Expense Tracker – Project Report

Expense Tracker

Submitted by: Kurra Sanjay
Roll Number: 25MIM10079

Course: Introduction to problem solving

1. Introduction

Expense management is essential for personal financial planning. This project presents a Python-based console application that helps users record, view, and analyze their daily spending by category. Built as a course project, this solution demonstrates key programming skills and real-world problem solving.

2. Problem Statement

Tracking spending manually can lead to missed expenses and poor financial control. Users often lack a centralized tool to add, view, and categorize spending in a streamlined way. This project addresses the need for a simple, user-friendly expense tracker.

3. Functional Requirements

- Users can add an expense with amount and category.
- View all recorded expenses in a list.
- Calculate and show total expenses.
- Generate a summary of expenses by category.
- Provide a menu-driven console interface.

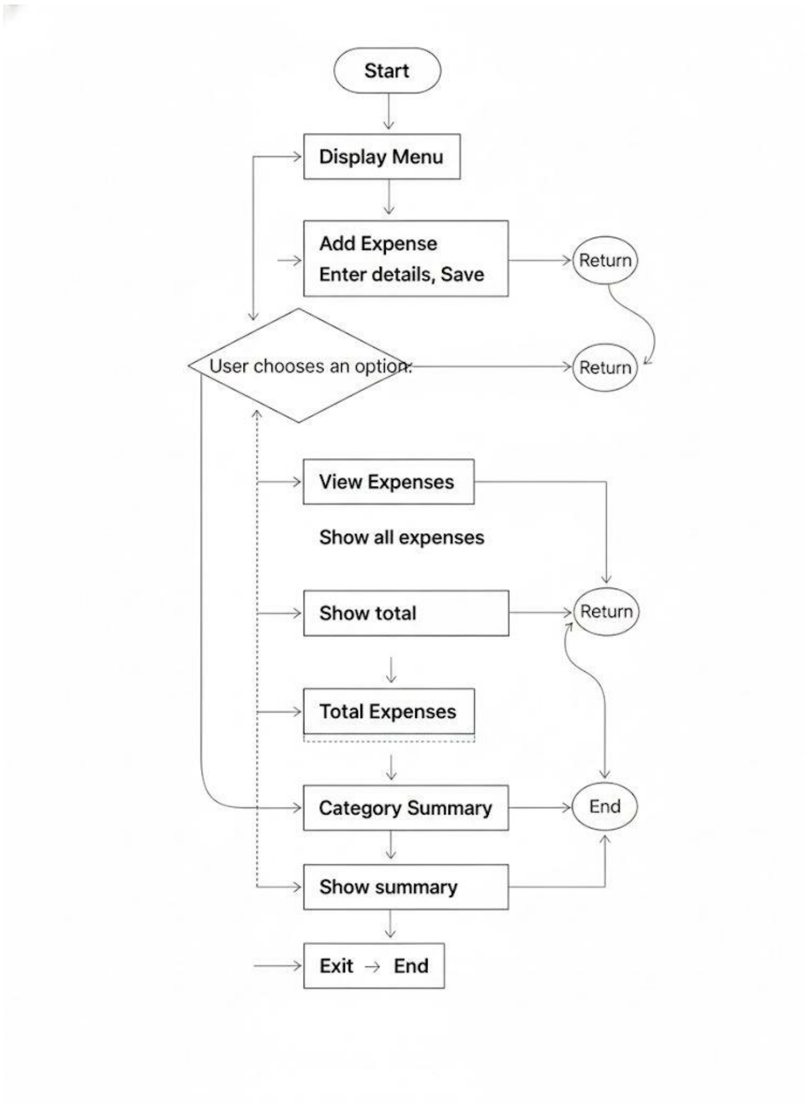
4. Non-functional Requirements

- User-friendly text-based interface.
- Input validation (e.g., numeric amount, handled invalid options).
- Reliable calculations and data accuracy.
- Modular code for easy enhancements.
- Runs on any system with Python 3.
- Well-documented code and clear project structure.

5. System Architecture

Architecture Overview:
User interacts with a menu → Python functions process input and update a list of expense entries → Output (expense list, totals, summaries) displayed in terminal.

6. Design Diagrams



7. Design Decisions & Rationale

- Used list of dictionaries for flexible in-memory data storage.
- Modular functions for each feature to separate logic and make updates easy.

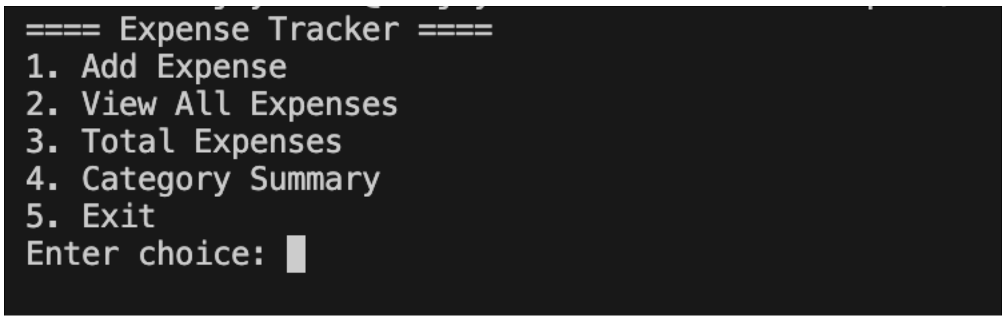
- Menu-based navigation for user-friendliness.
- Designed for beginners, no external dependencies required.

8. Implementation Details

- Implemented in Python 3.
- Uses only built-in functions (input, print, sum, etc.).
- Code divided into logical functions for each operation.
- Compatible with any OS running Python 3

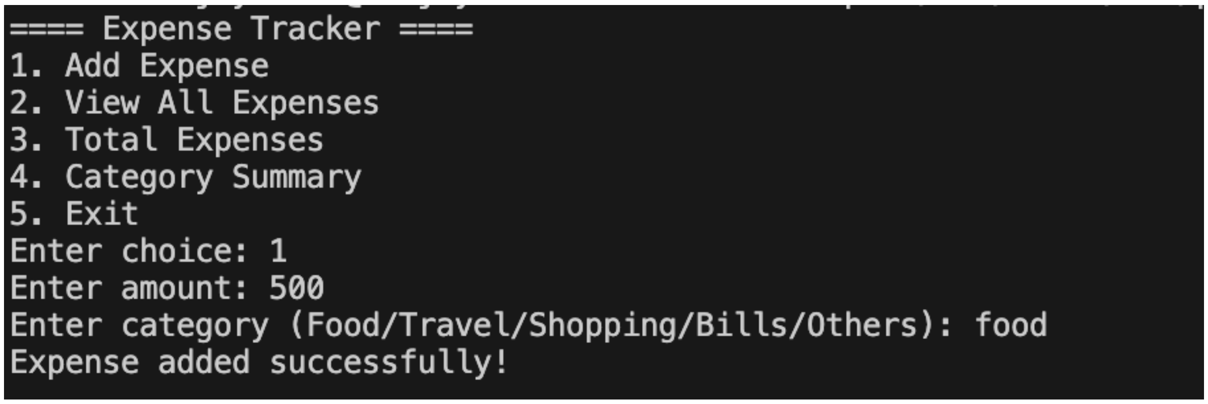
9. Screenshots / Results

- Menu.png



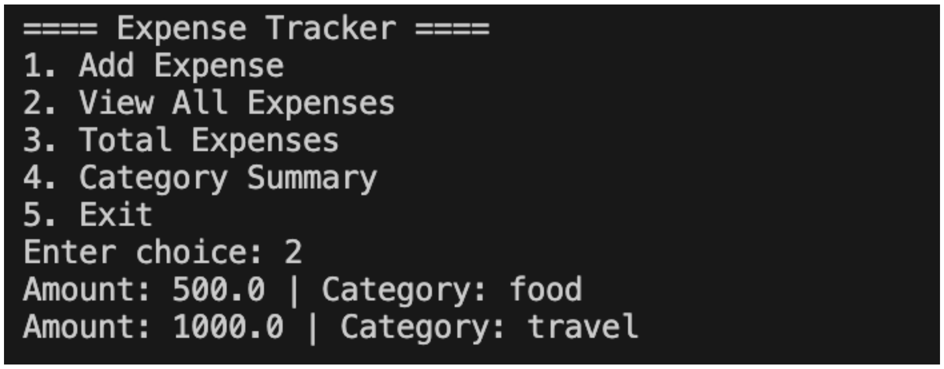
```
==== Expense Tracker ====
1. Add Expense
2. View All Expenses
3. Total Expenses
4. Category Summary
5. Exit
Enter choice: █
```

- add_expense.png



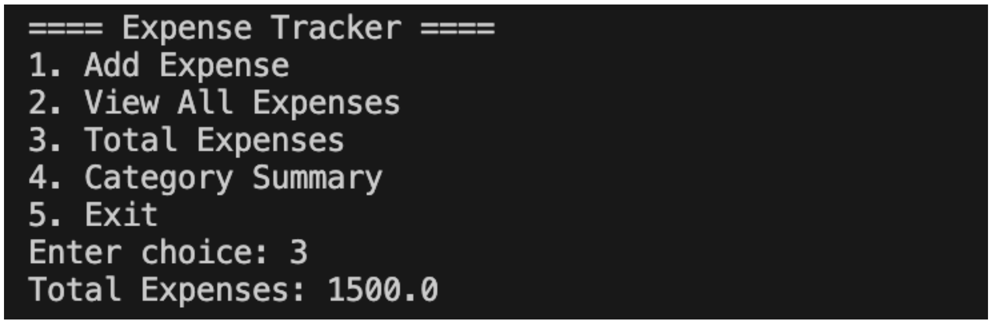
```
==== Expense Tracker ====
1. Add Expense
2. View All Expenses
3. Total Expenses
4. Category Summary
5. Exit
Enter choice: 1
Enter amount: 500
Enter category (Food/Travel/Shopping/Bills/Others): food
Expense added successfully!
```

- view_expenses.png



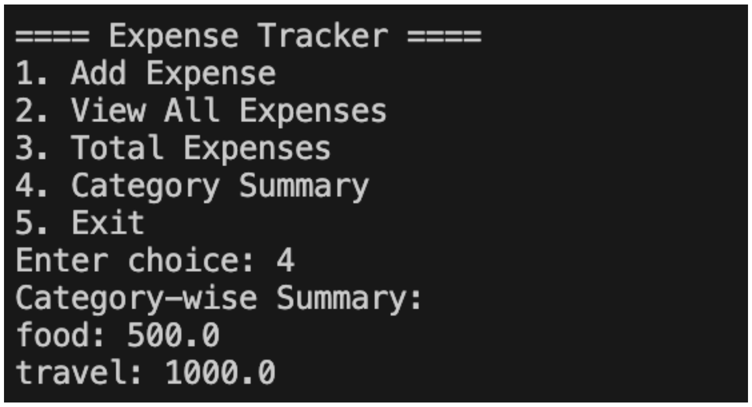
```
==== Expense Tracker ====
1. Add Expense
2. View All Expenses
3. Total Expenses
4. Category Summary
5. Exit
Enter choice: 2
Amount: 500.0 | Category: food
Amount: 1000.0 | Category: travel
```

- category_summary.png



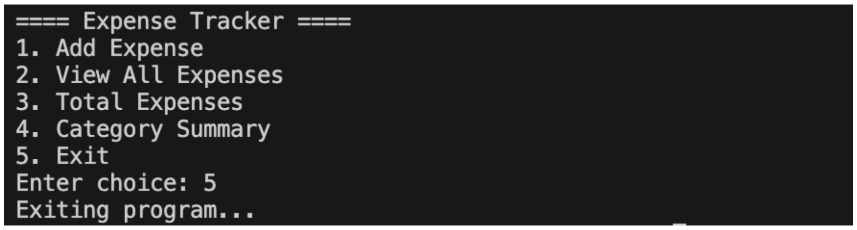
```
==== Expense Tracker ====
1. Add Expense
2. View All Expenses
3. Total Expenses
4. Category Summary
5. Exit
Enter choice: 3
Total Expenses: 1500.0
```

. total_expenses.png



```
==== Expense Tracker ====
1. Add Expense
2. View All Expenses
3. Total Expenses
4. Category Summary
5. Exit
Enter choice: 4
Category-wise Summary:
food: 500.0
travel: 1000.0
```

. Exit program



```
==== Expense Tracker ====
1. Add Expense
2. View All Expenses
3. Total Expenses
4. Category Summary
5. Exit
Enter choice: 5
Exiting program...
```

10. Testing Approach

- Manual testing by entering various valid and invalid expenses.
- Checked for handling of invalid inputs, empty records, and large totals.
- Verified output matches expected calculations.

11. Challenges Faced

- Handling invalid amounts and empty input.
- Ensuring program does not crash on bad input.
- Keeping user interface simple yet functional.

12. Learnings & Key Takeaways

- Effective use of Python lists and dictionaries.
- Modular programming and code readability.
- Importance of input validation and user feedback.
- Documenting and structuring code for collaborative work.

13. Future Enhancements

- Add data saving and loading using files.
- Implement a GUI with Tkinter.
- Add filtering/searching options for expenses.
- Expand to support different currencies or recurring expenses.

14. References

- Python Documentation (docs.python.org)
- Online Python tutorials (w3schools, programiz)
- Course material provided by faculty