2024

Personal Budget Tracking Application

PROJECT REPORT BY

ANUSHA MULUKUTLA

Contents

PERSONAL BUDGET TRACKING APPLICATION

PROJECT_DESCRIPTION

This project involves developing a Personal Budget Tracker application using Python and the Tkinter library for the GUI. The application allows users to track their income, expenses, and budgets on a monthly basis. Users can input their total salary and additional income, categorize their expenses, and set budgets for different categories. The data is stored in CSV files for each month, enabling users to maintain detailed financial records. The application includes features for generating visualizations such as bar graphs to compare actual expenditures against budgets and pie charts to analyze spending patterns. It also provides validation for user inputs and ensures a smooth user experience with clear feedback messages and an easy-to-navigate interface.

GITHUB LINK (PERSONAL_BUDGET_TRACKING_APPLICATION)

https://github.com/anushamulukutla/Personal_Budget_traking_App/blob/main/Main_PBT

PROGRAMMING CONCEPTS IMPLEMENTED

1. Graphical User Interface (GUI) with Tkinter

- Tkinter: Utilized the Tkinter library to create the GUI for the application. Employed widgets within ttk module such as Frame, Label, Entry, Combobox, and Button to create a structured and user-friendly interface. Specifically, I used the Frame concept inside the ttk module to create four frames, each performing different functionalities. The grid geometry manager was used for an organized layout.

2.Interactive User Interface

- Defined functions to manage button click events, ensuring the application responds dynamically to user actions such as submitting income, recording expenses, and setting budgets. Used the command attribute in ttk.Button to link buttons to their respective event handler functions, enabling an interactive and responsive user experience.

3. Input Validation and Error Handling

- **Validation**: Ensured user inputs are valid using try-except blocks and conditional checks.
- **Error Messages**: Displayed informative messages using messagebox for incorrect or incomplete inputs.

4.FILEHANDLING

- **CSV Module**: Used Python’s csv module (import csv) to read from and write to CSV files for data persistence.

5. Data Manipulation with Pandas

- **DataFrames**: Leveraged Pandas DataFrames for data manipulation and analysis.
- **Filtering and Merging**: Filtered and merged data from different CSV files for comprehensive reports.

6. Data Visualization with Matplotlib

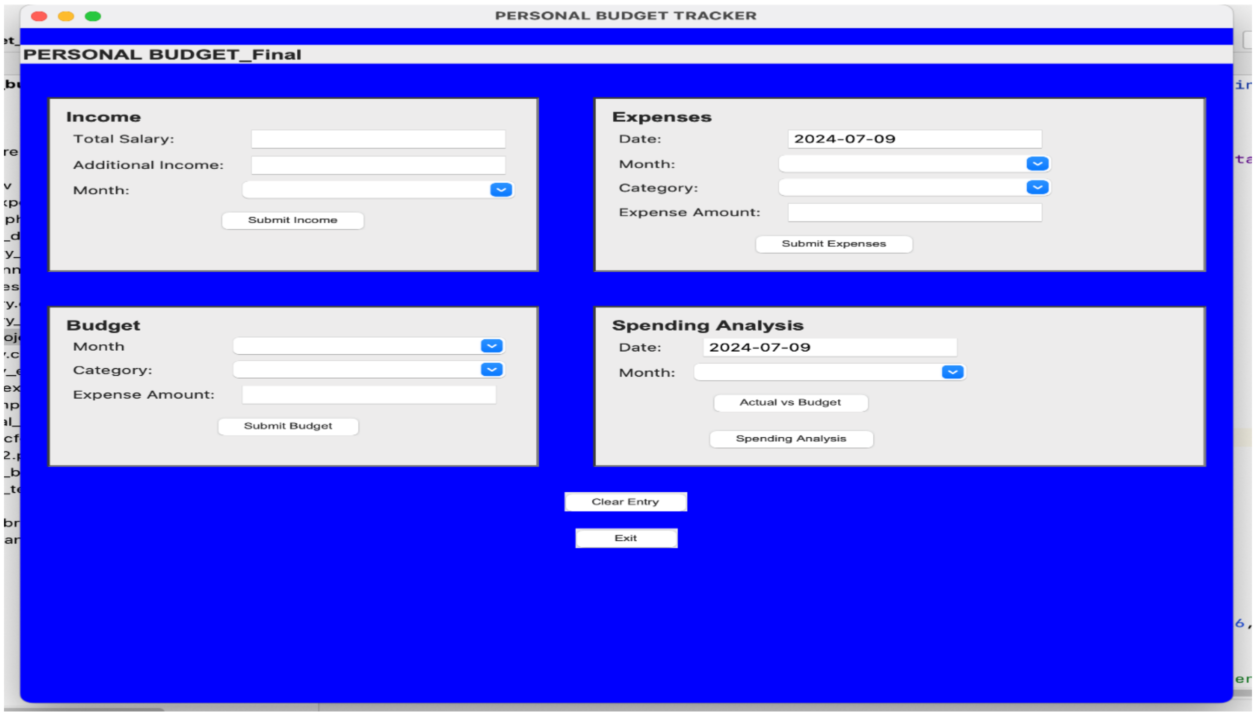
- **Bar Charts and Pie Charts**: Created visualizations to represent budget vs. actual spending and spending Analysis.

NEW TECHNIQUES YOU LEARNT DURING THIS PROJECT

- **Pandas**: Learned how to effectively use the Pandas library for data manipulation and analysis. This included reading and writing CSV files, filtering, and merging DataFrames, and performing various data operations to prepare data for visualization.
- **Event Handling and Validation**: Improved my ability to manage user inputs and events, including input validation and error handling to ensure a robust and user-friendly application.

SCREENSHOTS

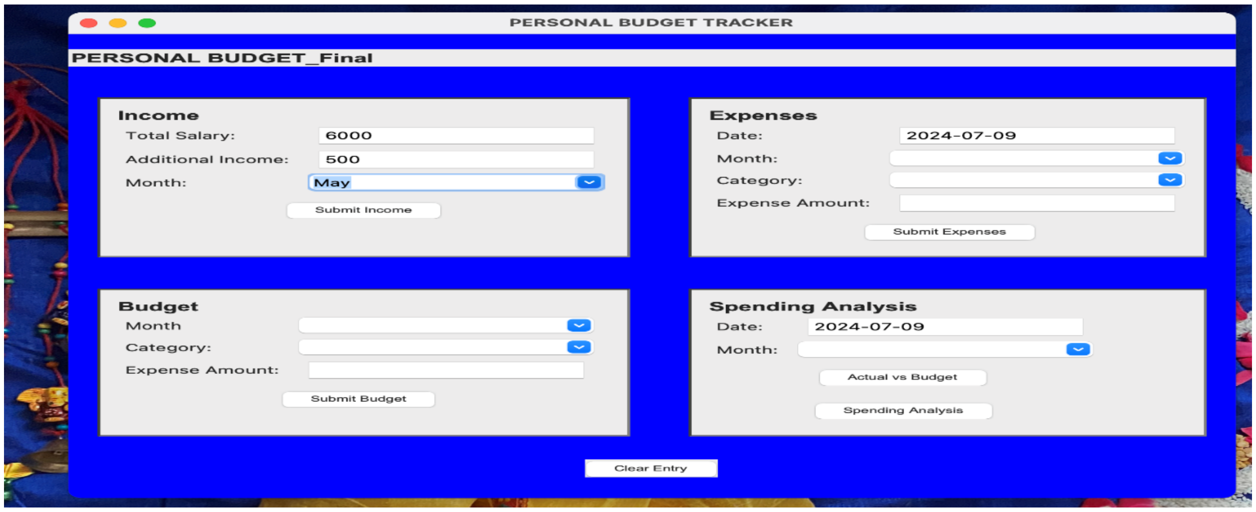
Main window



Date: 2024-07-09

Date: 2024-07-09

Enter Income details and click on 'Submit Income'



PERSONAL BUDGET TRACKER

PERSONAL BUDGET_Final

Income

Total Salary:	6000
Additional Income:	500
Month:	May

Submit Income

Expenses

Date:	2024-07-09
Month:	
Category:	
Expense Amount:	

Submit Expenses

Budget

Month	
Category:	
Expense Amount:	

Submit Budget

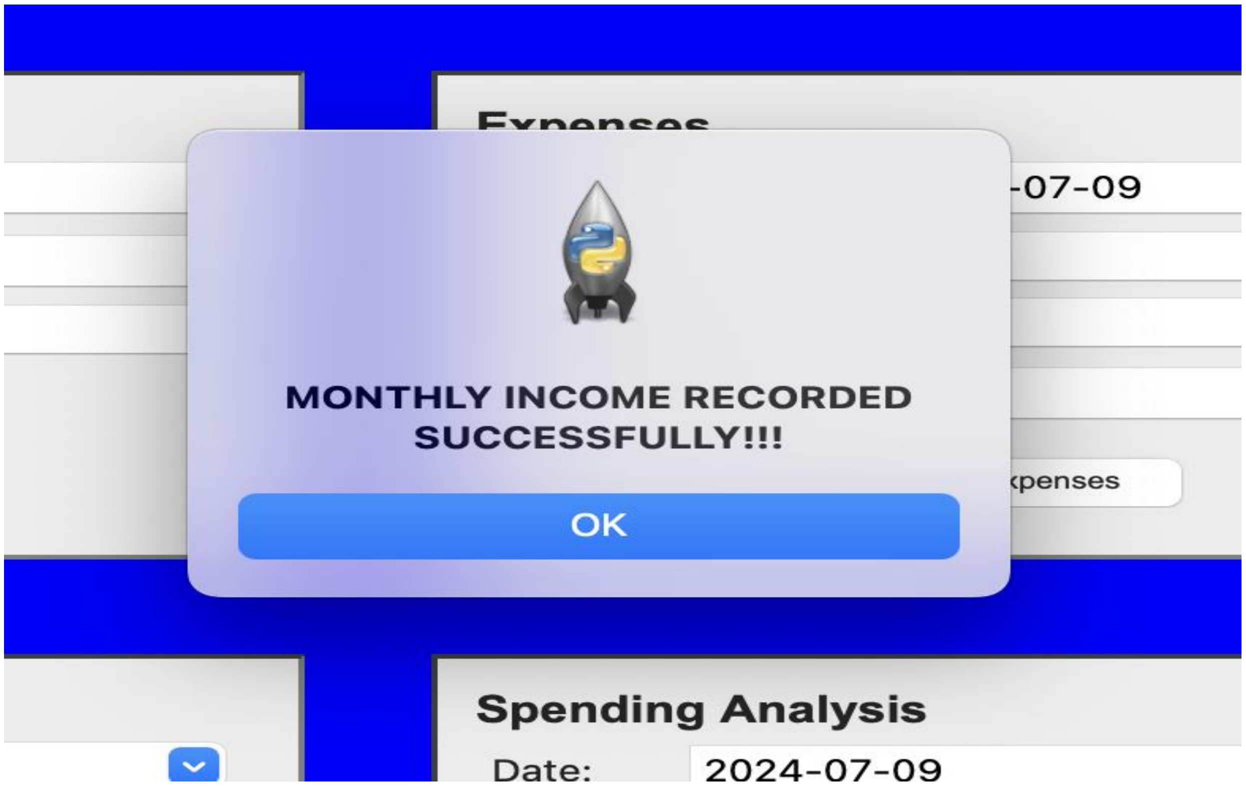
Spending Analysis

Date:	2024-07-09
Month:	

Actual vs Budget

Spending Analysis

Clear Entry

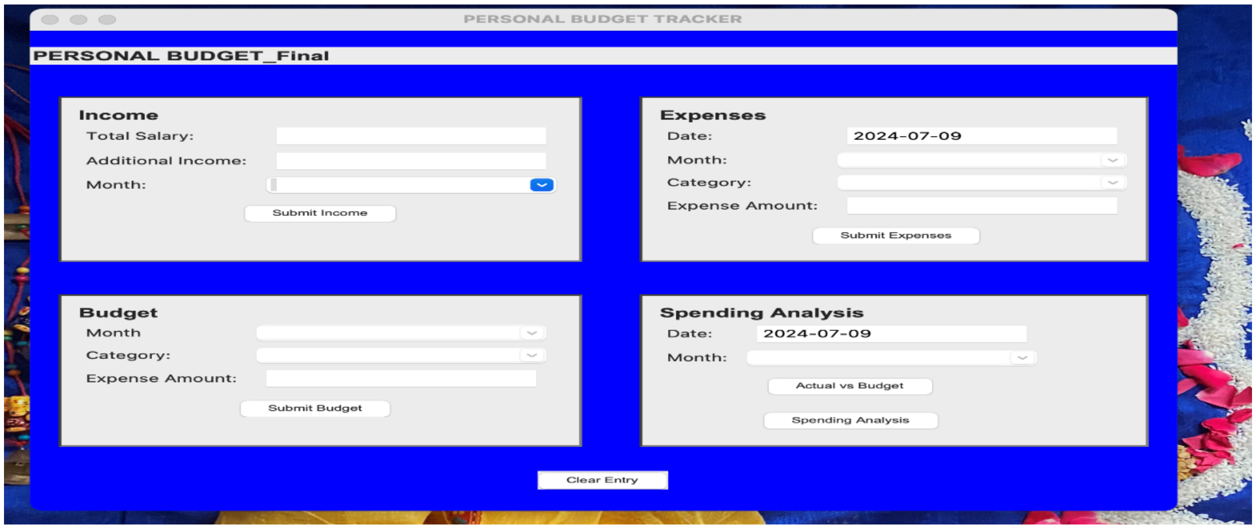


MONTHLY INCOME RECORDED SUCCESSFULLY!!!

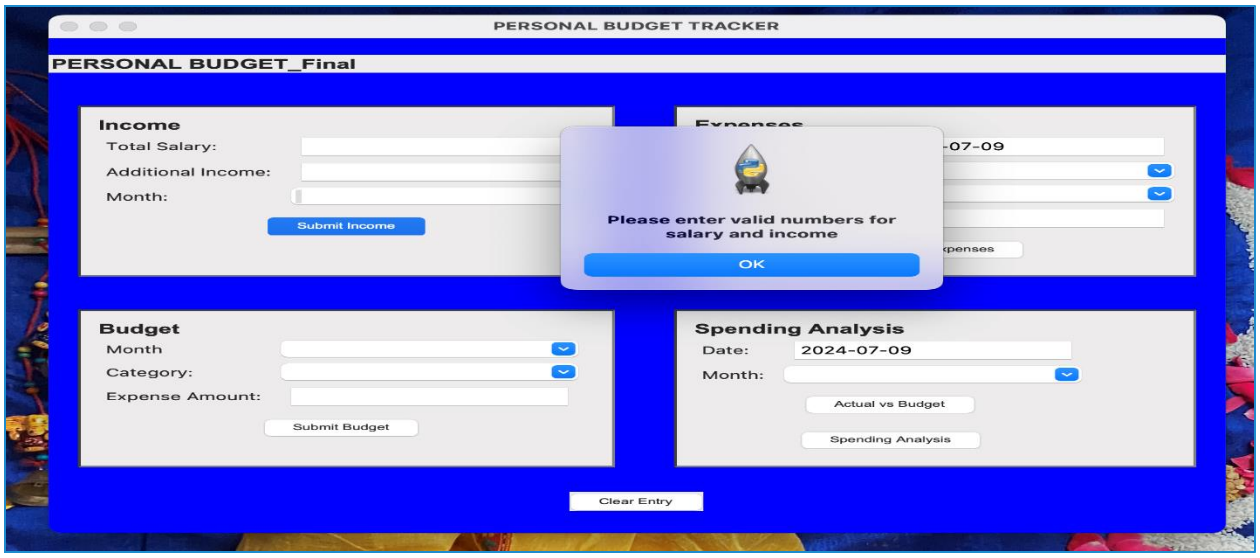
OK

Click on clear button

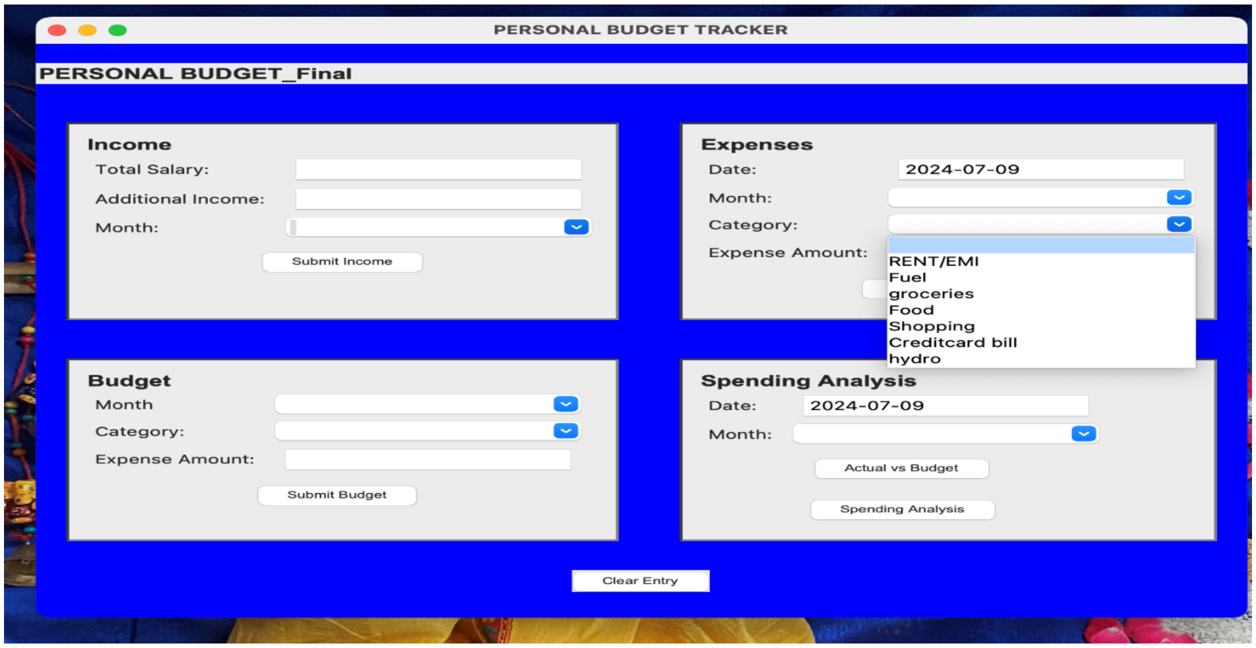
Upon clicking on clear button, it clears the input fields entered.



Click on ‘submit income’ without entering data fields

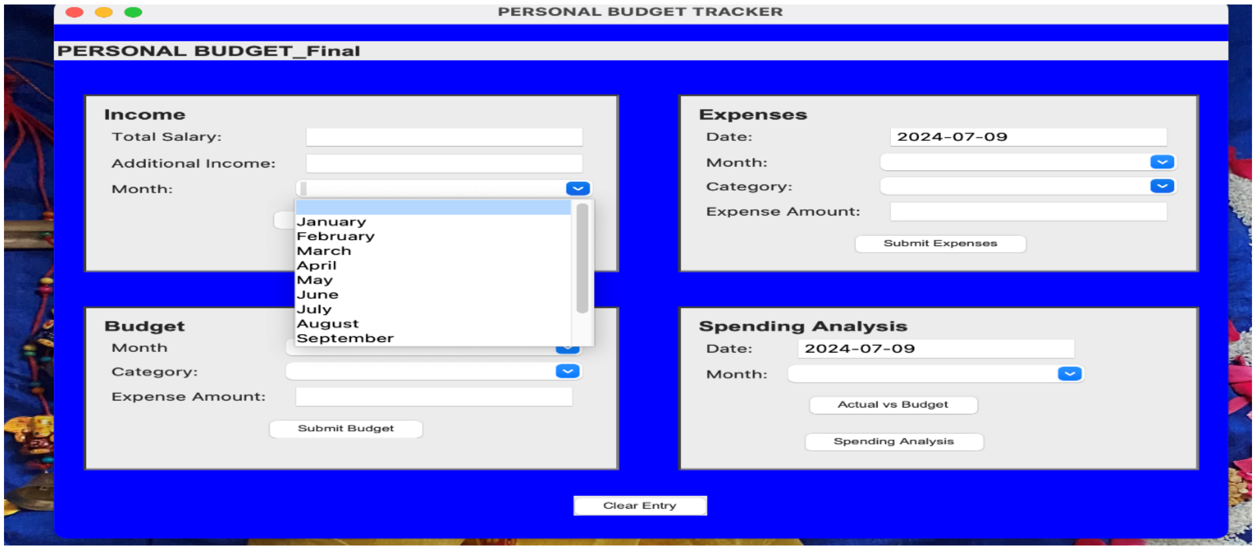


Dropdowns for Month and Category



Date: 2024-07-09

Date: 2024-07-09

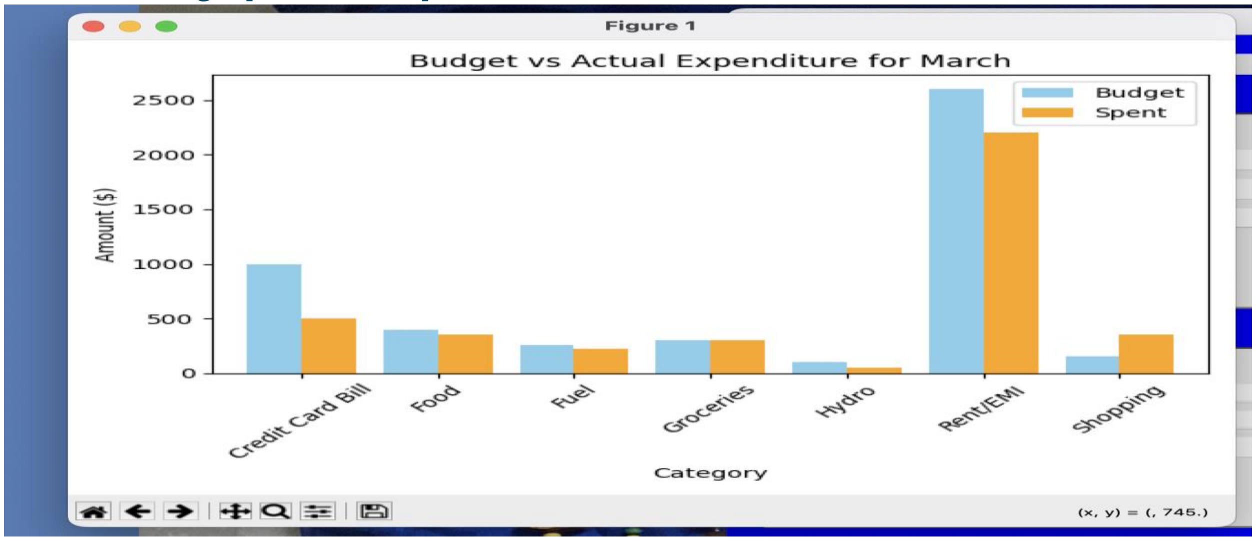


Date: 2024-07-09

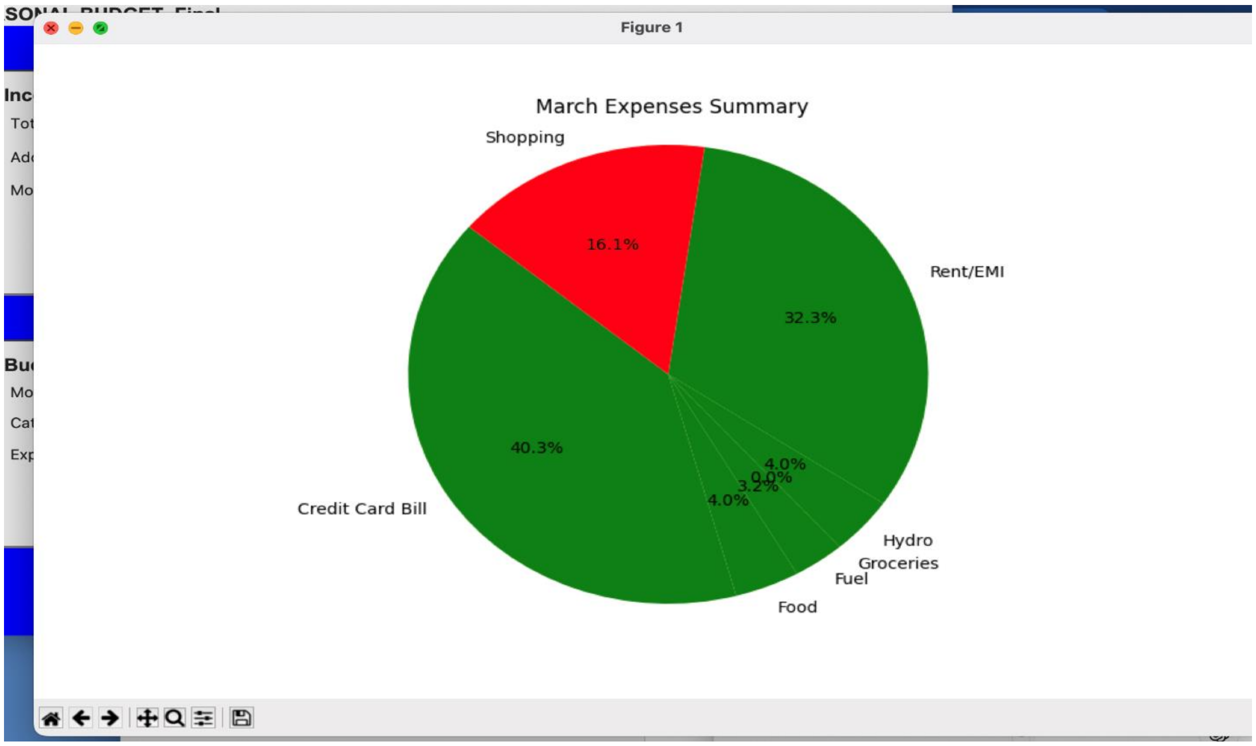
Date: 2024-07-09

Select a month from spending Analysis Frame

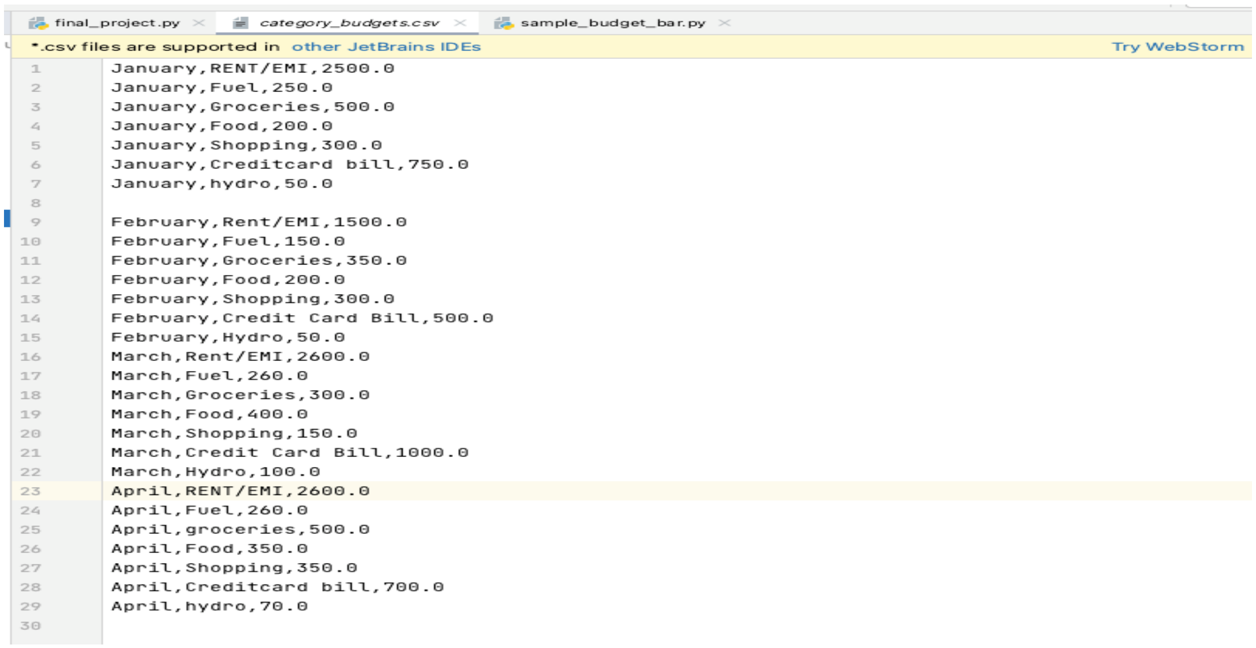
Actual vs Budget [BAR GRAPH]



Expenses summary[pie chart]

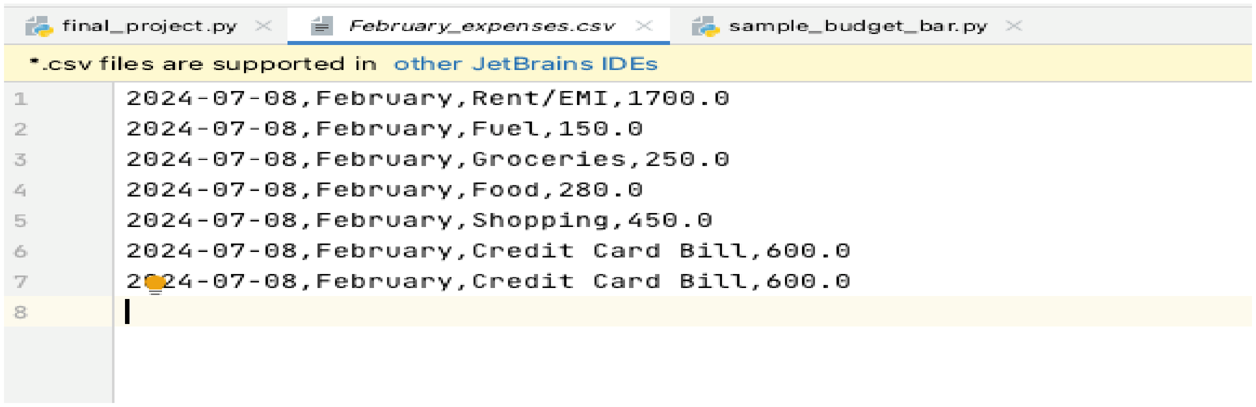


Category_budgets_csv_file[SAMPLE]



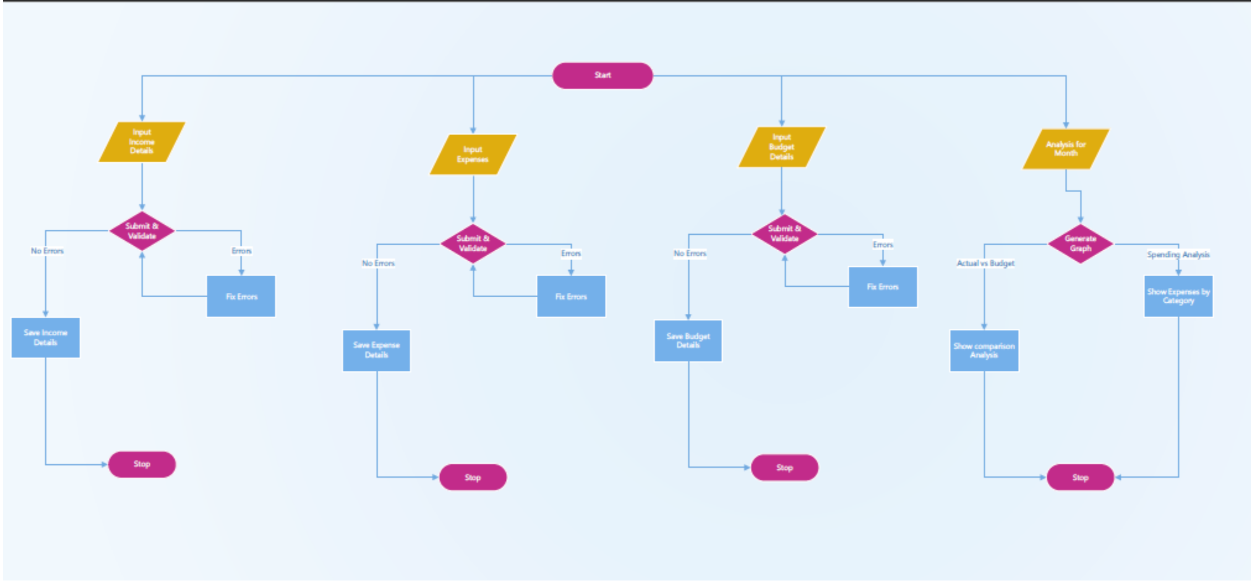
```
January,RENT/EMI,2500.0
January,Fuel,250.0
January,Groceries,500.0
January,Food,200.0
January,Shopping,300.0
January,Creditcard bill,750.0
January,hydro,50.0

February,Rent/EMI,1500.0
February,Fuel,150.0
February,Groceries,350.0
February,Food,200.0
February,Shopping,300.0
February,Credit Card Bill,500.0
February,Hydro,50.0
March,Rent/EMI,2600.0
March,Fuel,260.0
March,Groceries,300.0
March,Food,400.0
March,Shopping,150.0
March,Credit Card Bill,1000.0
March,Hydro,100.0
April,RENT/EMI,2600.0
April,Fuel,260.0
April,groceries,500.0
April,Food,350.0
April,Shopping,350.0
April,Creditcard bill,700.0
April,hydro,70.0
```

February_expenses_csv_file[SAMPLE]



```
2024-07-08,February,Rent/EMI,1700.0
2024-07-08,February,Fuel,150.0
2024-07-08,February,Groceries,250.0
2024-07-08,February,Food,280.0
2024-07-08,February,Shopping,450.0
2024-07-08,February,Credit Card Bill,600.0
2024-07-08,February,Credit Card Bill,600.0
```

FLOW OF THE PERSONAL BUDGET TRACKING APPLICATION



NAMES OF ALL FUNCTIONS AND THEIR INPUT/OUTPUT

Function 1:

def Money_In_Frame():

> ***Input:*** *It takes user entered Input from GUI and stores them to variable.*
> ***Output:*** *function returns a tuple, and the tuple contains Month selected by the user and sum of income and additional income. If any of the input field is empty it throws an error Message and returns **NONE***

Function 2:

def write_money_in_csv ():

> ***Input:*** *The function write_money_in_csv does not take any parameters as input. It relies on the Money_In_Frame function to gather necessary input from the G*
> ***Output:*** *This function does not return any value. Instead, it performs the action of writing income data to a CSV file and displays a success message.*

Function 3:

def Money_out_frame():

> ***Input:*** *The function Money_out_frame does not take any parameters as input. It relies on getting values directly from the GUI components*
>
> ***Output:*** *Returns a tuple (category, date, month, expenditure) if all fields are valid and correctly filled. Returns None if any field is empty or if the expenditure value is not numeric.*

Function 4:

def write_money_out_csv():

Input: *The function write_money_out_csv does not take any parameters as input. It relies on the Money_out_frame function to gather necessary input from the GUI.*

Output: *This function does not return any value. I nstead, it performs the action of writing expense data to a CSV file and displays a success message.*

Function 5:

def clear_function():

Input: *The function clear_function does not take any parameters as input. It directly interacts with the GUI components to clear their contents*

Output: *This function does not return any value. Instead, it performs the action of clearing all input fields in the GUI.*

Function 6:

def write_category_budget_to_csv():

Input: *The function write_category_budget_to_csv does not take any parameters as input. It gathers the necessary input directly from the GUI components.*

***Output**: This function does not return any value. Instead, it performs the action of writing budget data to a CSV file and displays appropriate messages based on the success or failure of the operation.*

Function 7:

def check_month_category_budget_exists(month, category):

Input: month *(str): The month for which the budget is being checked.*
category (str): The category for which the budget is being checked.
Output: True if a budget for the specified month and category already exists. False if a budget for the specified month and category does not exist.

Function 8:

def generate_graph_actual_budget(month):

Input: *This function takes one parameter as input: month (a string representing the month for which the graph is to be generated).*

Output: *This function does not return any value. Instead, it generates and displays a bar chart comparing the budgeted and actual expenditures for the specified month.*
If an error occurs (e.g., the data files are missing), it displays an appropriate error

Function 9:

def generate_pie_chart(month):

Input: *month (str): The month for which the pie chart is to be generated.*
Output: *None The function generates and displays a pie chart and shows an error message if any exceptions occur.*

Function 10:

def exit_application():

Input: *The function exit_application does not take any parameters as input.*
Output: *This function does not return any value. Instead, it performs the action of closing the Tkinter application window.*