

Expense Tracker – Version 1 Review

1. Project Overview

The Expense Tracker is a Python-based application designed to help users record, view, edit, and analyze their daily expenses. It provides features such as data filtering, category summaries, monthly breakdowns, and exporting reports. Version 1 was implemented as a command-line interface (CLI) application, using CSV files to store data, pandas for data management, and tabulate for clean table display.

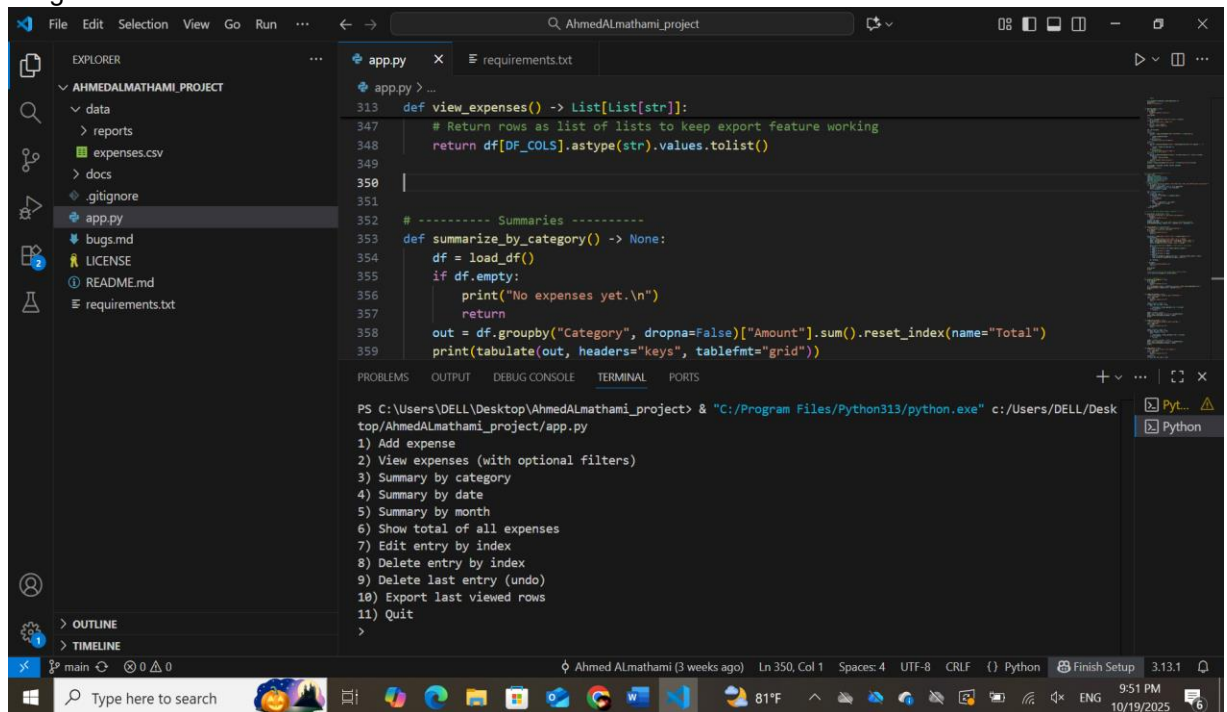
2. Features Implemented

- Add new expense entries (Date, Amount, Category, Description).
- View all expenses with optional filters (category, date range, keyword search).
- Edit or delete any expense entry by index.
- Summaries by category, date, and month.
- Export filtered reports to timestamped CSV files.
- Graceful exit handling (Ctrl+C).
- Data stored and loaded using pandas DataFrame for improved performance and flexibility.

3. Demonstration

The following screenshots demonstrate the working state of the program:

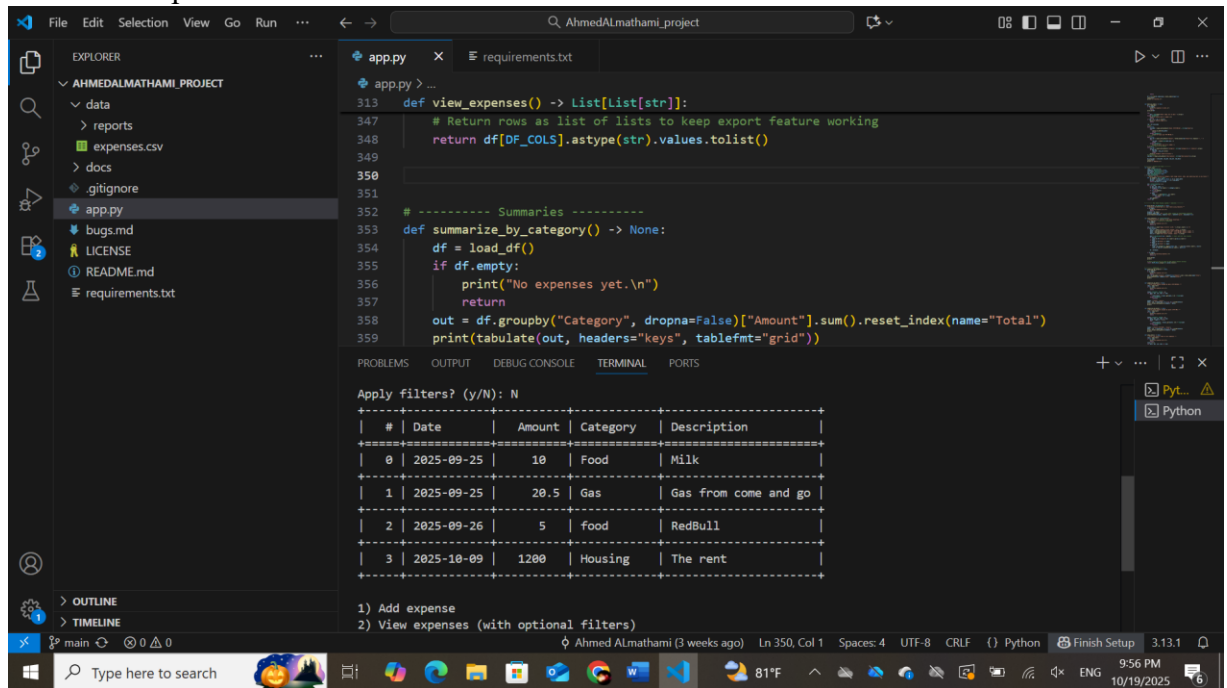
- Program menu



The screenshot shows a Visual Studio Code editor with a file explorer on the left displaying the project structure: AHMEDALMATHAMI_PROJECT, data, reports, expenses.csv, docs, .gitignore, app.py, bugs.md, LICENSE, README.md, and requirements.txt. The main editor window shows the app.py file with Python code for viewing expenses and summarizing by category. The terminal window at the bottom displays the program menu:

```
PS C:\Users\DELL\Desktop\AhmedAlmathami_project> & "C:/Program Files/Python313/python.exe" c:/Users/DELL/Desktop/AhmedAlmathami_project/app.py
1) Add expense
2) View expenses (with optional filters)
3) Summary by category
4) Summary by date
5) Summary by month
6) Show total of all expenses
7) Edit entry by index
8) Delete entry by index
9) Delete last entry (undo)
10) Export last viewed rows
11) Quit
>
```

➤ Tabulated expense view:



The screenshot shows the Visual Studio Code interface with a project named 'AHMEDALMATHAMI_PROJECT'. The Explorer panel on the left shows the project structure, including a 'data' folder with 'expenses.csv'. The main editor displays the 'app.py' file with Python code. The code includes a function 'view_expenses()' that returns a list of lists, and a function 'summarize_by_category()' that uses 'df.groupby()' and 'df.sum()' to calculate totals. The terminal output shows the command 'Apply filters? (y/N): N' followed by a tabulated view of expenses.

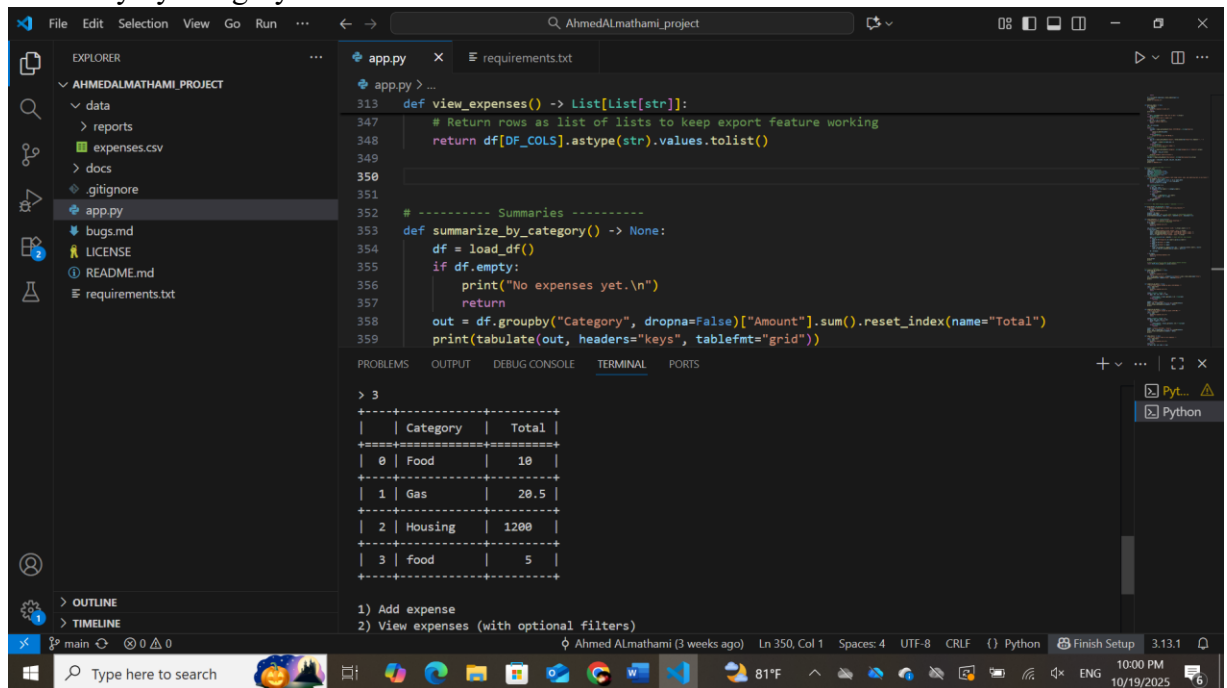
```
def view_expenses() -> List[List[str]]:
    # Return rows as list of lists to keep export feature working
    return df[DF_COLS].astype(str).values.tolist()

# ----- Summaries -----
def summarize_by_category() -> None:
    df = load_df()
    if df.empty:
        print("No expenses yet.\n")
        return
    out = df.groupby("Category", dropna=False)["Amount"].sum().reset_index(name="Total")
    print(tabulate(out, headers="keys", tablefmt="grid"))
```

#	Date	Amount	Category	Description
0	2025-09-25	10	Food	Milk
1	2025-09-25	20.5	Gas	Gas from come and go
2	2025-09-26	5	food	RedBull
3	2025-10-09	1200	Housing	The rent

1) Add expense
2) View expenses (with optional filters)

➤ Summary by category:

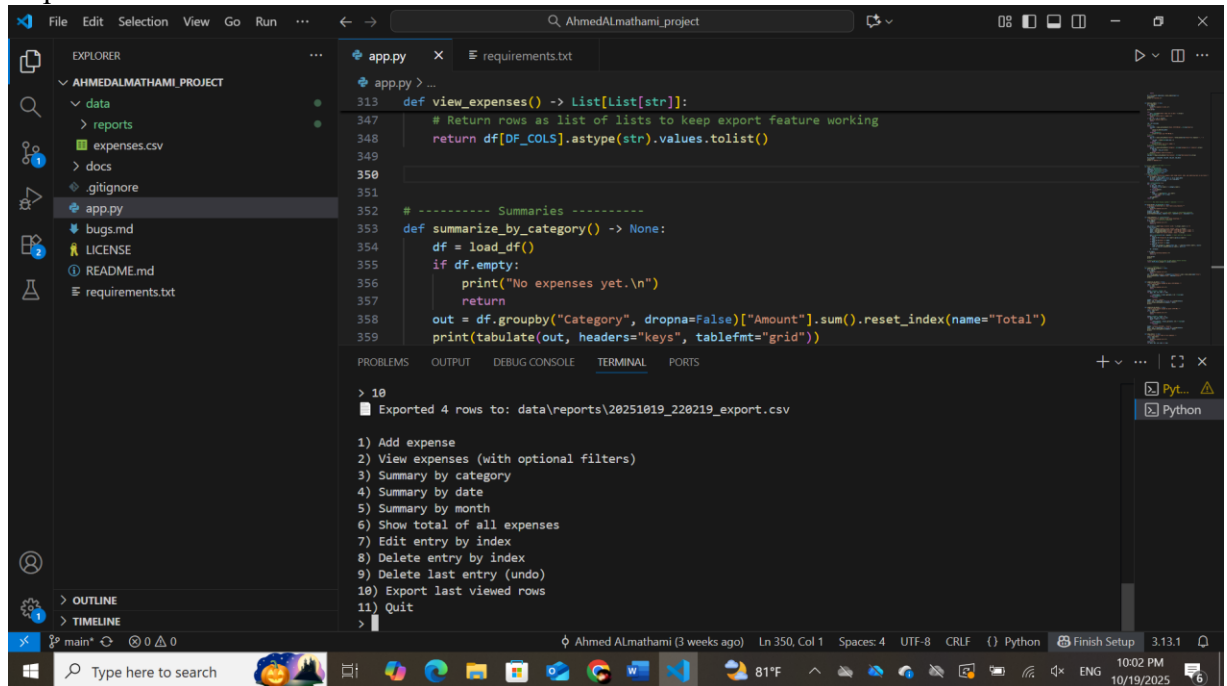


The screenshot shows the Visual Studio Code interface with the same project. The terminal output now shows the command '> 3' followed by a summary by category view, which is a tabulated view of the total amount for each category.

```
> 3
|  | Category | Total |
|  |-----|-----|
| 0 | Food    | 10    |
| 1 | Gas     | 20.5  |
| 2 | Housing | 1200  |
| 3 | food    | 5     |
```

1) Add expense
2) View expenses (with optional filters)

➤ Export confirmation:



The screenshot shows a VS Code editor window for a project named 'AhmedALmathami_project'. The Explorer sidebar on the left shows the project structure: 'data' (containing 'reports' and 'expenses.csv'), 'docs', 'gitignore', 'app.py', 'bugs.md', 'LICENSE', 'README.md', and 'requirements.txt'. The main editor area shows the 'app.py' file with the following code:

```
313 def view_expenses() -> List[List[str]]:
347     # Return rows as list of lists to keep export feature working
348     return df[DF_COLS].astype(str).values.tolist()
349
350
351
352 # ----- Summaries -----
353 def summarize_by_category() -> None:
354     df = load_df()
355     if df.empty:
356         print("No expenses yet.\n")
357     return
358     out = df.groupby("Category", dropna=False)["Amount"].sum().reset_index(name="Total")
359     print(tabulate(out, headers="keys", tablefmt="grid"))
```

The terminal at the bottom shows the following output:

```
> 10
Exported 4 rows to: data/reports/20251019_220219_export.csv

1) Add expense
2) View expenses (with optional filters)
3) Summary by category
4) Summary by date
5) Summary by month
6) Show total of all expenses
7) Edit entry by index
8) Delete entry by index
9) Delete last entry (undo)
10) Export last viewed rows
11) Quit
>
```

4. Issues I faced and Solutions:

- Data structure: Initially used nested lists, which made filtering difficult. Solved by switching to pandas DataFrame.
- Misaligned text output: Resolved by using tabulate module for clean table formatting.
- Category input: Previously allowed blanks, now validated to avoid empty values.
- Graceful exit: Added sys.exit() handling to prevent abrupt program termination.

5. Help Needed:

I just would like your advice about your recommendations for structuring Streamlit components for Version 2 GUI.

6. Milestones for Version 2:

- Milestone 1 : Streamlit Setup: Create Streamlit interface with sidebar input fields for adding expenses.
- Milestone 2 : Data Integration: Connect Streamlit interface with pandas-based backend for live updates.
- Milestone 3 : Visualization: Add summary charts (totals by category, monthly trends).

- Milestone 4 :Export/Download: Implement export and download buttons directly in Streamlit.
- Milestone 5 : Polish and Testing: Improve layout, add icons, and perform usability testing.

7. Self Reflection

I am satisfied with my progress so far. I was able to complete all core features and make my code more organized and maintainable. Using pandas and tabulate was a big improvement from my earlier approach. Some tasks took longer than expected, especially understanding pandas operations, but I learned a lot from the process. I believe I can finish the Streamlit version within the next few weeks. The biggest 'light bulb' moment was realizing how pandas can simplify filtering and summarizing operations that were previously done manually.