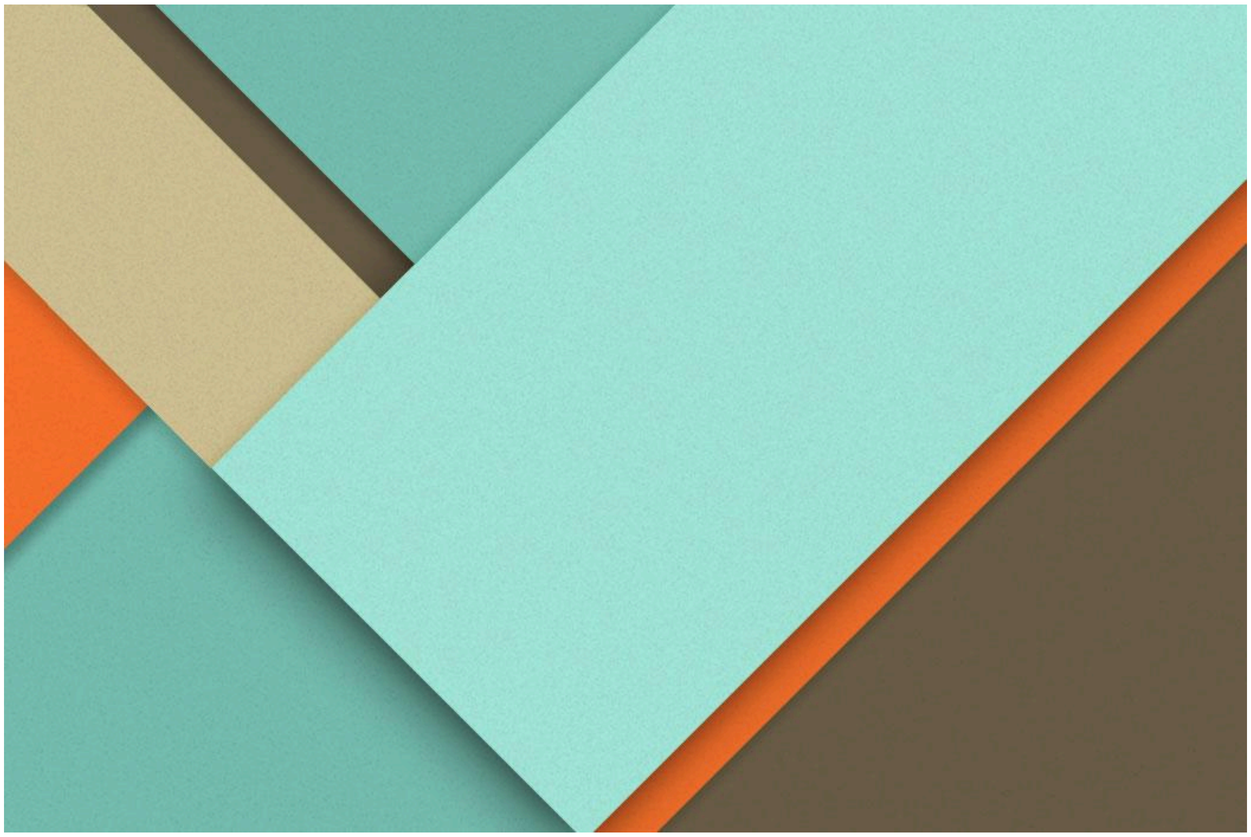


Financial Assistant Project

11.22.2024

Murtaza Gohari
Lenoir Rhyne University
Programming for Everyone I
CSC II (Python Programming)

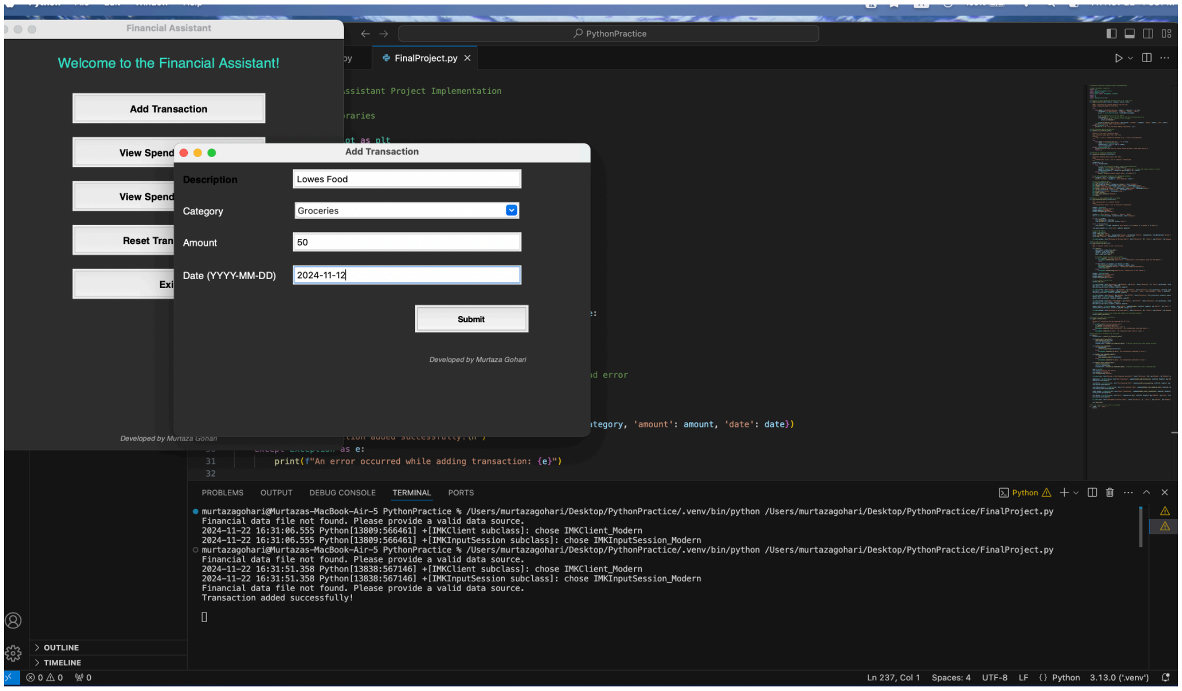
Project Overview

The Financial Assistant is a desktop application designed to help users manage and visualize their financial transactions. This project addresses a common problem: many people struggle to keep track of their spending, which often leads to overspending and poor budgeting. By providing a user-friendly interface to add transactions, view spending summaries, and reset data, the Financial Assistant helps users stay organized and gain better insights into their finances.

The application allows users to add financial transactions by providing a description, category, amount, and date. Users can view their expenses in a tabular form or visualize them with a spending chart. The main purpose is to provide an easy-to-use tool for financial management, particularly focusing on spending awareness and control.

Example User Interaction:

- **Adding a Transaction**: The user clicks "Add Transaction," and enters details like description (e.g., "Groceries"), category (e.g., "Food"), amount (e.g., "50"), and date (e.g., "2024-11-22"). Upon submission, a confirmation message appears, indicating that the transaction was added successfully.

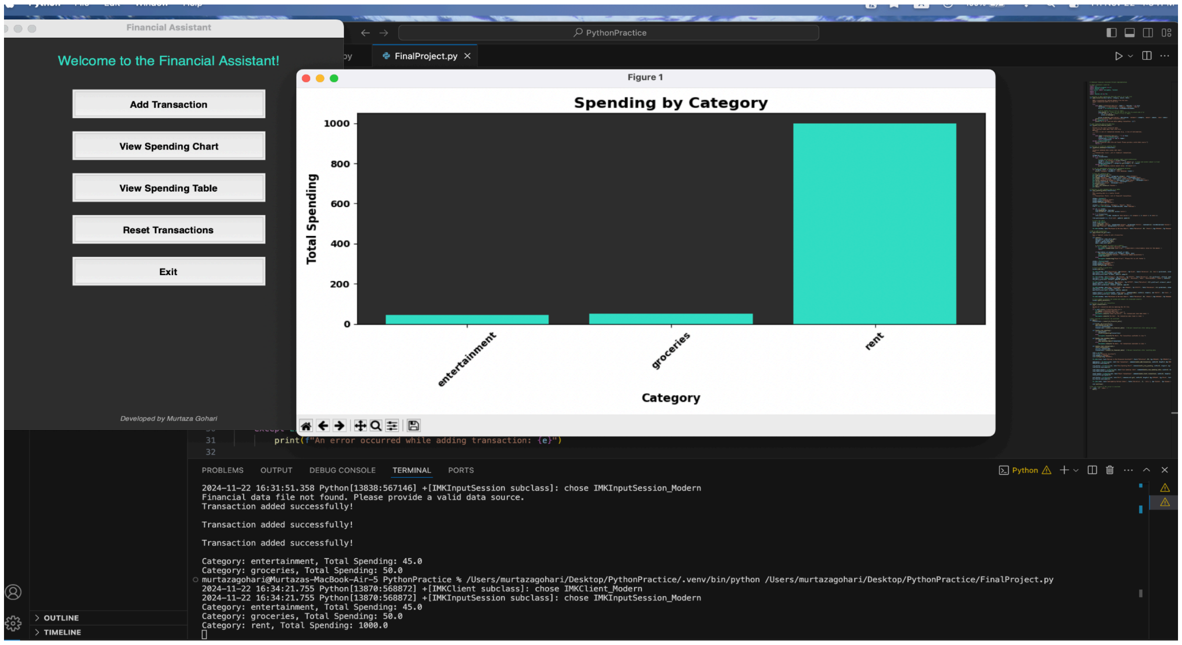


Description	Lowes Food
Category	Groceries
Amount	50
Date (YYYY-MM-DD)	2024-11-12

```
print(f"An error occurred while adding transaction: {e}")
```

```
murtazagohari@Murtazas-MacBook-Air-5 PythonPractice % /Users/murtazagohari/Desktop/PythonPractice/.venv/bin/python /Users/murtazagohari/Desktop/PythonPractice/FinalProject.py
Financial data file not found. Please provide a valid data source.
2024-11-22 16:31:06.555 Python[13809:566461] +[IMKClient subclass]: chose IMKClient_Modern
2024-11-22 16:31:06.555 Python[13809:566461] +[IMKInputSession subclass]: chose IMKInputSession_Modern
murtazagohari@Murtazas-MacBook-Air-5 PythonPractice % /Users/murtazagohari/Desktop/PythonPractice/.venv/bin/python /Users/murtazagohari/Desktop/PythonPractice/FinalProject.py
Financial data file not found. Please provide a valid data source.
2024-11-22 16:31:51.358 Python[13838:567146] +[IMKClient subclass]: chose IMKClient_Modern
2024-11-22 16:31:51.358 Python[13838:567146] +[IMKInputSession subclass]: chose IMKInputSession_Modern
Financial data file not found. Please provide a valid data source.
Transaction added successfully!
```

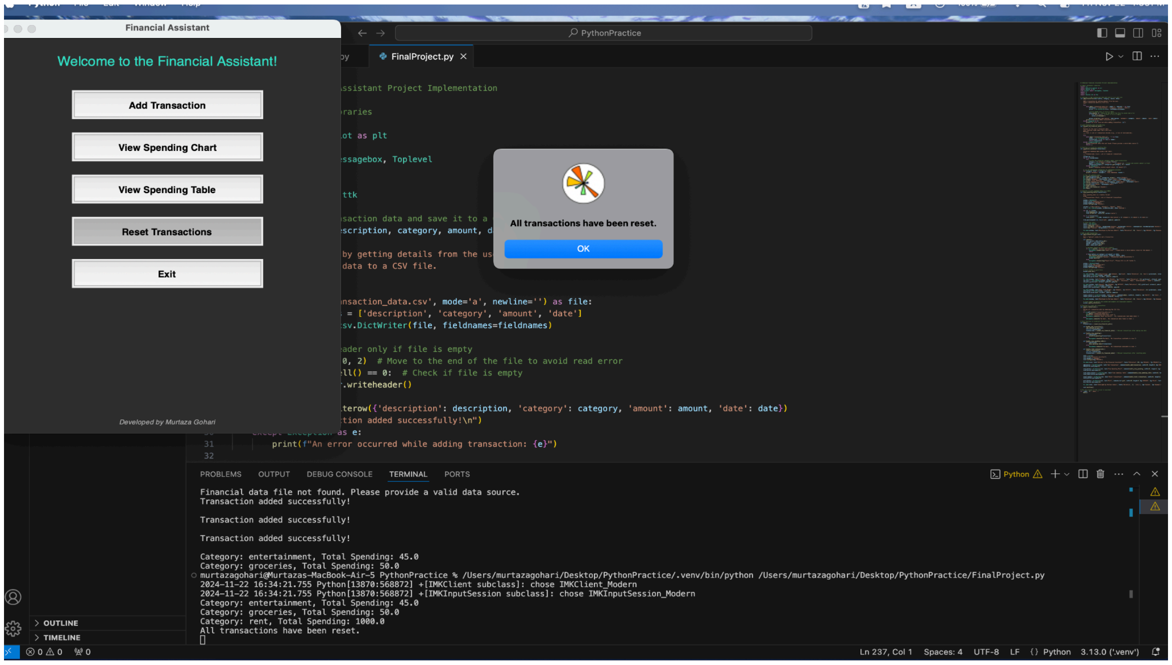
- **Viewing Spending Chart**: The user clicks "View Spending Chart" and sees a bar chart showing how much they have spent on different categories.



```
print(f"An error occurred while adding transaction: {e}")
```

```
2024-11-22 16:31:51.358 Python[13838:567146] +[IMKInputSession subclass]: chose IMKInputSession_Modern
Financial data file not found. Please provide a valid data source.
Transaction added successfully!

Transaction added successfully!

Transaction added successfully!

Category: entertainment, Total Spending: 45.0
Category: groceries, Total Spending: 50.0
murtazagohari@Murtazas-MacBook-Air-5 PythonPractice % /Users/murtazagohari/Desktop/PythonPractice/.venv/bin/python /Users/murtazagohari/Desktop/PythonPractice/FinalProject.py
2024-11-22 16:34:21.755 Python[13870:568872] +[IMKClient subclass]: chose IMKClient_Modern
2024-11-22 16:34:21.755 Python[13870:568872] +[IMKInputSession subclass]: chose IMKInputSession_Modern
Category: entertainment, Total Spending: 45.0
Category: groceries, Total Spending: 50.0
Category: rent, Total Spending: 1000.0
```

- **Resetting Transactions**: The user clicks "Reset Transactions" and receives confirmation that all transaction data has been cleared.

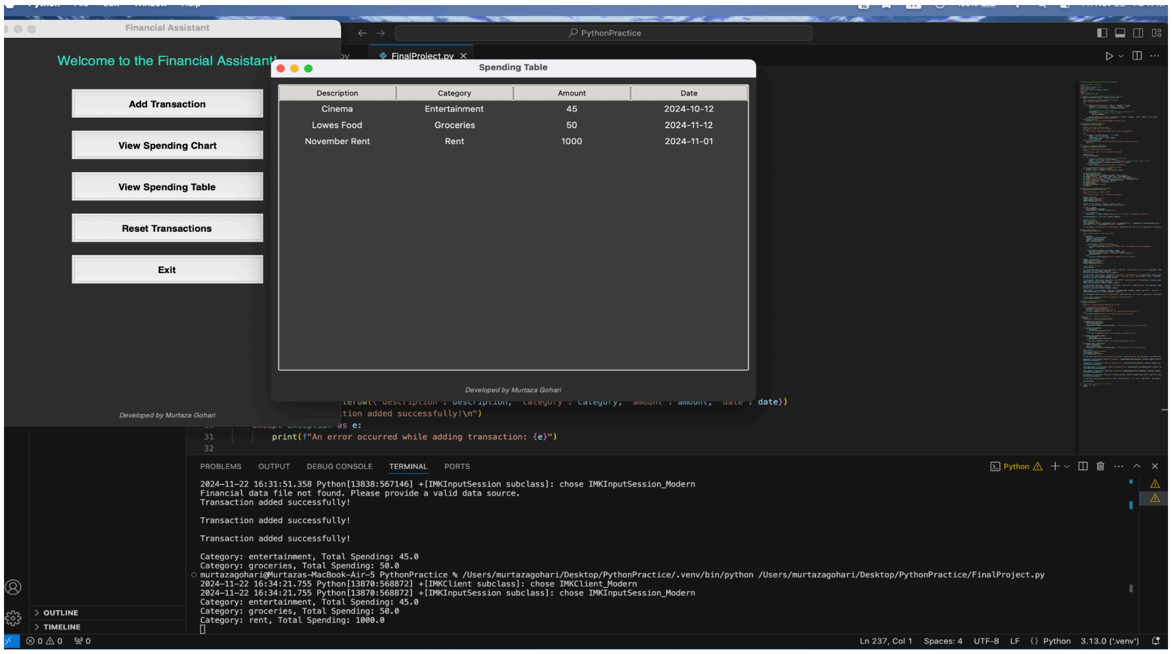


```
ssistant Project Implementation

raries

ot as plt

ssagebox, Toplevel


ttk

saction data and save it to a
scription, category, amount, d

by getting details from the us
data to a CSV file.


nsaction_data.csv', mode='a', newline='') as file:
 = ['description', 'category', 'amount', 'date']
sv.DictWriter(file, fieldnames=fieldnames)

ader only if file is empty
0, 2)  # Move to the end of the file to avoid read error
ll() == 0:  # Check if file is empty
.writeheader()

terow({'description': description, 'category': category, 'amount': amount, 'date': date})
tion added successfully!\n")
as e:
    print(f"An error occurred while adding transaction: {e}")
```

```
Financial data file not found. Please provide a valid data source.
Transaction added successfully!

Transaction added successfully!

Transaction added successfully!

Category: entertainment, Total Spending: 45.0
Category: groceries, Total Spending: 50.0
murtazagohari@Murtazas-MacBook-Air-5 PythonPractice % /Users/murtazagohari/Desktop/PythonPractice/.venv/bin/python /Users/murtazagohari/Desktop/PythonPractice/FinalProject.py
2024-11-22 16:34:21.755 Python[13870:568872] +[IMKClient subclass]: chose IMKClient_Modern
2024-11-22 16:34:21.755 Python[13870:568872] +[IMKInputSession subclass]: chose IMKInputSession_Modern
Category: entertainment, Total Spending: 45.0
Category: groceries, Total Spending: 50.0
Category: rent, Total Spending: 1000.0
All transactions have been reset.
```

Concept Application

- **Variables**: Variables are used to store user inputs like transaction details (description, category, amount, and date). These variables help to organize the data flow between different functions.
- **Data Types**: Different data types are used, such as strings for categories, dates, and descriptions, and floating-point numbers for amounts. This ensures that financial calculations are done accurately, and data is appropriately represented.
- **Functions**: Functions are employed extensively to modularize the code and avoid redundancy. For instance, add_transaction(), visualize_spending(), and reset_transactions() are defined to handle specific actions, making the codebase organized and easy to manage.
- **Modularization**: The project is divided into multiple functions, each responsible for a specific task. Modularization allows for better readability and maintainability, making it easier to test individual components separately.
- **Loops**: A loop is used to iterate through transaction records for visualization purposes. For example, when showing transactions in a table or calculating total spending per category.
- **Conditional Statements**: Conditional statements are used to validate user input, ensuring that all necessary information is provided before processing a transaction. For instance, ensuring the amount entered is numeric and the description is not left empty.
- **Libraries**: The application uses several Python libraries to enhance functionality:
    - csv: To read and write transaction data.
    - matplotlib: To visualize spending through bar charts.
    - Tkinter: To create the graphical user interface (GUI).
    - re: For input validation, such as ensuring the amount entered is a valid number.



Description	Category	Amount	Date
Cinema	Entertainment	45	2024-10-12
Lowes Food	Groceries	50	2024-11-12
November Rent	Rent	1000	2024-11-01

```
        print(f"An error occurred while adding transaction: {e}")
```

```
2024-11-22 16:31:51.358 Python[13838:567146] +[IMKInputSession subclass]: chose IMKInputSession_Modern
Financial data file not found. Please provide a valid data source.
Transaction added successfully!

Transaction added successfully!

Transaction added successfully!

Category: entertainment, Total Spending: 45.0
Category: groceries, Total Spending: 50.0
murtazagohari@Murtazas-MacBook-Air-5 PythonPractice % /Users/murtazagohari/Desktop/PythonPractice/.venv/bin/python /Users/murtazagohari/Desktop/PythonPractice/FinalProject.py
2024-11-22 16:34:21.755 Python[13870:568872] +[IMKClient subclass]: chose IMKClient_Modern
2024-11-22 16:34:21.755 Python[13870:568872] +[IMKInputSession subclass]: chose IMKInputSession_Modern
Category: entertainment, Total Spending: 45.0
Category: groceries, Total Spending: 50.0
Category: rent, Total Spending: 1000.0
```

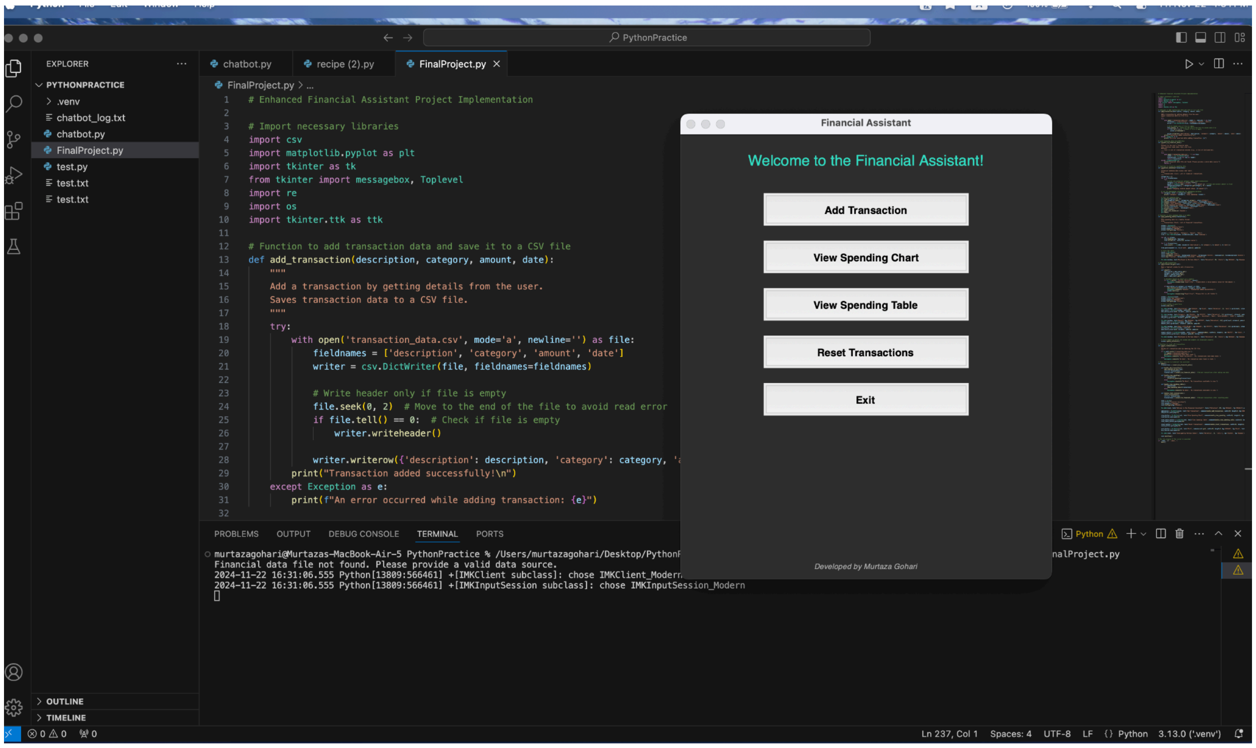
Feedback Addressed

- **Feedback**: The UI was initially too basic, and users reported difficulty reading certain labels due to color choices.
    - **Response**: The UI was enhanced by adding a dark theme and choosing more readable colors for buttons and labels. Contrasting colors were used to ensure readability.
- **Feedback**: Some users found it confusing to manually enter dates in the correct format.
    - **Response**: A prompt was added to guide users to enter dates in YYYY-MM-DD format. The potential use of a date-picker widget was considered but not implemented due to library compatibility issues.

Tools Used

- **Web Resources**: I took the initiative to research various online resources to enhance the project. Stack Overflow was used extensively for resolving Tkinter-related GUI issues and for understanding advanced matplotlib plotting techniques.

- **Code Examples**: I adapted the structure of matplotlib bar charts from tutorials found on the official Matplotlib documentation. These resources helped me understand the best practices for visualizing financial data in an engaging way.
- **Generative AI Assistance**:
  - **Prompts Used**: I used ChatGPT as a collaborative tool to brainstorm ideas for implementing Tkinter GUI elements and customizing matplotlib visualizations. Specific prompts included "How to add a chart in Tkinter using matplotlib" and "Best practices for styling a Tkinter GUI." These prompts were designed to improve both the functionality and aesthetics of the project.
  - **Success**: The AI was particularly helpful in providing suggestions to optimize the user experience, such as enhancing the visual elements of the user interface, and refining input validation mechanisms.
  - **Limitations**: While AI provided valuable insights, there were some limitations in terms of specific library compatibility solutions. For instance, integrating a date-picker widget with the current Tkinter version required more in-depth research beyond what the AI could offer.



```python
# Enhanced Financial Assistant Project Implementation

# Import necessary libraries
import csv
import matplotlib.pyplot as plt
import tkinter as tk
from tkinter import messagebox, Toplevel
import re
import os
import tkinter.ttk as ttk

# Function to add transaction data and save it to a CSV file
def add_transaction(description, category, amount, date):
    """
    Add a transaction by getting details from the user.
    Saves transaction data to a CSV file.
    """
    try:
        with open('transaction_data.csv', mode='a', newline='') as file:
            fieldnames = ['description', 'category', 'amount', 'date']
            writer = csv.DictWriter(file, fieldnames=fieldnames)

            # Write header only if file is empty
            file.seek(0, 2)  # Move to the end of the file to avoid read error
            if file.tell() == 0:  # Check if file is empty
                writer.writeheader()

            writer.writerow({'description': description, 'category': category, 'a
        print("Transaction added successfully!\n")
    except Exception as e:
        print(f"An error occurred while adding transaction: {e}")

```

```
murtazagohari@Murtazas-MacBook-Air-5 PythonPractice % /Users/murtazagohari/Desktop/PythonP
Financial data file not found. Please provide a valid data source.
2024-11-22 16:31:06.555 Python[13809:566461] +[IMKClient subclass]: chose IMKClient_Modern
2024-11-22 16:31:06.555 Python[13809:566461] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```

Conclusion

The Financial Assistant application successfully addresses the need for personal finance management by allowing users to easily track and visualize their spending. The user-friendly interface, combined with clear data visualization tools, makes it an effective solution for managing personal transactions. The feedback-driven improvements, such as enhanced UI and date prompts, further contribute to the overall usability and accessibility of the application.

Future Development:

- **Add More Visualization Types**: Introducing more charts, such as pie charts or line graphs, could give users more options to understand their spending patterns visually.
- **Improve Date Entry**: Integrating a date-picker widget would further simplify the user experience, reducing potential errors in date formatting.
- **Add a Summary Dashboard**: Including a summary dashboard that gives an overview of monthly or yearly spending can provide users with more comprehensive financial insights.

These enhancements would contribute to making the Financial Assistant even more robust and user-friendly in addressing personal finance management needs.

To run the Financial Assistant application in VS Code, you will need the following files:

- **FinalProject.py**: This is the main script that contains the entire program.
- **transaction_data.csv**: This file is used to store and retrieve transaction records. The program will create this file automatically if it doesn't exist.