Midterm Project Report
Advanced Computer Programming

Web Services with Python

Student Name : Maximilliano Felix Gunawan

Student ID : 112021182

Teacher : DINH-TRUNG VU

2024-04

Chapter 1 Introduction

1.1 Github

1) **Personal Github Account** : FullyMed (Max. Felix)
2) **Group Github Account** : FullyMed
3) **Group Project Repository** : Amigo
4) **List of submitted file**s :
    - **Maximilliano Felix Gunawan_112021182.py**

1.2 Topic

We choose the topic of web services for finance tracker. We chose this topic to make it easier for people to record their finances and targets for saving and targets for outcome. And my part is Add / Remove / Edit Expenses.

Project Overview

The purpose of this project is to create a simple expense management system that allows users to track their expenses, perform operations such as adding, removing, and editing expenses, calculate average expenditures, and visualize expenses over time. In the first display you will be shown a program for entering spending limits, then after that 6 options will be displayed which can be selected according to the user's wishes. Starting from Add Expense, Remove Expense, Edit Expense, Calculate Average Expenditures, Plot Expenses Over Time, and the last is Finish, these are all programs for expenses. there is a program to increase income, there is also a program to calculate the remaining money, for income, we will combine all of that in the final project. In the current project I present only expenses, for further explanation of my program, I have explained it in the implementation section.

Chapter 2 Implementation

2.1 Class 1

The code defines a class named **Money_Management** which serves as a tool for managing expenses within a specified budget. It includes methods to add, remove, edit expenses, and calculate the average monthly expenditures. Additionally, there's a **main()** function that acts as the entry point for the program, providing a user-friendly menu interface for interacting with the **Money_Management** class. Also it's easier for the user to use, what they only need to do is to call the class(), which one they want and which one they need.

Features :

- **Expense Management :** Users can add, remove, and edit expenses, ensuring they stay within their budget.
- **Budget Limitation :** The class enforces a maximum spending limit per month, preventing expenses from exceeding this limit.
- **Average Calculation :** Users can calculate the average monthly expenses based on the recorded expenditures.
- **User Interface :** The **main()** function presents a menu-driven interface, allowing users to easily navigate and utilize the expense management functionalities.

2.1.1 Fields

- **Outcome** : This field is a list that stores the expenses entered by the user.
- **Max_outcome** : This field represents the maximum spending limit per month entered by the user.
- **Timestamps** : This field is a list that stores the timestamps corresponding to each expense entry.

2.1.2 Methods

- **__init__(self, max_outcome)** : Constructor method that initializes a **Money_Management** object with the provided maximum spending limit

(**max_outcome**) and empty lists for storing expenses and timestamps.

- **Add_outcome(self, total_outcome)** : This method adds an expense (**total_outcome**) to the list of expenses if it does not exceed the maximum spending limit. It appends the expense amount to the **outcome** list and the current timestamp to the **timestamps** list if the expense is successfully added.

- **Remove_outcome(self, index)** : This method removes an expense from the list based on the provided index. It checks if the index is valid (within the range of the **outcome** list) and removes the expense at that index along with its corresponding timestamp.

- **Edit_outcome(self, index, new_amount)** : This method edits an expense based on the provided index and new amount. It updates the expense amount at the specified index in the **outcome** list and updates the corresponding timestamp to the current time.

- **Calculate_avg(self)** : This method calculates the average monthly expenses. It sums up all the expenses stored in the **outcome** list and divides by the number of expenses to get the average.

- **Plot_expenses_over_time(self)** : This method plots the expenses over time using matplotlib. It converts the timestamps stored in **self.timestamps** to datetime objects and plots the expenses against these timestamps. The plot shows the trend of expenses over time, helping visualize spending patterns.

2.1.3 Functions

- **__init__(self, max_outcome) :**

- **Purpose** : Initializes a new instance of the **Money_Management** class.

- **Functionality** : Sets the maximum spending limit (**max_outcome**) provided by the user. Initializes empty lists (**outcome** and **timestamps**) to store expenses and corresponding timestamps.

- **Add_outcome(self, total_outcome) :**

- **Purpose** : Adds a new expense to the list of expenses.

- **Functionality** : Checks if the total expense (**total_outcome**) exceeds the maximum spending limit (**max_outcome**). If the expense is within the limit, appends the expense amount to the **outcome** list. Adds the current timestamp to the **timestamps** list. Prints a success message if the expense is added, otherwise prints a message indicating that the expenses exceed the maximum limit.

- **Remove_outcome(self, index) :**

- **Purpose** : Removes an expense from the list based on the provided index.

- **Functionality** : Checks if the provided index is within the range of valid indices for the **outcome** list. If the index is valid, removes the expense amount and corresponding timestamp at that index. Prints a success message if the expense is removed, otherwise prints a message indicating an invalid index.

- **Edit_outcome(self, index, new_amount) :**

- **Purpose** : Modifies an existing expense amount.

- **Functionality** : Checks if the provided index is within the range of valid indices for the **outcome** list. If the index is valid, updates the expense amount at that index with the new amount. Updates the corresponding timestamp to the current time. Prints a success message if the expense is edited, otherwise prints a message indicating an invalid index.

- **Calculate_avg(self) :**

- **Purpose** : Calculates the average monthly expenses.

- **Functionality** : Calculates the average by summing up all the expenses stored in the **outcome** list and dividing by the total number of expenses. Returns the average monthly expenses. If there are no expenses, it returns 0.

- **Plot_expenses_over_time(self) :**

- **Purpose** : Visualizes expenses over time using a line plot.

- **Functionality** : Converts the timestamps stored in **self.timestamps** to datetime objects. Plots the expenses against these timestamps using matplotlib. Sets appropriate labels for the x-axis and y-axis. Rotates the x-axis labels for better readability. Displays the plot using **plt.show()**.

Chapter 3 Results

3.1 Result

Enter the maximum spending limit per month : 800000

Menu:

1. Add Expense
2. Remove Expense
3. Edit Expense
4. Calculate Average Expenditures
5. Plot Expenses Over Time
6. Finish

Choose (1/2/3/4/5/6): 1

Enter the expenditure amount : 1000000

Expenses exceed maximum limit!

Menu:

1. Add Expense
2. Remove Expense
3. Edit Expense
4. Calculate Average Expenditures
5. Plot Expenses Over Time
6. Finish

Choose (1/2/3/4/5/6): 1

Enter the expenditure amount : 10000

Expense added successfully on 2024-04-17 11:05:20: 10000.0

Menu:

1. Add Expense
2. Remove Expense
3. Edit Expense
4. Calculate Average Expenditures

5. Plot Expenses Over Time

6. Finish

Choose (1/2/3/4/5/6): 1

Enter the expenditure amount : 20000

Expense added successfully on 2024-04-17 11:05:26: 20000.0

Menu:

1. Add Expense

2. Remove Expense

3. Edit Expense

4. Calculate Average Expenditures

5. Plot Expenses Over Time

6. Finish

Choose (1/2/3/4/5/6): 1

Enter the expenditure amount : 30000

Expense added successfully on 2024-04-17 11:05:36: 30000.0

Menu:

1. Add Expense

2. Remove Expense

3. Edit Expense

4. Calculate Average Expenditures

5. Plot Expenses Over Time

6. Finish

Choose (1/2/3/4/5/6): 4

Average monthly expenses : 20000.0

Menu:

1. Add Expense

2. Remove Expense

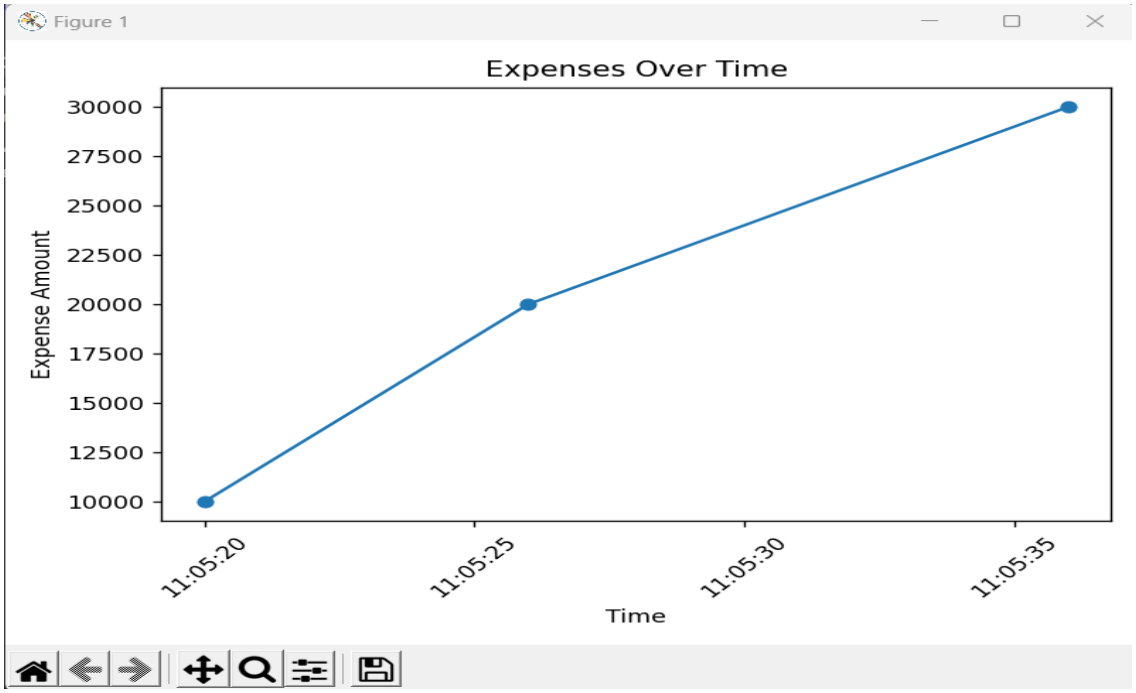
3. Edit Expense

4. Calculate Average Expenditures

5. Plot Expenses Over Time

6. Finish

Choose (1/2/3/4/5/6): 5



Menu:

1. Add Expense
2. Remove Expense
3. Edit Expense
4. Calculate Average Expenditures
5. Plot Expenses Over Time
6. Finish

Choose (1/2/3/4/5/6): 2

Enter the index of the expense to remove: 0

Expense 10000.0 at 2024-04-17 11:05:20 removed successfully.

Menu:

1. Add Expense
2. Remove Expense
3. Edit Expense
4. Calculate Average Expenditures
5. Plot Expenses Over Time
6. Finish

Choose (1/2/3/4/5/6): 4

Average monthly expenses : 25000.0

Menu:
1. Add Expense
2. Remove Expense
3. Edit Expense
4. Calculate Average Expenditures
5. Plot Expenses Over Time
6. Finish
Choose (1/2/3/4/5/6): 3
Enter the index of the expense to edit: 1
Enter the new amount: 20000
Expense edited successfully.

Menu:
1. Add Expense
2. Remove Expense
3. Edit Expense
4. Calculate Average Expenditures
5. Plot Expenses Over Time
6. Finish
Choose (1/2/3/4/5/6): 4
Average monthly expenses : 20000.0

Menu:
1. Add Expense
2. Remove Expense
3. Edit Expense
4. Calculate Average Expenditures
5. Plot Expenses Over Time
6. Finish
Choose (1/2/3/4/5/6): 6
Program finished.

Chapter 4 Conclusions

The conclusion is that I made a program to first enter a spending target, then we can choose the option to increase spending every day (Add Expense). I also added a feature to replace expenses if an error occurs when entering data (Edit Expense). Not only that, I also provide a feature to delete expenses if a refund occurs or you want to record them separately (Remove Expense). Then, I also didn't forget to provide the option to calculate the average so that users can calculate the average expenditure per day or month or year (Calculate Average Expenditures). Overall, the Expense Management System offers users a convenient way to track their expenses, analyze spending patterns, and stay within their budget. With its intuitive interface and powerful features, the system empowers users to take control of their finances and achieve their financial goals effectively.