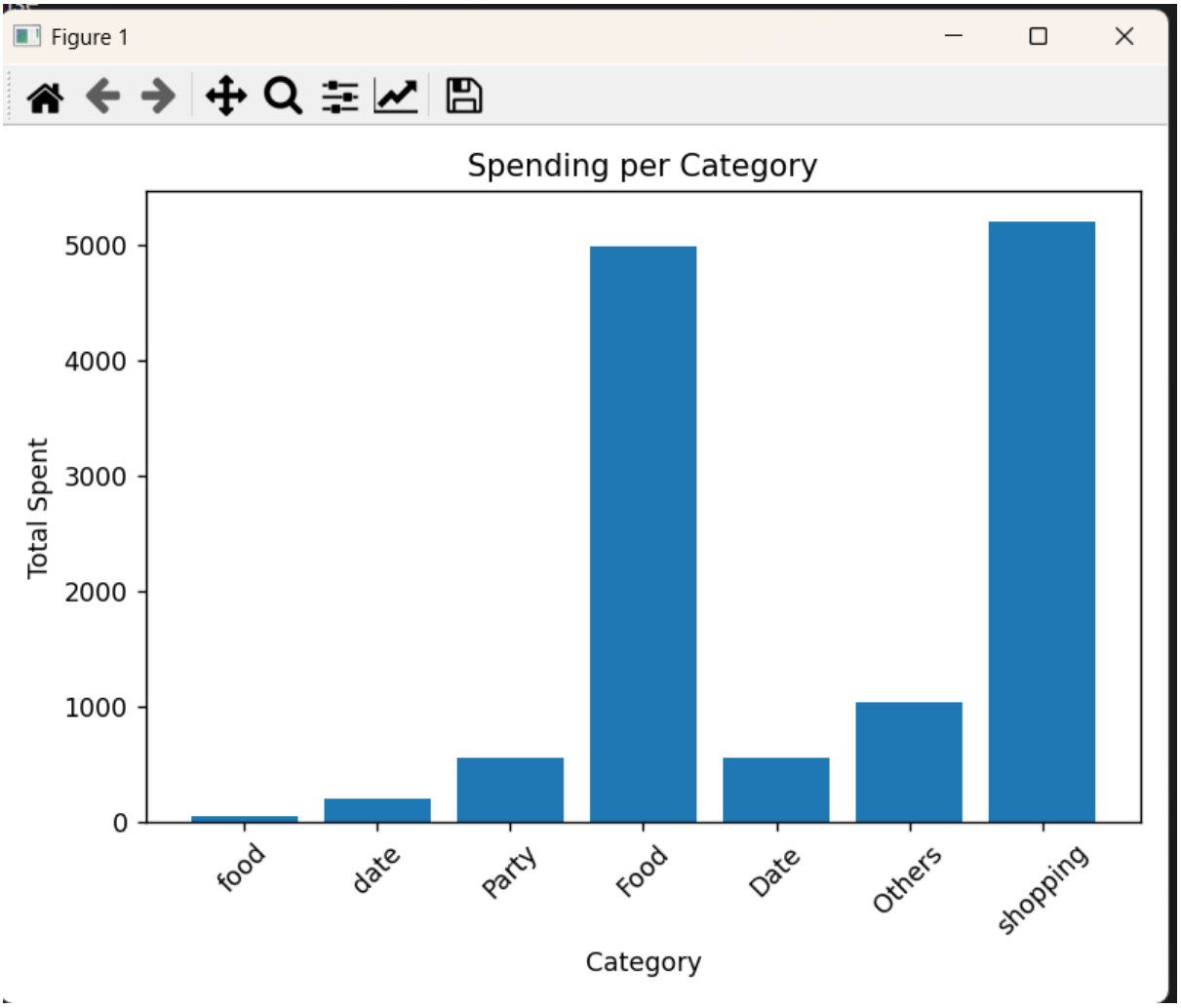
Output of “View expenses”.

```
Enter your choice: 2

Expenses List:
-----------------------------------------
0. 2025-9-6 | food | 55 | dinner
1. 2025-8-4 | date | 205 | ....
2. 2025-4-12 | Party | 562.0 | had fun
3. 2025-2-6 | Food | 4536.0 | Dinner with family
4. 2025-12-12 | Date | 563.0 | Noice
5. 2025-8-23 | Others | 1031.0 | kdkjsk
6. 2025-8-13 | shopping | 5200.0 | jeans
7. 2025-12-13 | Food | 450.0 | Mc.donalds
```

Output of “Statistics”.

```
Enter your choice: 6
********** STATISTICS **********
Total amount spent: 12602.0
Average spending per expense: 1575.25

Amount & count per category:
  food: 55.0 (entries: 1)
  date: 205.0 (entries: 1)
  Party: 562.0 (entries: 1)
  Food: 4986.0 (entries: 2)
  Date: 563.0 (entries: 1)
  Others: 1031.0 (entries: 1)
  shopping: 5200.0 (entries: 1)
Highest spending category is : shopping (5200.0)
Monthly totals (YYYY-MM):
  2025-09: 55.0
  2025-08: 6436.0
  2025-04: 562.0
  2025-02: 4536.0
  2025-12: 1013.0
**************************************
```

Bar chart window displaying category-wise spending.



11. Testing Approach

Testing was mainly done through manual test cases:

--Adding valid and invalid expenses (wrong date, non-numeric amount).

--Deleting first, middle, and last record.

--Filtering by category with no matches and with multiple matches.

--Filtering by date ranges (inside, outside, and overlapping).

--Verifying statistics manually for small datasets.

--Checking that CSV file contents match the expected data after each operation.

12. Challenges Faced:

--Ensuring that the CSV file is not overwritten incorrectly and data remains persistent.

-Handling invalid user input without crashing the program.

-Designing a clean menu structure and avoiding deeply nested code.

-Understanding Matplotlib usage for simple but readable charts.

13. Learnings & Key Takeaways:

-Practical understanding of Python file handling with CSV.(especially in the starting where csv was being written over again and again, which was later solved by me)

-Use of dictionaries and lists to manage structured data.

-Importance of input validation and error handling for user-facing programs.

-Basic data analysis (grouping, aggregation, and statistics).

-Fundamentals of data visualization with bar charts.

-Experience in organizing code into reusable functions and maintaining a simple architecture.

14. Future Enhancements:

Here are some prospects of enhancement in this project in the future-

--Add line charts and pie charts for more detailed visual analysis.

--Also Introduce sorting options (by date, amount, or category).

--We can Implement search by amount or note keyword.

--We can Add monthly budget limits and warnings when nearing limits.

--Export reports to PDF or Excel

--We can also Convert the command-line application into a GUI using Tkinter or a web app using Flask.

15. References:

Python Official Documentation – csv, datetime, and os modules.

Matplotlib Documentation – plotting APIs and examples.

Course materials and lecture notes on Python programming and file handling.