विश्वजीवनामृतं ज्ञानम्

# IIITM GWALIOR

## Object Oriented Programming Assignment

# PERSONAL EXPENSE TRACKER

## Members:

1. ANURAG THAKUR (2023IMG-008)
2. KSHITIJ DHAMANIKAR (2023IMG-029)
3. KUSHAGRA AGARWAL (2023IMG-030)
4. VIBHOR KUMAR (2023IMG-054)

# Documentation

## Overview

The Expense Tracker is a simple yet effective tool designed to help users manage their expenses and budgets. It provides functionalities for recording expenses, tracking spending by category, managing budgets, and visualizing spending patterns. This application is developed using object-oriented programming principles in C++.

## Problem Statement

In today's fast-paced life, managing personal finances efficiently is crucial. However, keeping track of expenses and maintaining a budget can be challenging for many individuals. To address this issue, we have developed a Personal Expense Tracker. The Expense Tracker will allow users to record their expenses, manage their budgets, and visualize their spending patterns.

## Features

### 1. Expense Tracking

- Users can record their expenses by providing details such as expense name, amount, date, and category.
- Expenses are categorized into different categories including food, shopping, bills & utilities, fun, and miscellaneous.

### 2. Budget Management

- Users can add money to their budget, which is then used to cover their expenses.
- The application keeps track of the remaining budget after deducting expenses.

### 3. Expense Visualization

- Users can visualize their spending patterns using matplotlib in Python.
- The application provides a simple interface to plot expenses over time.

### 4. Expense and Budget Graph

- Users can see the graphs of budget and expense that user made and keep track of their expenditure with date.
- The application provides interactive and colorful pie chart of expenses category-wise.

## Why did we use Class?

In the provided code for the Expense Tracker application, a class named Finance is used to encapsulate all the functionalities related to expense tracking, budget management, and visualization. Here are some reasons why a class is used in this context:

1. **Encapsulation:** The Finance class encapsulates related data (such as expense details, budget information) and functionalities (such as adding expenses, managing budget) into a single unit. This promotes modularity and allows for better organization of code.

2. **Abstraction:** The class provides an abstraction layer that hides the internal implementation details from the user. Users of the class (in this case, the main function or other parts of the program) interact with it through a well-defined interface (i.e., public member functions) without needing to know the internal workings.

3. **Code Reusability:** By encapsulating related functionalities within a class, the code becomes more reusable. The same class can be instantiated multiple times in different parts of the program or in different projects, without duplicating the code for expense tracking and budget management.

Overall, using a class in the Expense Tracker application helps in achieving better organization, modularity, and maintainability of the codebase, while also adhering to the principles of object-oriented programming.

## Usage

### 1. Initialization:

- Upon starting the application, users are prompted to choose whether to reset data or continue with previous data.
- The application creates or resets expense and budget files accordingly.

## 2. Main Menu:

- Users are presented with a main menu containing the following options:

1. Show Budget
2. Show Expenses
3. Add Money
4. Add Expenses
5. Plot Expenses
6. Exit

## 3. Functionality:

- Users can select an option from the menu to perform various tasks such as viewing budget details, adding money to the budget, recording expenses, and visualizing spending patterns.

## 4. Input Validation:

- The application validates user inputs to ensure correctness and consistency.
- Error messages are displayed for invalid inputs, and users are prompted to enter correct values.

# Contributions

This Expense Tracker project was a collaborative effort of four team members, each contributing to different aspects of the project:

## 1. User Interface and Input Handling:

Contributor: **VIBHOR KUMAR**

- Implemented the user interface including menu options and input handling.
- Ensured user inputs are validated and errors are handled gracefully.

## 2. File Management and Data Persistence:

Contributor: **KSHITIJ DHAMANIKAR**

- Implemented file management functionalities for creating, reading, and writing expense and budget data.
- Ensured data persistence across multiple sessions of the application.

### 3. Expense Tracking and Budget Management:

Contributor: <u>**KUSHAGRA AGARWAL**</u>

- Developed the core functionalities for tracking expenses, managing budgets, and categorizing expenses.
- Implemented algorithms for calculating remaining budget and updating expense records.
- Helped in finding various graphical and logical error.

### 4. Visualization and Integration with Python:

Contributor: <u>**ANURAG THAKUR**</u>

- Integrated matplotlib library in Python for visualizing spending patterns.
- Developed scripts for plotting expenses and generating visualizations based on expense data.
- Error handling was also done by him.

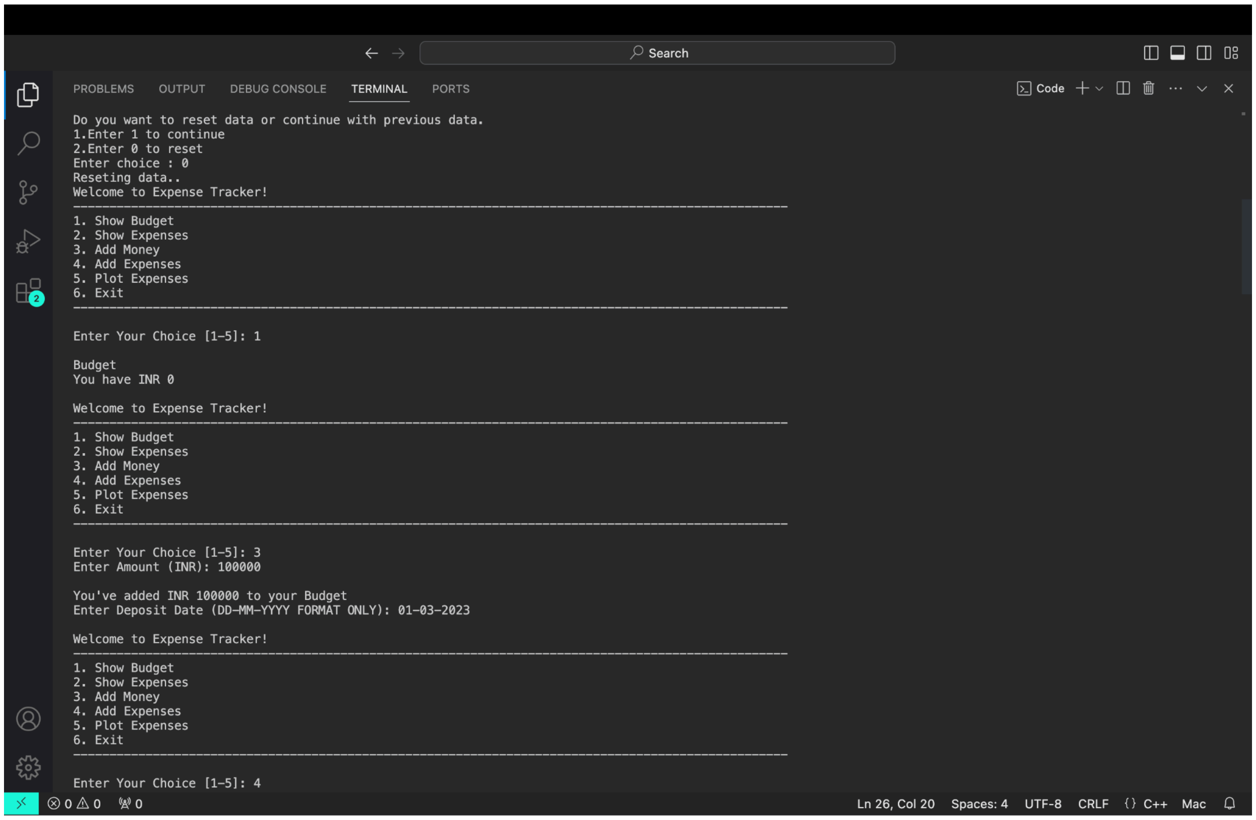
## <u>Conclusion</u>

The Expense Tracker project demonstrates the application of object-oriented programming principles in developing a practical tool for managing expenses and budgets. Through collaborative efforts, the team successfully implemented various functionalities and ensured a user-friendly experience.

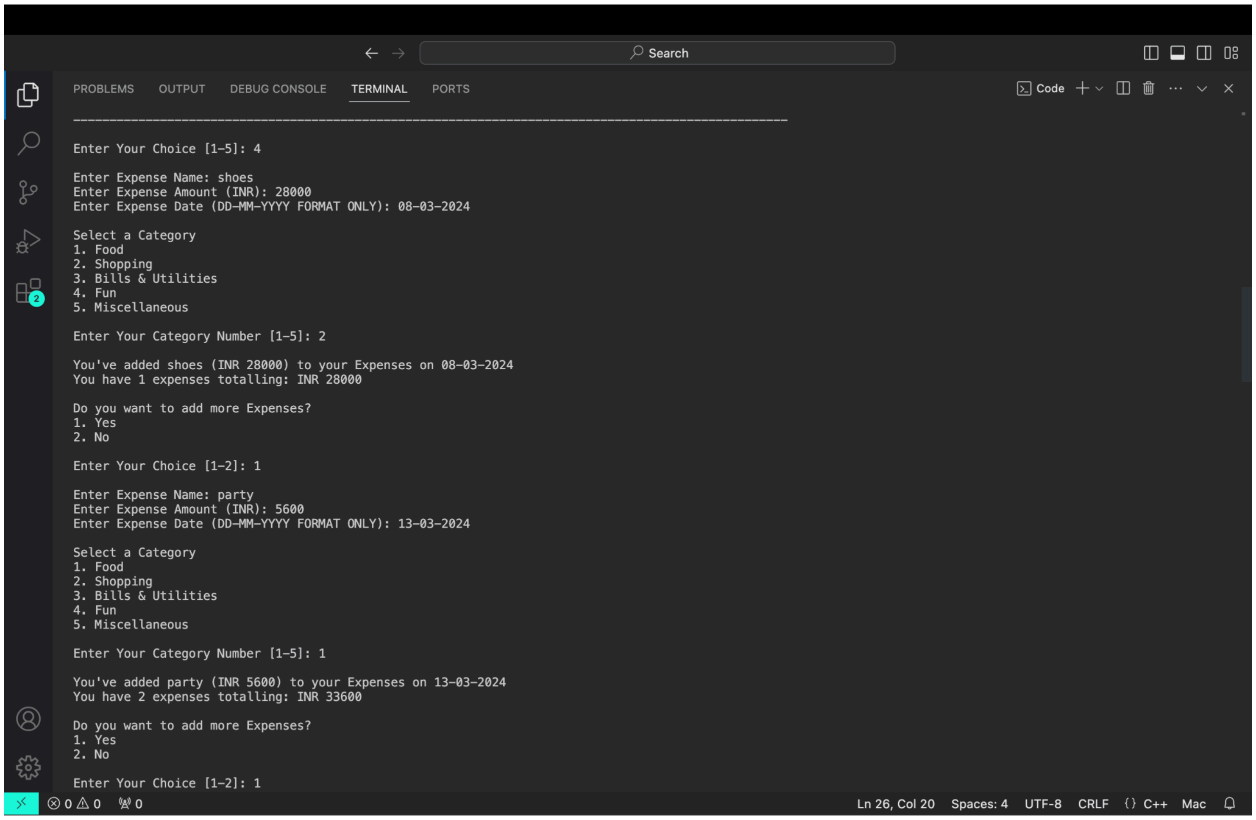
## <u>Members</u>

1. ANURAG THAKUR (2023IMG-008)
2. KSHITIJ DHAMANIKAR (2023IMG-029)
3. KUSHAGRA AGARWAL (2023IMG-030)
4. VIBHOR KUMAR (2023IMG-054)

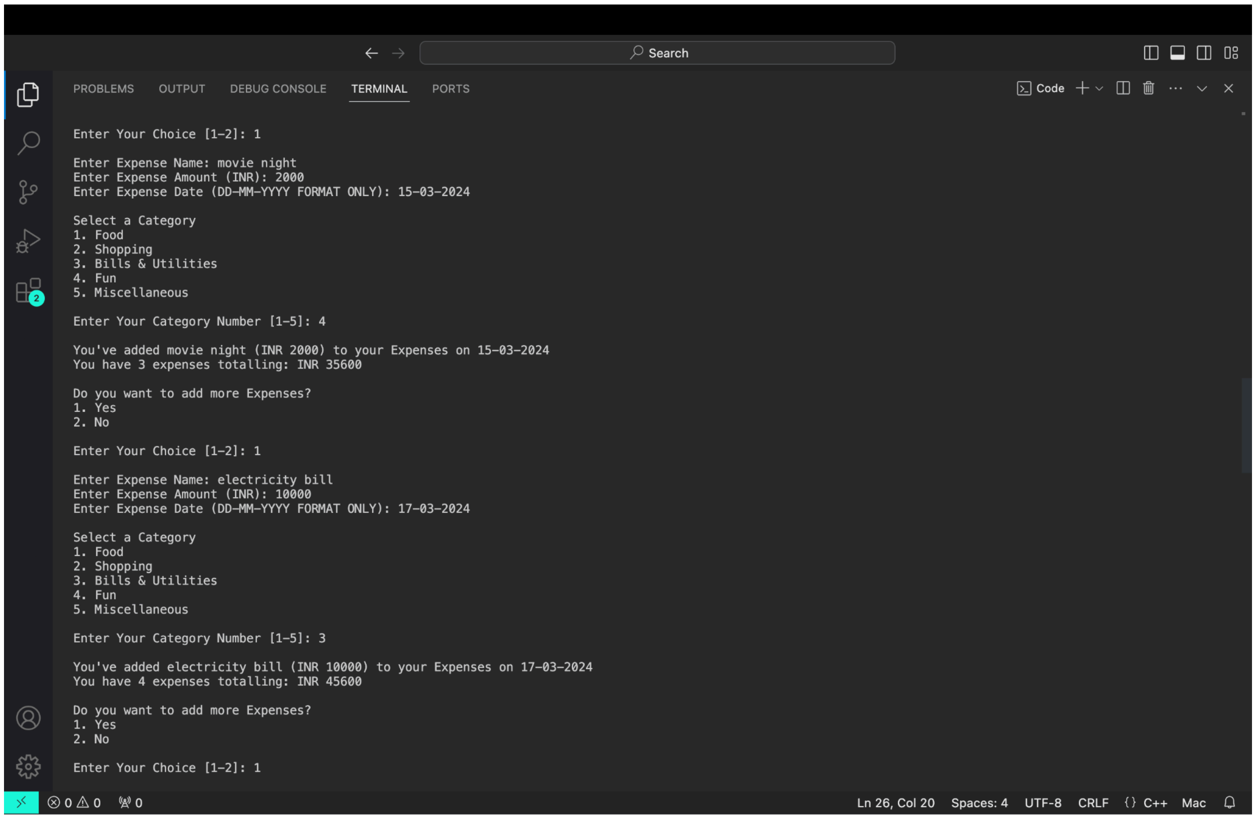
## Output



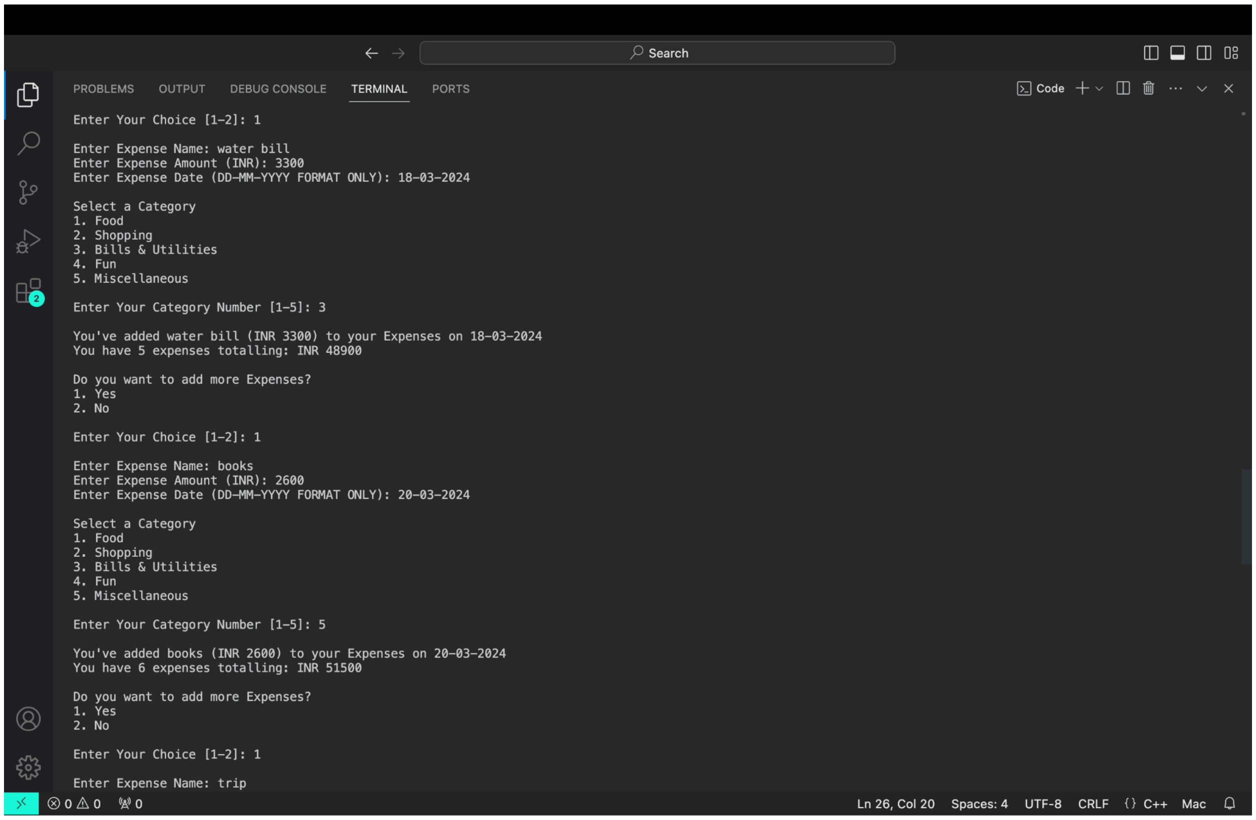
```
Do you want to reset data or continue with previous data.
1.Enter 1 to continue
2.Enter 0 to reset
Enter choice : 0
Reseting data..
Welcome to Expense Tracker!
---------------------------------------------------------------------------------------------------
1. Show Budget
2. Show Expenses
3. Add Money
4. Add Expenses
5. Plot Expenses
6. Exit
---------------------------------------------------------------------------------------------------

Enter Your Choice [1-5]: 1

Budget
You have INR 0

Welcome to Expense Tracker!
---------------------------------------------------------------------------------------------------
1. Show Budget
2. Show Expenses
3. Add Money
4. Add Expenses
5. Plot Expenses
6. Exit
---------------------------------------------------------------------------------------------------

Enter Your Choice [1-5]: 3
Enter Amount (INR): 100000

You've added INR 100000 to your Budget
Enter Deposit Date (DD-MM-YYYY FORMAT ONLY): 01-03-2023

Welcome to Expense Tracker!
---------------------------------------------------------------------------------------------------
1. Show Budget
2. Show Expenses
3. Add Money
4. Add Expenses
5. Plot Expenses
6. Exit
---------------------------------------------------------------------------------------------------

Enter Your Choice [1-5]: 4
```



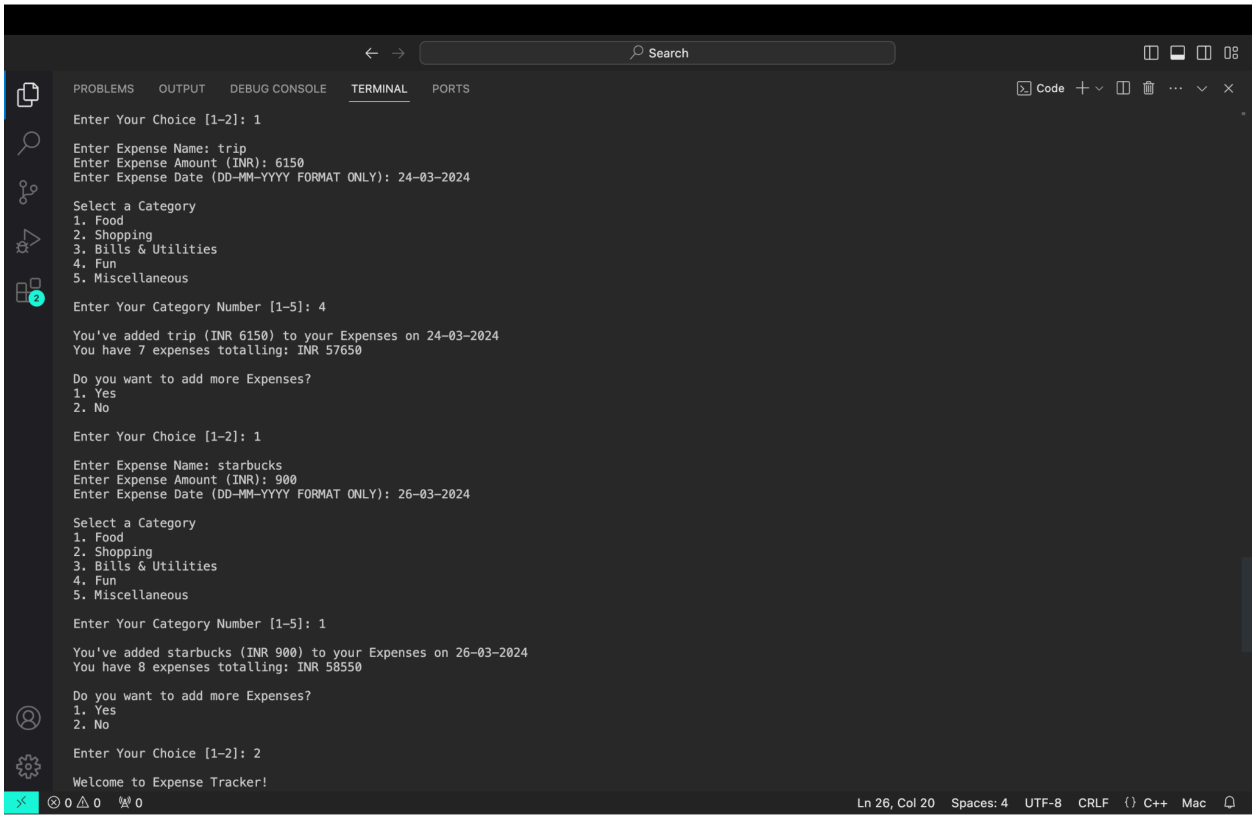
```
---------------------------------------------------------------------------------------------------

Enter Your Choice [1-5]: 4

Enter Expense Name: shoes
Enter Expense Amount (INR): 28000
Enter Expense Date (DD-MM-YYYY FORMAT ONLY): 08-03-2024

Select a Category
1. Food
2. Shopping
3. Bills & Utilities
4. Fun
5. Miscellaneous

Enter Your Category Number [1-5]: 2

You've added shoes (INR 28000) to your Expenses on 08-03-2024
You have 1 expenses totalling: INR 28000

Do you want to add more Expenses?
1. Yes
2. No

Enter Your Choice [1-2]: 1

Enter Expense Name: party
Enter Expense Amount (INR): 5600
Enter Expense Date (DD-MM-YYYY FORMAT ONLY): 13-03-2024

Select a Category
1. Food
2. Shopping
3. Bills & Utilities
4. Fun
5. Miscellaneous

Enter Your Category Number [1-5]: 1

You've added party (INR 5600) to your Expenses on 13-03-2024
You have 2 expenses totalling: INR 33600

Do you want to add more Expenses?
1. Yes
2. No

Enter Your Choice [1-2]: 1
```



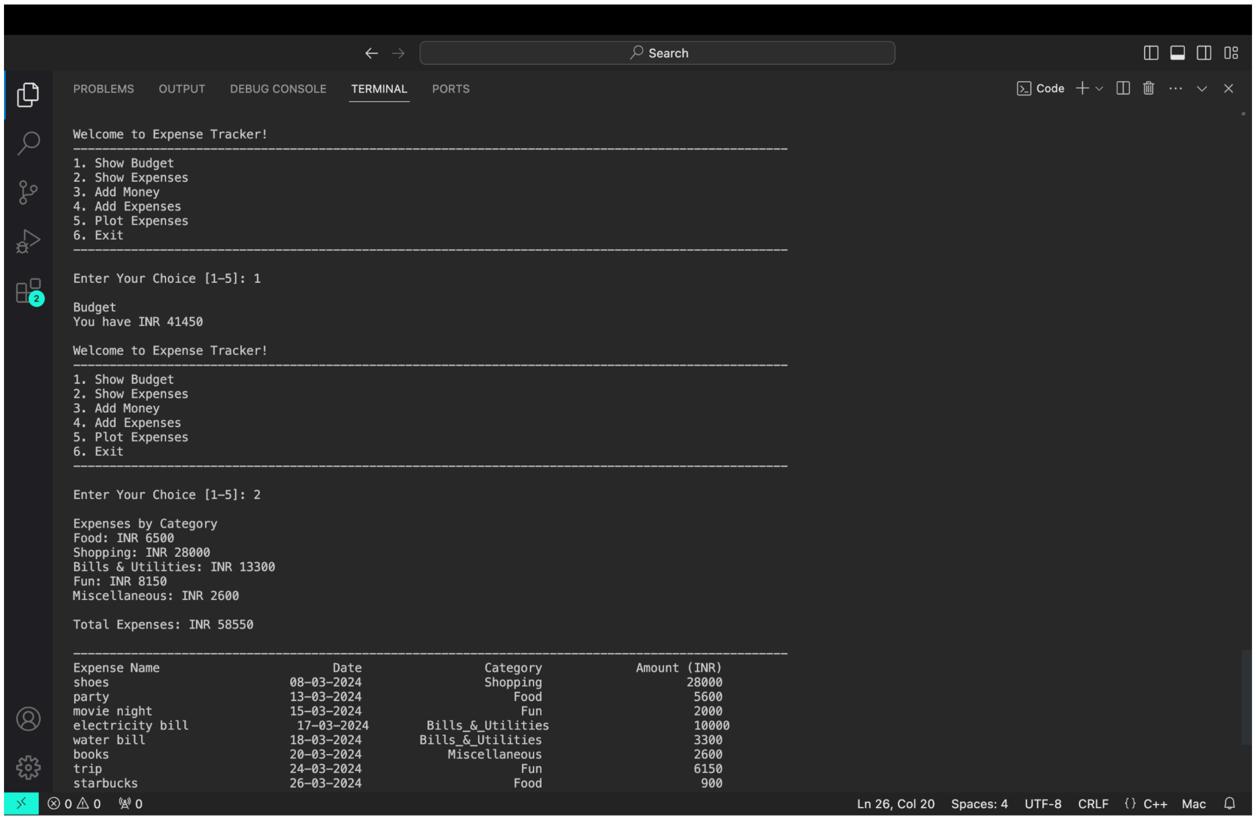
```
Enter Your Choice [1-2]: 1

Enter Expense Name: movie night
Enter Expense Amount (INR): 2000
Enter Expense Date (DD-MM-YYYY FORMAT ONLY): 15-03-2024

Select a Category
1. Food
2. Shopping
3. Bills & Utilities
4. Fun
5. Miscellaneous

Enter Your Category Number [1-5]: 4

You've added movie night (INR 2000) to your Expenses on 15-03-2024
You have 3 expenses totalling: INR 35600

Do you want to add more Expenses?
1. Yes
2. No

Enter Your Choice [1-2]: 1

Enter Expense Name: electricity bill
Enter Expense Amount (INR): 10000
Enter Expense Date (DD-MM-YYYY FORMAT ONLY): 17-03-2024

Select a Category
1. Food
2. Shopping
3. Bills & Utilities
4. Fun
5. Miscellaneous

Enter Your Category Number [1-5]: 3

You've added electricity bill (INR 10000) to your Expenses on 17-03-2024
You have 4 expenses totalling: INR 45600

Do you want to add more Expenses?
1. Yes
2. No

Enter Your Choice [1-2]: 1
```



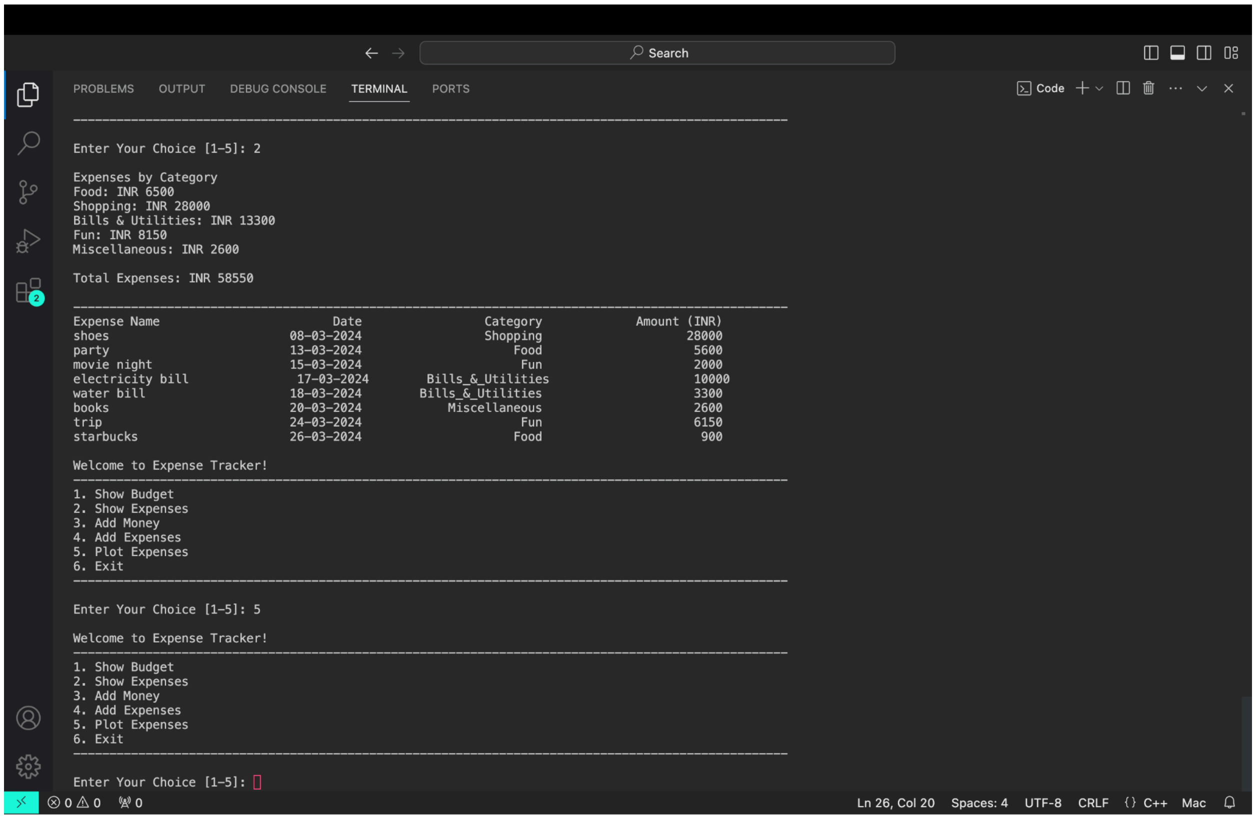
```
Enter Your Choice [1-2]: 1

Enter Expense Name: water bill
Enter Expense Amount (INR): 3300
Enter Expense Date (DD-MM-YYYY FORMAT ONLY): 18-03-2024

Select a Category
1. Food
2. Shopping
3. Bills & Utilities
4. Fun
5. Miscellaneous

Enter Your Category Number [1-5]: 3

You've added water bill (INR 3300) to your Expenses on 18-03-2024
You have 5 expenses totalling: INR 48900

Do you want to add more Expenses?
1. Yes
2. No

Enter Your Choice [1-2]: 1

Enter Expense Name: books
Enter Expense Amount (INR): 2600
Enter Expense Date (DD-MM-YYYY FORMAT ONLY): 20-03-2024

Select a Category
1. Food
2. Shopping
3. Bills & Utilities
4. Fun
5. Miscellaneous

Enter Your Category Number [1-5]: 5

You've added books (INR 2600) to your Expenses on 20-03-2024
You have 6 expenses totalling: INR 51500

Do you want to add more Expenses?
1. Yes
2. No

Enter Your Choice [1-2]: 1

Enter Expense Name: trip
```



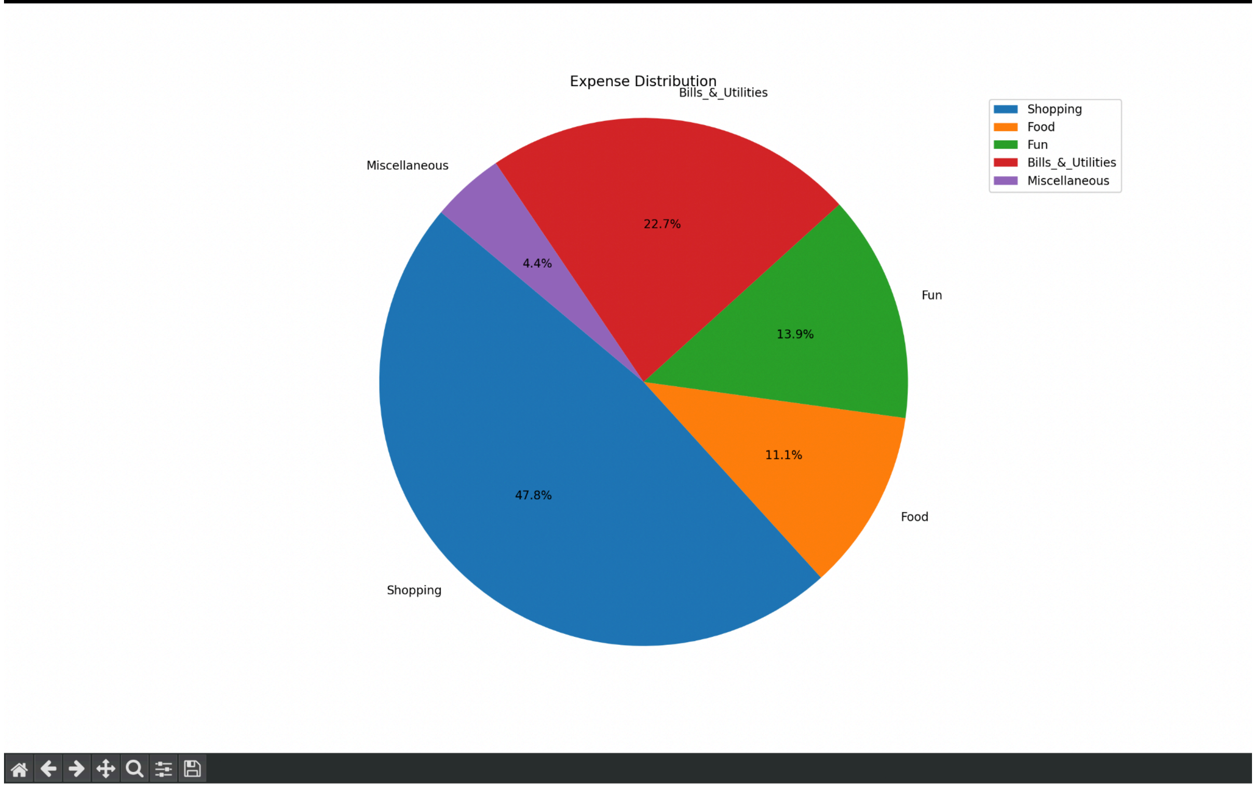
```
Enter Your Choice [1-2]: 1

Enter Expense Name: trip
Enter Expense Amount (INR): 6150
Enter Expense Date (DD-MM-YYYY FORMAT ONLY): 24-03-2024

Select a Category
1. Food
2. Shopping
3. Bills & Utilities
4. Fun
5. Miscellaneous

Enter Your Category Number [1-5]: 4

You've added trip (INR 6150) to your Expenses on 24-03-2024
You have 7 expenses totalling: INR 57650

Do you want to add more Expenses?
1. Yes
2. No

Enter Your Choice [1-2]: 1

Enter Expense Name: starbucks
Enter Expense Amount (INR): 900
Enter Expense Date (DD-MM-YYYY FORMAT ONLY): 26-03-2024

Select a Category
1. Food
2. Shopping
3. Bills & Utilities
4. Fun
5. Miscellaneous

Enter Your Category Number [1-5]: 1

You've added starbucks (INR 900) to your Expenses on 26-03-2024
You have 8 expenses totalling: INR 58550

Do you want to add more Expenses?
1. Yes
2. No

Enter Your Choice [1-2]: 2

Welcome to Expense Tracker!
```

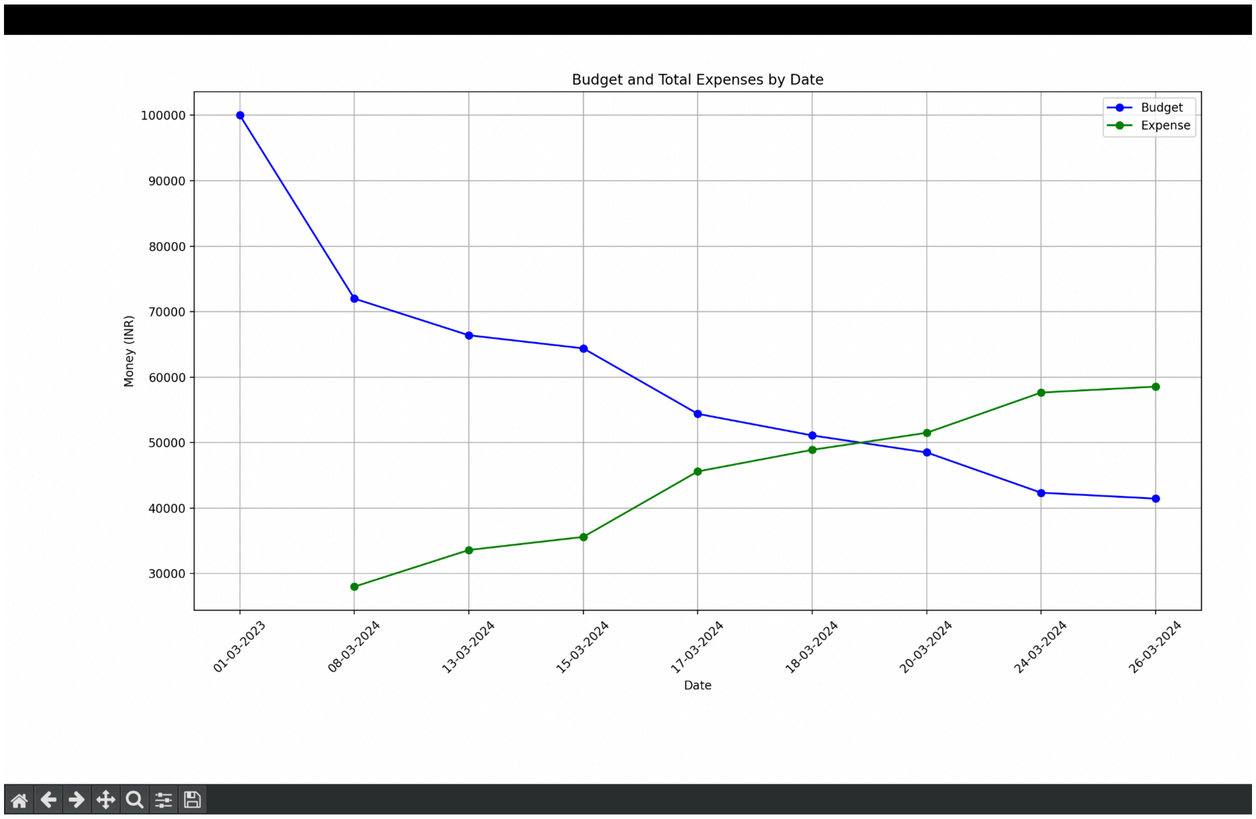


```
Welcome to Expense Tracker!
---------------------------------------------------------------------------------------------------
1. Show Budget
2. Show Expenses
3. Add Money
4. Add Expenses
5. Plot Expenses
6. Exit
---------------------------------------------------------------------------------------------------

Enter Your Choice [1-5]: 1

Budget
You have INR 41450

Welcome to Expense Tracker!
---------------------------------------------------------------------------------------------------
1. Show Budget
2. Show Expenses
3. Add Money
4. Add Expenses
5. Plot Expenses
6. Exit
---------------------------------------------------------------------------------------------------

Enter Your Choice [1-5]: 2

Expenses by Category
Food: INR 6500
Shopping: INR 28000
Bills & Utilities: INR 13300
Fun: INR 8150
Miscellaneous: INR 2600

Total Expenses: INR 58550

---------------------------------------------------------------------------------------------------
Expense Name                        Date                 Category             Amount (INR)
shoes                         08-03-2024                 Shopping                    28000
party                         13-03-2024                     Food                     5600
movie night                   15-03-2024                      Fun                     2000
electricity bill               17-03-2024        Bills_&_Utilities                    10000
water bill                    18-03-2024        Bills_&_Utilities                     3300
books                         20-03-2024            Miscellaneous                     2600
trip                          24-03-2024                      Fun                     6150
starbucks                     26-03-2024                     Food                      900
```



```
--------------------------------------------------------------------------------------------------

Enter Your Choice [1-5]: 2

Expenses by Category
Food: INR 6500
Shopping: INR 28000
Bills & Utilities: INR 13300
Fun: INR 8150
Miscellaneous: INR 2600

Total Expenses: INR 58550

--------------------------------------------------------------------------------------------------
Expense Name                        Date                  Category                 Amount (INR)
shoes                          08-03-2024                 Shopping                        28000
party                          13-03-2024                     Food                         5600
movie night                    15-03-2024                      Fun                         2000
electricity bill                17-03-2024           Bills_&_Utilities                      10000
water bill                     18-03-2024          Bills_&_Utilities                        3300
books                          20-03-2024            Miscellaneous                         2600
trip                           24-03-2024                      Fun                         6150
starbucks                      26-03-2024                     Food                          900

Welcome to Expense Tracker!
--------------------------------------------------------------------------------------------------
1. Show Budget
2. Show Expenses
3. Add Money
4. Add Expenses
5. Plot Expenses
6. Exit
--------------------------------------------------------------------------------------------------

Enter Your Choice [1-5]: 5

Welcome to Expense Tracker!
--------------------------------------------------------------------------------------------------
1. Show Budget
2. Show Expenses
3. Add Money
4. Add Expenses
5. Plot Expenses
6. Exit
--------------------------------------------------------------------------------------------------

Enter Your Choice [1-5]:
```





## Source Code (C++)

```cpp
#include<iostream>
#include<math.h>
#include<string>
#include<fstream>
#include<iomanip>
#include<sstream>

using namespace std;

class Finance {

private:

   string exp_name;
   string exp_date;

   int no_exp = 0;
```

```
    int category;

    float total_expenses = 0;
    float total_expense2 = 0;
    float amt;
    float food = 0;
    float bills = 0;
    float shopping = 0;
    float misc = 0;
    float fun = 0;
    float cash = 0;
    float budget = 0;

    fstream Expenses;
    fstream Budget;


public:

    // Function to create or reset the expense file
    void CreateExpenseFile(int choice) {
        if(!choice){
            cout<<"Reseting data..";
            Expenses.open("Expenses.txt", ios::out);
        }
        else{
            cout<<"Continuing with previous data...";
            Expenses.open("Expenses.txt");
        }
        if (Expenses.is_open()) {
            Expenses << setw(15) << left << "Expense Name"
                     << setw(25) << right << "Date"
                     << setw(25) << right << "Category"
                     << setw(25) << right << "Amount (INR)"<< endl;

            Expenses.close();
        }
    }

    // Function to add expenses to file
    void AddExpensesToFile() {
        Expenses.open("Expenses.txt", ios::app);
```

```cpp
        if (Expenses.is_open()) {
            Expenses << setw(15) << left << exp_name
                     << setw(25) << right << exp_date
                     << setw(25) << right << getCategoryName(category)
                     << setw(25) << right << amt<< endl;

            Expenses.close();
        }
    }

    // Function to create or reset the budget file
    void CreateBudgetFile(int choice) {

        if(!choice){
            Budget.open("Budget.txt", ios::out);
        }
        else{
            Budget.open("Budget.txt");
            //budget= last element of budget
            ifstream file("Budget.txt");
            string line;
            while (getline(file,line)) {
                stringstream ss(line);
                float bud;
                ss >> bud;
                if (ss) {
                    budget = bud; // Update last budget only if conversion was
successful
                }
            }
        }
        if (Budget.is_open()) {
            Budget << setw(15) << left << "Budget (INR)"
                   <<setw(15)<<right<<"Date"<< endl;
            Budget.close();
        }
    }

    // Function to add expenses to file
    void AddBudgetToFile() {
        Budget.open("Budget.txt", ios::app);
        if (Budget.is_open()) {
```

```cpp
        Budget << setw(15) << left << budget
               <<setw(15)<<right<<exp_date<< endl;
        Budget.close();
    }
}

// Function to add money to the budget
void AddMoney() {
    cout << "Enter Amount (INR): ";
    cin >> cash;
    Cash:
    if (cash <= 0) {
        cout << "Invalid Input!" << endl;
        goto Cash;
    }
    cout << "\nYou've added INR " << cash << " to your Budget" << endl;
    budget += cash;

    Date:
    exp_date="";
    cout << "Enter Deposit Date (DD-MM-YYYY FORMAT ONLY): ";
    // cin.ignore();
    // getline(cin, exp_date);
    cin>>exp_date;
    if (exp_date.length() != 10 || exp_date[2] != '-' || exp_date[5] != '-') {
        cout<<"\nWrong format"<<endl;
        goto Date;
    }

    AddBudgetToFile();
}

// Function to add an expense
void AddExpense() {
    AddExpense:
    cout << "\nEnter Expense Name: ";
    cin.ignore();
    getline(cin, exp_name);

    Amt:
    cout << "Enter Expense Amount (INR): ";
    cin >> amt;
```

```cpp
        if (amt <= 0) {
            cout << "Invalid Input!" << endl;
            goto Amt;
        }

        Date:
        exp_date="";
        cout << "Enter Expense Date (DD-MM-YYYY FORMAT ONLY): ";
        cin>>exp_date;
        if (exp_date.length() != 10 || exp_date[2] != '-' || exp_date[5] != '-') {
            cout<<"\nWrong format"<<endl;
            goto Date;
        }


        cout << "\nSelect a Category" << endl;
        cout << "1. Food" << endl;
        cout << "2. Shopping" << endl;
        cout << "3. Bills & Utilities" << endl;
        cout << "4. Fun" << endl;
        cout << "5. Miscellaneous" << endl;

        Category:
        cout << "\nEnter Your Category Number [1-5]: ";
        cin >> category;
        if (category < 1 || category > 5) {
            cout << "Invalid Input!" << endl;
            goto Category;
        }

        // Update total expenses and category expenses
        total_expenses += amt;
        // total_expense2 += amt; //for budget
        no_exp++;
        switch(category) {
            case 1:
                food += amt;
                break;
            case 2:
                shopping += amt;
                break;
            case 3:
                bills += amt;
                break;
            case 4:
                fun += amt;
                break;
            case 5:
                misc += amt;
                break;
        }

        budget -= amt;
        // Add expenses to file
        AddExpensesToFile();
        AddBudgetToFile();

        cout << "\nYou've added " << exp_name << " (INR " << amt << ") to your Expenses on "<<exp_date<<" "<< endl;
        cout << "You have " << no_exp << " expenses totalling: INR " << total_expenses << endl;


        // Ask user if they want to add more expenses
        int addmore;
        cout << "\nDo you want to add more Expenses?" << endl;
        cout << "1. Yes" << endl;
        cout << "2. No" << endl;

        Addmore:
        cout << "\nEnter Your Choice [1-2]: ";
        cin >> addmore;
        if (addmore != 1 && addmore != 2) {
            cout << "Invalid Input!" << endl;
            goto Addmore;
        }
        if (addmore == 1)
            goto AddExpense;
    }


    // Function to display the remaining budget
    void DisplayBudget() {
       // budget -= total_expense2;
```

```cpp
        if (budget < 0) {
            cout << "\n!!!Budget Overdrawn!!!" << endl;
        }
        // total_expense2 = 0;
        cout << "\nBudget" << endl;
        cout << "You have INR " << budget << endl;
    }

    // Function to display expenses by category
    void DisplayExpense() {
        cout << "\nExpenses by Category" << endl;
        cout << "Food: INR " << food << endl;
        cout << "Shopping: INR " << shopping << endl;
        cout << "Bills & Utilities: INR " << bills << endl;
        cout << "Fun: INR " << fun << endl;
        cout << "Miscellaneous: INR " << misc << endl;
        cout << "\nTotal Expenses: INR " << total_expenses << endl;
        // Display all expenses from file
        cout <<
"\n------------------------------------------------------------------------------------
----------------" << endl;
        DisplayAllExpenses();
    }



    // Function to get category name from category number
    string getCategoryName(int category) {
        switch(category) {
            case 1:
                return "Food";
            case 2:
                return "Shopping";
            case 3:
                return "Bills_&_Utilities";
            case 4:
                return "Fun";
            case 5:
                return "Miscellaneous";
            default:
                return "Unknown";
        }
```

```
   }

   // Function to display all expenses from file with category
   void DisplayAllExpenses() {
       Expenses.open("Expenses.txt", ios::in);
       if (Expenses.is_open()) {
           string line;
           while (getline(Expenses, line)) {
               cout << line << endl;
           }
           Expenses.close();
       }
   }



   // Function to plot expenses using matplotlib in Python
   void PlotExpenses() {
       system("python3 plot_expenses.py");
   }
};

// Main function
int main() {
   Finance f;
   int choice,ch;
   cout<<"Do you want to reset data or continue with previous data.\n1.Enter 1 to
continue\n2.Enter 0 to reset\nEnter choice : ";
   cin>>ch;
   f.CreateExpenseFile(ch);
   f.CreateBudgetFile(ch);
   if(!ch)
       f.AddBudgetToFile();




while (true){

   cout << "\nWelcome to Expense Tracker!" << endl;
   cout <<
"-------------------------------------------------------------------------------------
---------------" << endl;
   cout << "1. Show Budget" << endl;
```

```cpp
    cout << "2. Show Expenses" << endl;
    cout << "3. Add Money" << endl;
    cout << "4. Add Expenses" << endl;
    cout << "5. Plot Expenses" << endl;
    cout << "6. Exit" << endl;
    cout <<
"-------------------------------------------------------------------------------------
--------------" << endl;

    cout << "\nEnter Your Choice [1-5]: ";
    cin >> choice;
    if (choice < 1 || choice > 6) {
        cout << "Invalid Input!" << endl;
        continue;
    }

    switch(choice) {
        case 1:
            f.DisplayBudget();
            break;
        case 2:
            f.DisplayExpense();
            break;
        case 3:
            f.AddMoney();
            break;
        case 4:
            f.AddExpense();
            break;
        case 5:
            f.PlotExpenses();
            break;
        case 6:
            return 0;
    }
}
return 0;
}
```

## Source Code (Python)

```
import matplotlib.pyplot as plt
from datetime import datetime

# Read expenses from file
with open("Expenses.txt", "r") as file:
   expenses = file.readlines()
   expenses.pop(0)   # Remove header

# Extract expense names and amounts
category_name = [expense.split()[-2] for expense in expenses]
expense_amounts = [float(expense.split()[-1]) for expense in expenses]
cat_dict={}
for i in range(len(category_name)):
   if category_name[i] not in cat_dict.keys():
       cat_dict[category_name[i]]=expense_amounts[i]
   else:
       cat_dict[category_name[i]]+=expense_amounts[i]

# Calculate total expense
total_expense = sum(expense_amounts)
category_name=list(cat_dict.keys())
expense_amounts2=list(expense_amounts)
expense_amounts=list(cat_dict.values())
# Calculate percentage for each expense category
percentages = [(amount / total_expense) * 100 for amount in expense_amounts]

# Plot expenses as a pie chart
plt.figure(figsize=(8, 8))
plt.pie(percentages, labels=category_name, autopct='%1.1f%%', startangle=140)
plt.axis('equal')
plt.title('Expense Distribution')
plt.legend()
plt.show()

with open("Budget.txt","r") as file:
   budget=file.readlines()
   budget.pop(0)
   budget.pop(0)
```

```python
bud=[float(b.split()[0]) for b in budget]
date_budget=[b.split()[-1] for b in budget]
date_exp=[e.split()[-3] for e in expenses]
# Example: ["100 2024-04-01", "200 2024-04-01", "150 2024-04-02"]

budget_dict = {}
expense_dict={}

# Iterate over each budget entry and update the budget dictionary
for entry in range(len(expense_amounts2)):
   amount=expense_amounts2[entry]
   date=date_exp[entry]
   if date in expense_dict:
       # If the date already exists in the dictionary, add the amount to the existing
value
       expense_dict[date] += float(amount)
   else:
       # If the date doesn't exist in the dictionary, initialize it with the amount
       expense_dict[date] = float(amount)

expense_dict = dict(sorted(expense_dict.items(),key=lambda x: datetime.strptime(x[0],
'%d-%m-%Y')))
e_dates = list(expense_dict.keys())
expense_amounts=list(expense_dict.values())
# Calculate cumulative expenses
cumulative_expenses = [sum(expense_amounts[:i+1]) for i in
range(len(expense_amounts))]

for entry in range(len(bud)):
   amount=bud[entry]
   date=date_budget[entry]
   budget_dict[date] = float(amount)

budget_dict = dict(sorted(budget_dict.items(),key=lambda x: datetime.strptime(x[0],
'%d-%m-%Y')))
b_dates = list(budget_dict.keys())
budget_amounts = list(budget_dict.values())

# Plot the budget by date as a line graph
plt.plot(b_dates, budget_amounts, label="Budget",marker='o', color='b', linestyle='-')
plt.plot(e_dates, cumulative_expenses, label="Expense",marker='o', color='g',
linestyle='-')
```

```python
plt.xlabel('Date')
plt.ylabel('Money (INR)')
plt.title('Budget and Total Expenses by Date')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.grid(True)
plt.tight_layout()
plt.legend()
plt.show()
```