Personal Budget Tracker

Aye Chan Maw, Washma Ali, Zarni Kaung Htut Kyaw

**Abstract:**

The personal budget tracker is a website application that is designed to track an individual’s personal finance as the name suggests. In this report, we delve into a person’s spending of their money and generate the categorical expense and remaining budget. Unfortunately, due to the limited scope of our skill sets, expanding the functionality of the tracker beyond our current capabilities is a huge struggle. Our approach involved creating the least amount of financial information to begin with and tracking the person’s daily expenses. We employed simple data analysis techniques and statistical graphs to identify the patterns of the person’s expenses although the app needs many improvements.

## **Introduction:**

In this era where the patterns of people’s spending are changing dynamically, it becomes challenging for the individual to manage their finances. Although the bank transactions of a personal use can be tracked easily, the ability to track their categorical spending and plan their financial goals is essential. With the advances of technology, the approach where we keep track of our budget has evolved into sophisticated digital solutions offering accuracy, convenience and real-time results. This report delves into the creation and functionality of a personal budget tracker built by employing the Flask framework which combines Python programming language and HTML and the Matplotlib library.

Throughout this report, we will explore the architecture, features, and implementation of the personal budget tracker, highlighting its capabilities in organising finances. This project is worth doing as its application is a practical need to gain control of the individuals’ finances and to work towards their financial goals. Zhu (2023) mentioned that such systems encourage the users for better financial attitudes and planning, by engaging more deeply with their financial data.

However, does it really equip the individual with the skills to gain control of their finances? Would that kind of application help the person to escape the financial difficulties? Well, the answer would be ‘no’, but the individual can make informed choices when they spend money because its purpose is to make them more aware of their financial situation. It will not help the person to get out of a tight budget, instead it can probably help to prevent them from getting into financial problems.

For this Python-based budget tracker project, we aim to provide a user-friendly application that allows the user to submit their expenses along with their descriptions and view their financial data dynamically. Moreover, we provide dynamic effective graphical visualisations such as pie charts and bar graphs, helping the user in understanding the behaviours of their spending.

In this report, we discussed the introduction in the first section, followed by a brief literature review and identification of gaps in the second section. Subsequently, we outline the Methodology section which includes subsections on the data presentation. In this section, I explore the categorical variables. Finally, we conclude this report with the Summary section.

## **Literature Review and Gaps**

Many studies and articles have explored the different aspects of personal budgeting, giving insights into its benefits, challenges and role of technology in effective financial management. Many people designed the budget or finance tracker by employing excel, python or javascript, aiming to achieve greater financial awareness and better financial management. Python, as a programming language, became a popular choice for developing personal finance applications due to its versatility, ease of use and extensive library support. McKinney (2017) and Sweigart (2019) highlight the usage of Python in financial analysis, data visualisation and automation, showcasing its efficacy in building the customizable budgeting solutions.

In this application, we could have asked the user for their budget and added the income, credit card(debt information) and visualisations of the additional data and also some functions that allow deleting or editing of the entries that we have submitted. However, this is a challenge as our knowledge, time and resources are limited. Despite the potential improvements, we managed to develop a proper functioning application that can help us to be more aware of our financial data.

**Methodology:**

In our second page of personal budget tracker, we used quantitative and qualitative methodologies to create a user friendly application. The quantitative portion was focused on the financial data backend processing, handling, and analysing data by using Python. Then, in order to create a functional and simple application for users, we have mainly focused on user interface design and accessibility testing for everyday budget tracking. For more code explanations, everything is written below appendices.

**Data Collection:**

For data collection, we added user inputs in categories that users have to enter such as shopping, bills, eating out, groceries, remaining budget, and total expenses. With python, we have used a dictionary to store this under “finance” which is the financial data for this application. This dictionary holds a list for amounts of each of the category expenses and then tracks the total budget and expenses. In order to return the expenses budget, we created a function that returns a list of dictionaries that represents each category with its description and respective amounts. Then to update the user's total expenses, we added an add expense function that adds new expenses to a specific category by appending the amount and description to the appropriate lists in the finance dictionary. This method provides realistic data by stimulating users’ interactions and financial management use cases. With HTML, we made a form that allows user inputs which has categories with four options that includes shopping, bills, eating out, and groceries. Then we have another one for users to enter the amount and another one for the description. We made elements such as select and input which allows user inputs and a method for submitting the form.

**Data overview**

We do not have any dataset. The data comes from the user input except the initial budget of 1000 GBP.

The variables are as follow:

* **category** : the category input from the user
* **category\_mapping** : the dictionary to connect the user input and the backend data
* **amount\_str (amount)** : the amount input from the user
* **description** : the description input from the user
* **finance** : the dictionary of the different expenses (include: Shopping, Bills, Eating out, Groceries, Remaining Budget and Total expenses)
* **expenses** : the list of data of the expenses with their description to generate in the data
* **initial\_budget** : the budget given (1000 GBP)

**Methods of Analysis:**

This application is built with Flask, which controls data management and analysis, that converts raw data into actionable insights for user request handling. It is a flexible framework that we used to create various forms of data bases. It allowed us to combine Python and HTML codes together, making our website manageable and functional. With Python, we have scripted the required logic necessary to handle, store, and calculate the financial data. In addition, we have used Matplotlib to generate and visualise our data in the form of graphs which are pie charts and bar graphs that will be presented below. Combining this with our HTML and CSS codes allowed us to build our website and style the user interface .

**Data Presentation :**

For the data presentation of our Personal-Budget-Tracker, we used HTML, combining it with CSS for our visual designs. This allows users to directly input their data into our form based interface where they submit their expenses which will later be calculated under “Your Financial Data”. This showcases the total expenses of users and remaining budget, along with a table of four categories we have implemented, with a description and amount for each. The submission of user’s data is then further displayed with graphical data, which are pie charts and bar graphs for your budget and for each category. The graphs are created to dynamically display user’s visual feedback on their spending pattern and financial situation. The purpose behind giving feedback to the user’s data with graphs is so that they can reflect on their spending habits and improve their financial data over time.

**Data Analysis**

**Key findings :**

The fundamental of our project is based on tracking and categorising user’s inputs in an efficient way. It provides users with accessibility and usability, data and visual feedback, and financial management. Because of our data categorization, users are able to identify their main areas of spendings which would help them with their spending habits and better decision making for the way they handle their finances. Our interactive elements and visual displays will also help increase user engagement since they would regularly want to check their spendings with our visual displays that provide immediate feedback. Our website also provides usability where it is easy for users to interact with our website due to how easy and straightforward our design structures are. These are the main key findings that make Personal Budget Tracker an effective website to track user’s financial data.

**Interpretations :**

Our website is a simple and easy platform for users, which accommodates a wider audience unlike other platforms for financial data tracking that might be too complex for users to understand. SInce our methods that have been implemented of how we display data, calculate, and provide immediate feedback makes it usable and accessible to all users despite their prior level of budgeting knowledge. Given our literature analysis, these findings comply with research that supports how improved user interface designs can improve user engagement while allowing users to achieve better financial management and financial awareness of their spendings.

The user engagement is also a good factor of our website due to our simplified system and designs. Due to this increase of user-interactions, this enables users to become more aware and mindful of their spending habits.

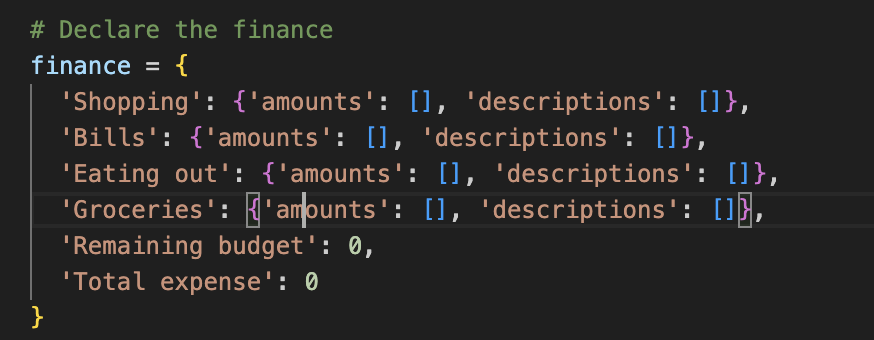
Our findings also show a correlation between better financial management and the use of interactive visualisations. Because of our graphical displays, users are able to see their spending trends much better whereas they might not from just reading their data sheets. In our website, pie charts and bar graphs are used to let users instantly recognize the categories that they spend the most money on, enabling them for better budgets adjustments they need to make.

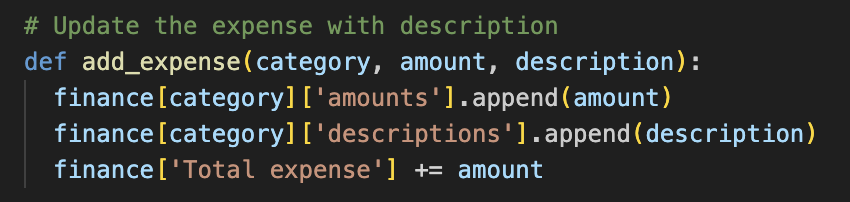
Therefore, how our website is set up differs from traditional budgeting tools since those focus on more static tables and are designed to be complex when analysing data, whereas our website is more efficient with data visualisations and accessibility.

**Implications:**

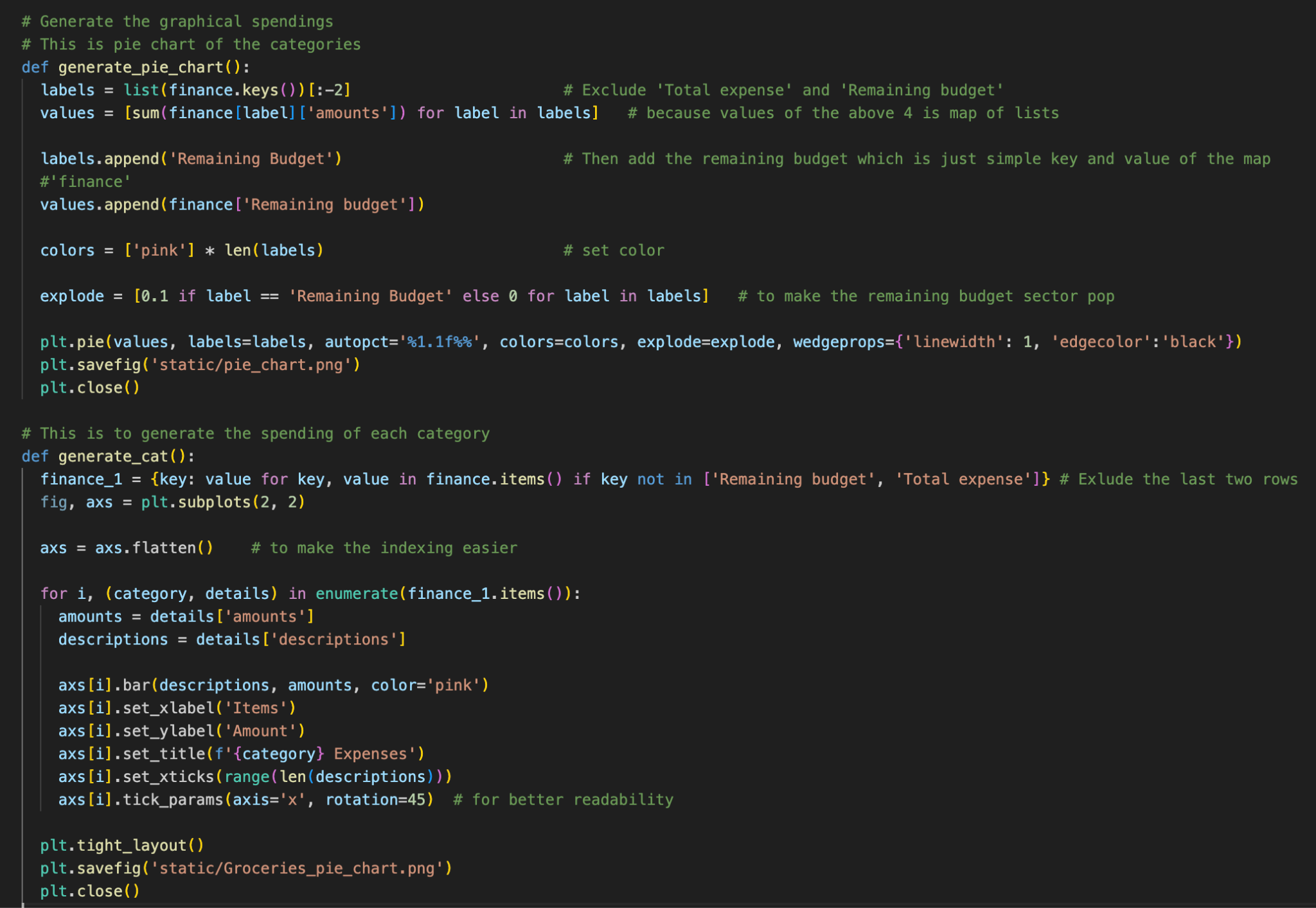
One of the primary challenges we encountered was ensuring data consistency and accuracy in our budget tracker application. Initially refreshing the page led to data loss, disrupting the user experience. To address this, we implemented a solution to persistently store the recorded data, ensuring continuity even after the page refreshes. In our Flask application, we utilised session store expense details and updating them with each user input, we ensured that the recorded data persists throughout the user session. We also had some complications regarding the data visuals. The graphical data did not display correctly, the labels of categories were overlapping. Here’s how we implemented corrections in our code :

We fixed the finance dictionary and add\_expense().





For graphs, we added the line “plt.close()” to fix the overlapping of the labels.



We added the line “plt.close()” to fix the overlapping of the labels.

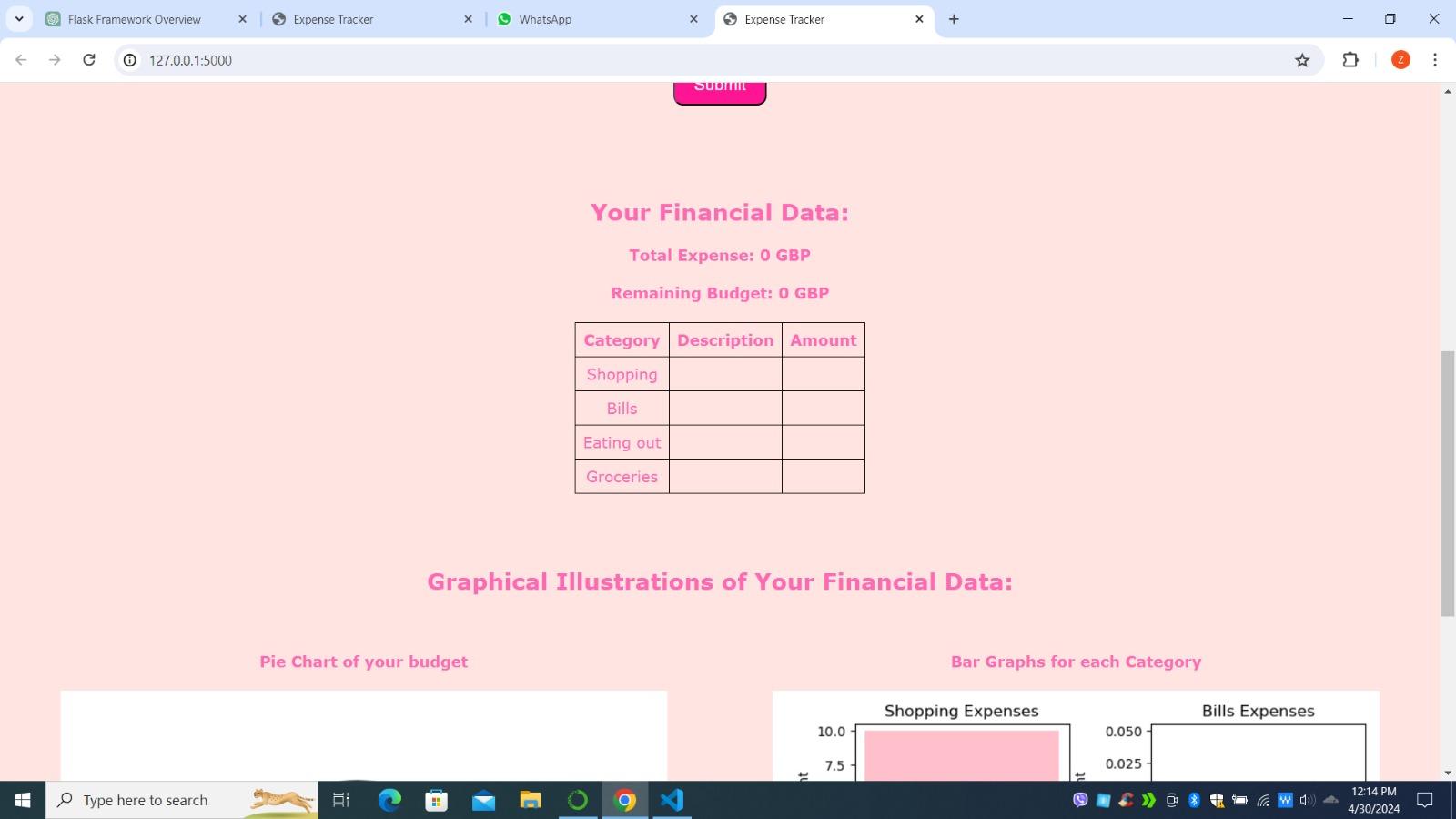
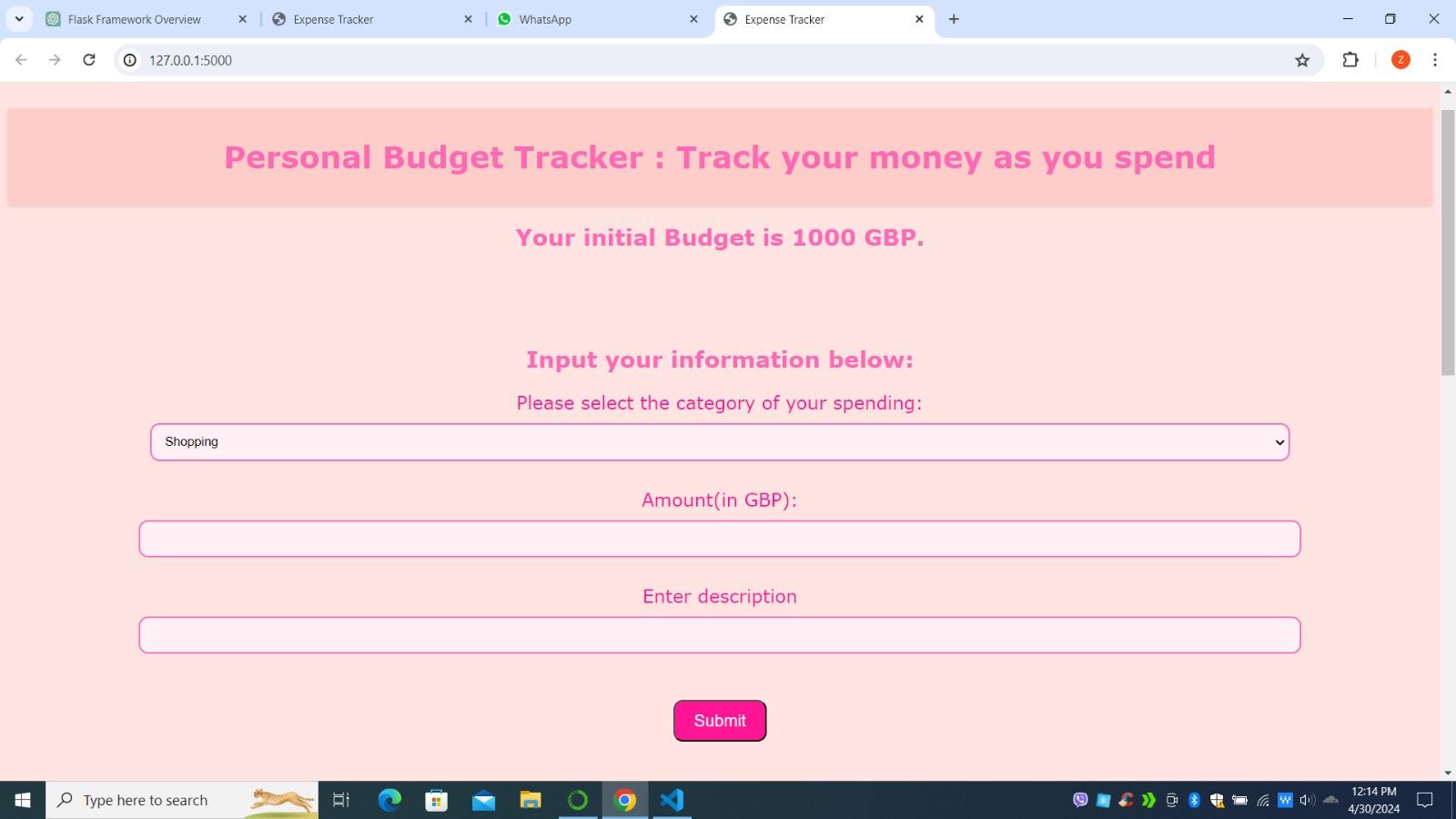
**Happy Path:**

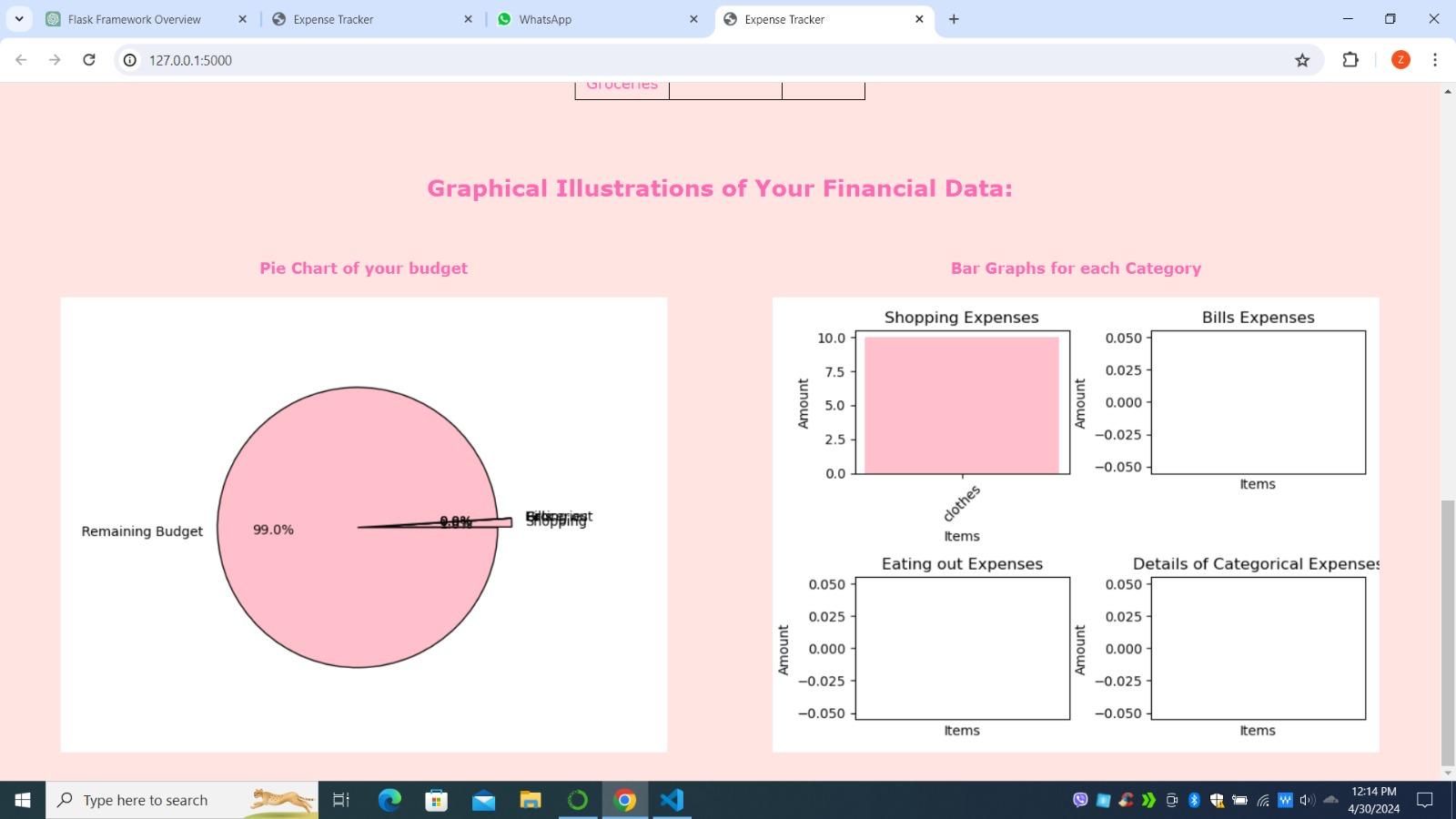
You run the python file in the terminal by entering “**python app.py**”

A link “<http://127.0.0.1:5000>” is generated.

You follow the link and You see a webpage as per the following images.

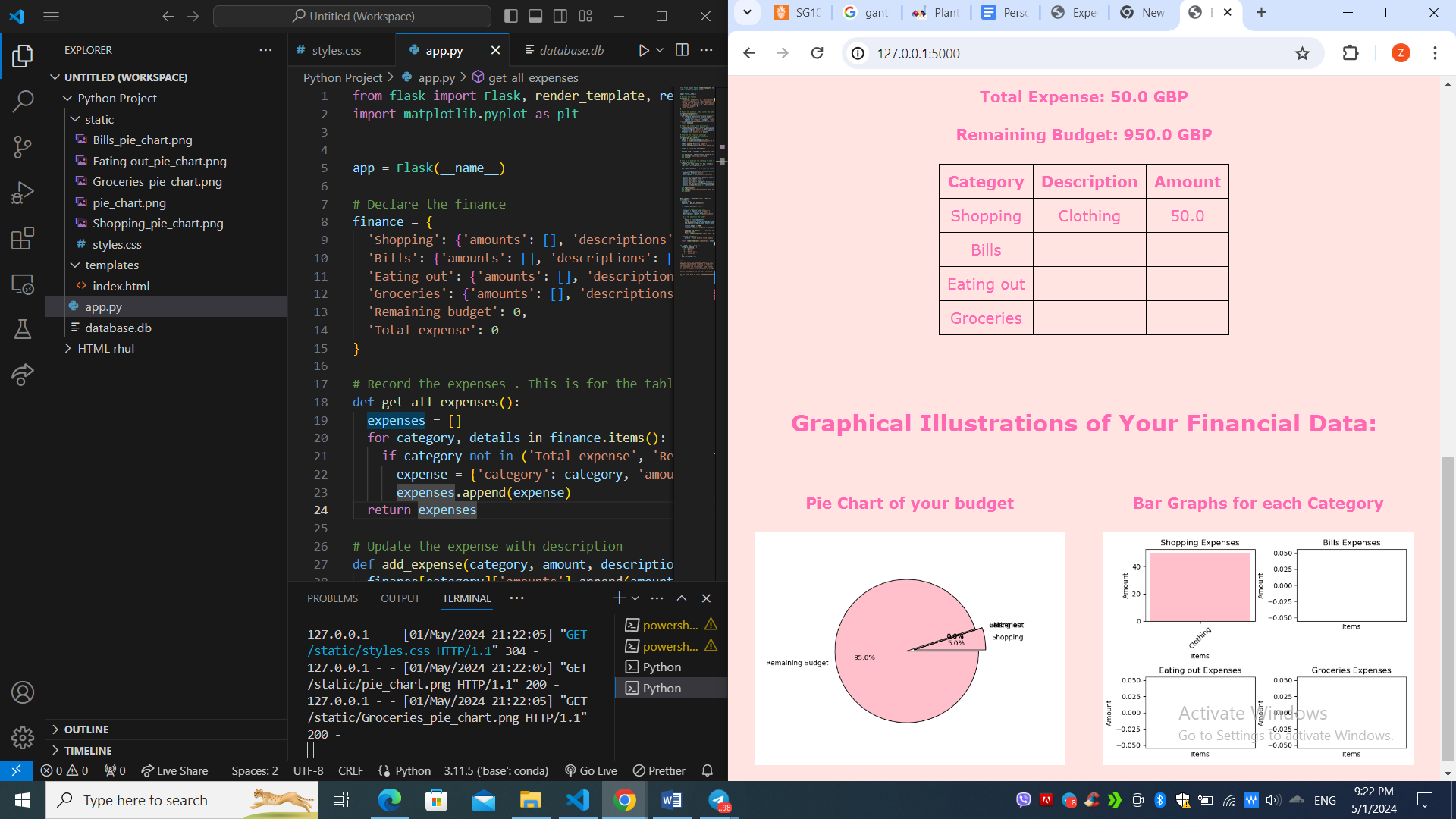
( not necessarily the same as the last picture )





You choose the “Shopping” category, add the amount in GBP “50”, enter the description “Clothing” and press the “Submit” button.

(Total expense, remaining budget, table and the graphs are updated)

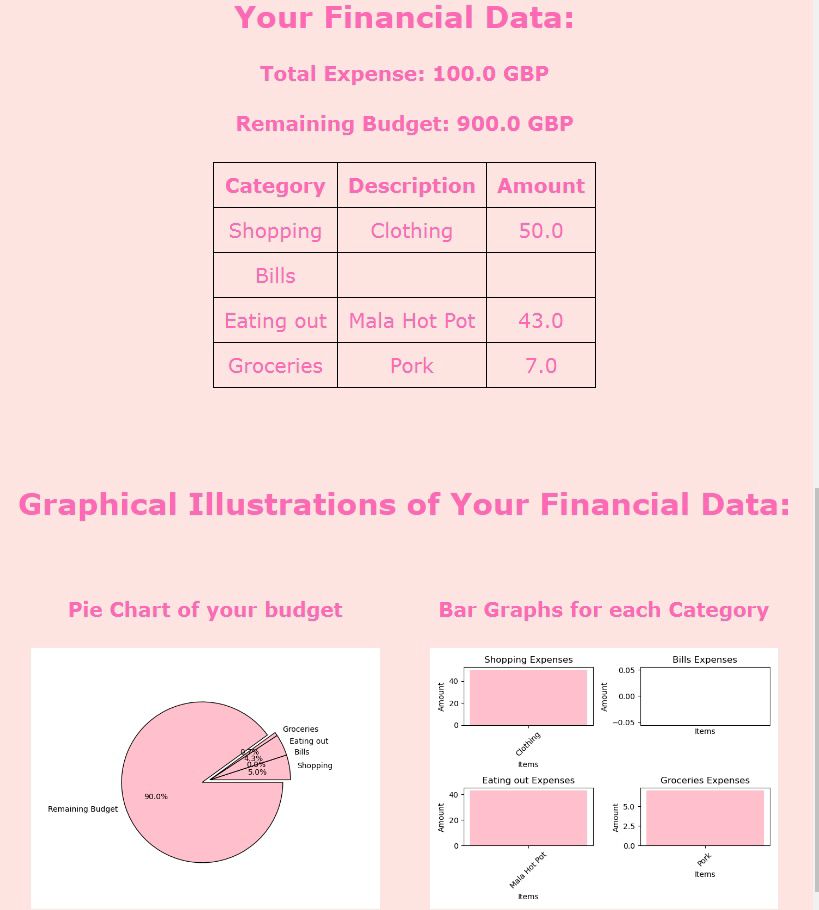


You enter more data:

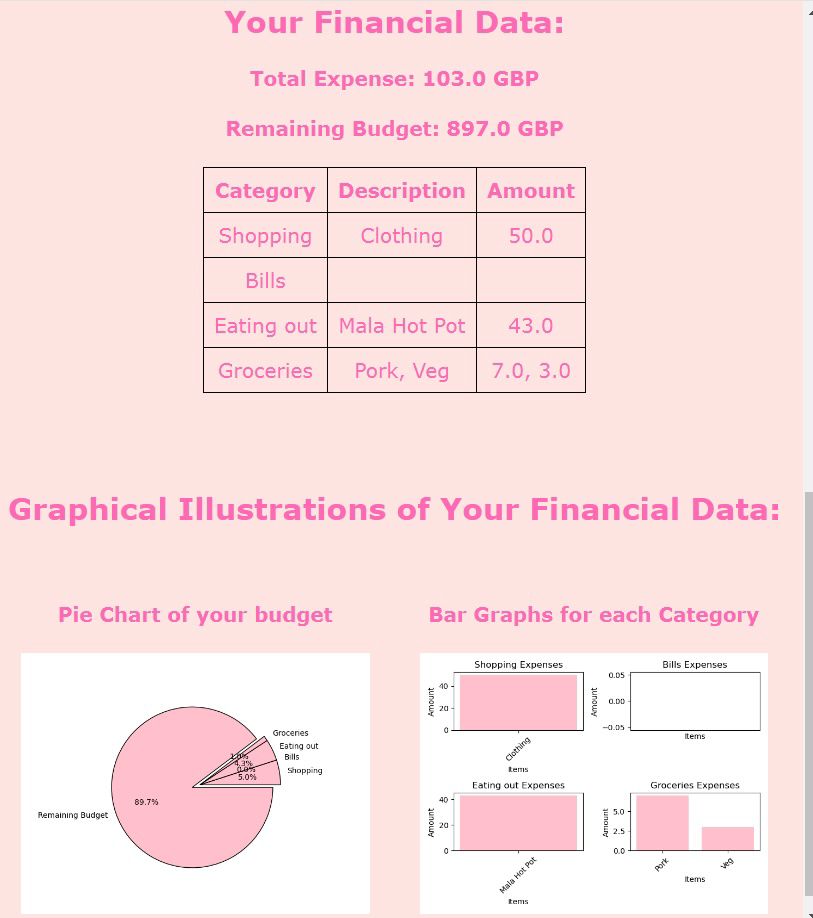
category “Eating out” , amount “43” , description “Mala Hot Pot”, enter “Submit”

category “Groceries”, amount “7”, description “Pork”, enter “Submit”

(The webpage updated)



Consequently, you choose the last category again “Groceries”, amount “3”, description “Veg”

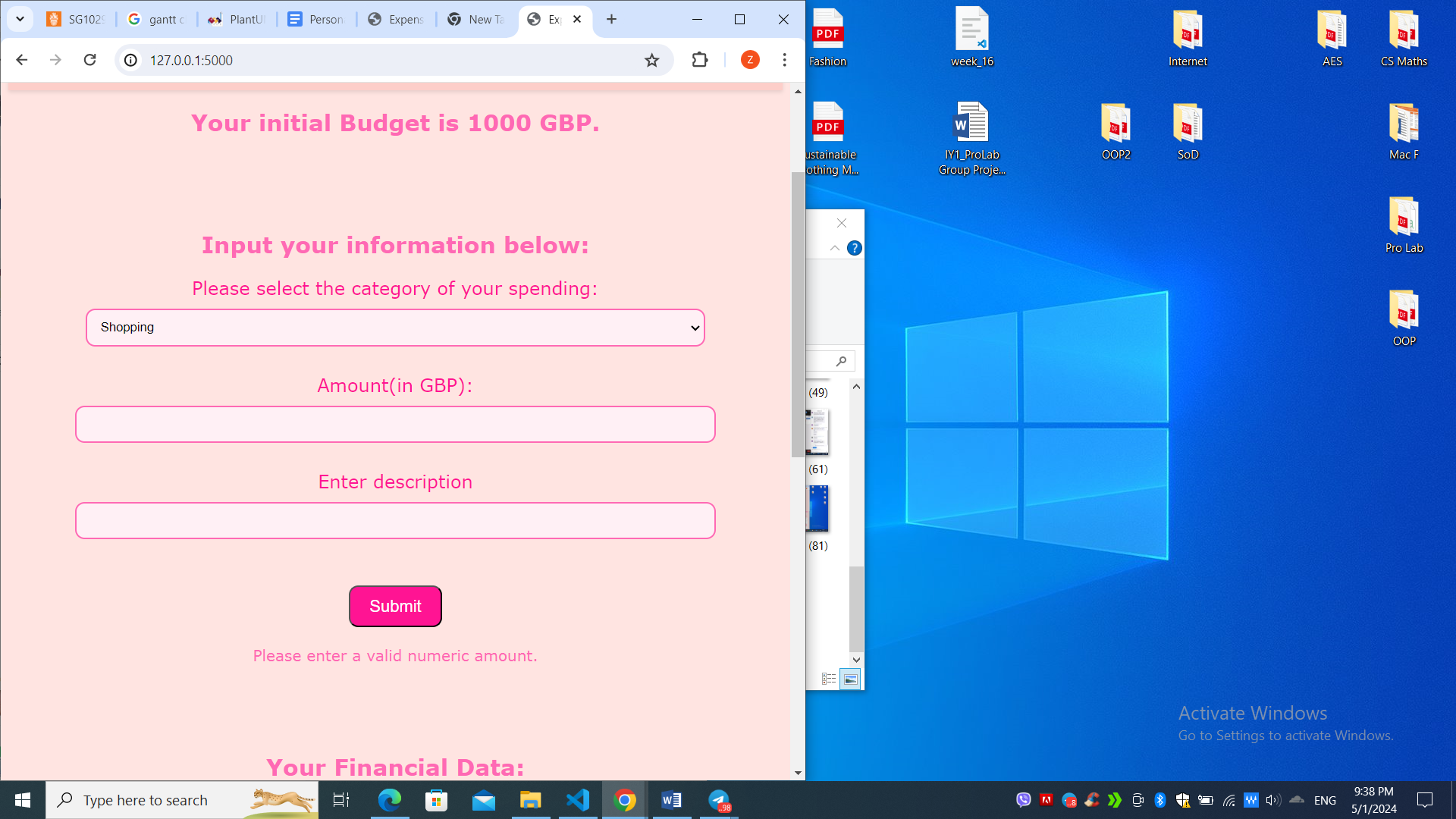


You close the webpage by pressing “Ctrl” and “C” in the terminal. You can make a new budget tracker webpage by running “python app.py” again.

Testing:

1. You enter the words in the amount, for example “thirty five” or “35 GBP” or “Clothing”.

Error raised and the data is rejected. You will be asked to enter the data again.



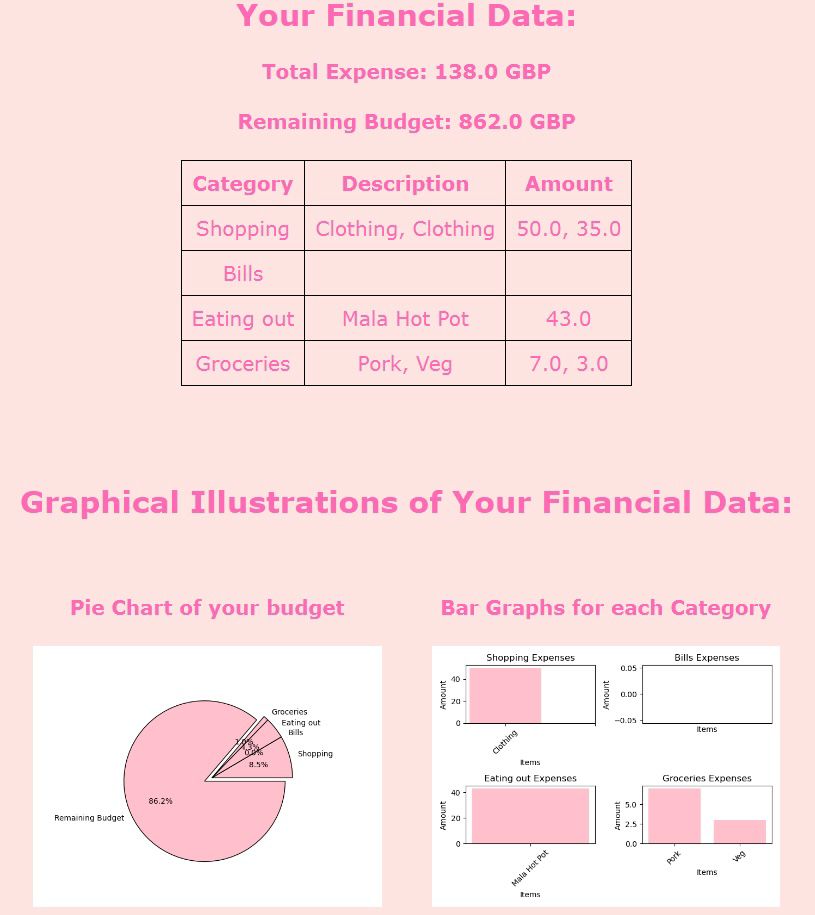
1. You enter the same description in the same category.

Eg. you submitted the spending of 'clothes' in the 'Shopping' twice like 50 and 35,

The bar graph only takes the first input. like clothes - 50

It doesn't combine into clothes-80 or changes into clothes-35

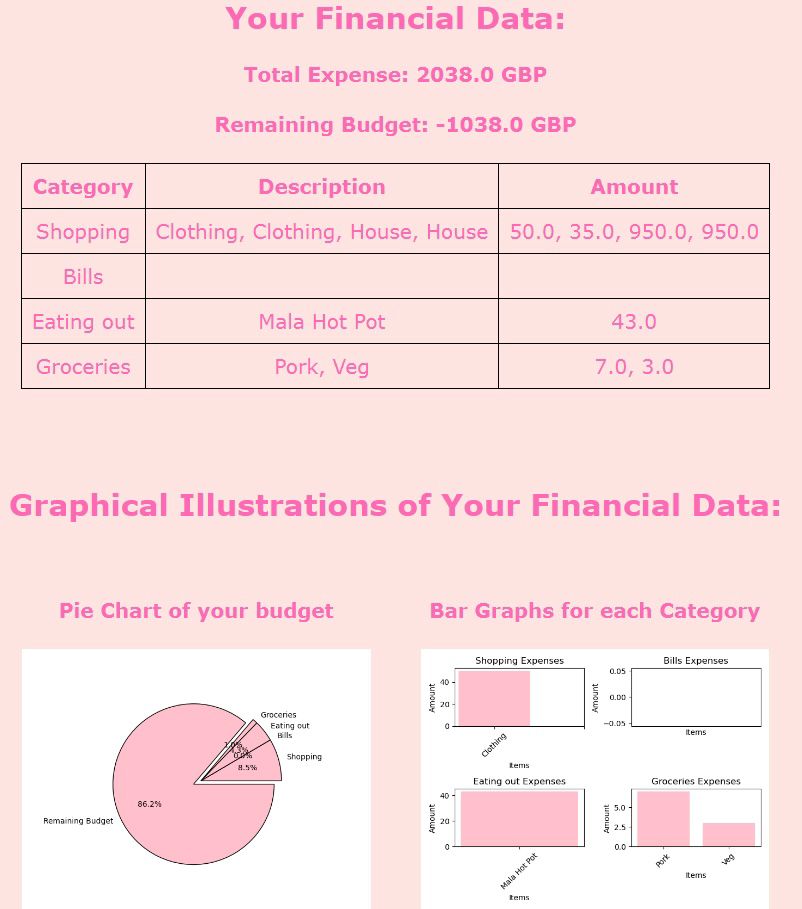
But it does update the other data correctly.



So you might have to input DIFFERENT DESCRIPTION each time you SUBMIT.

1. You enter the amount that exceeds the remaining budget.

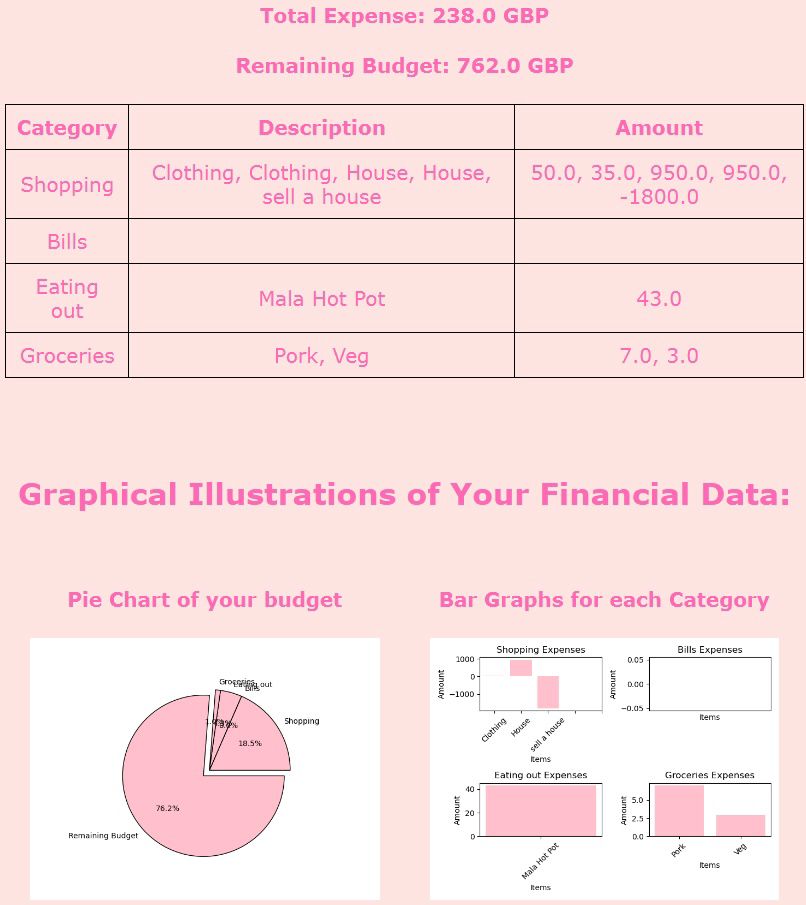
The graphs are not updated. Only total expense, remaining budget and the table are updated.



1. Consequently, you enter the negative amount. “

Shopping” category and “-1800” amount, “selling a house” description.

All data is updated properly.



Potential improvement :

1. In the current version, the selecting and input have to be done carefully. Update or delete functions can be added.
2. Could include an ‘income’ category to expand the functionality of the application.

## **Summary**

**Conclusion :**

**Research goal:** Our research goal was to create a resilient budget tracker application using Python, HTML and Flask. Through thorough planning and implementation, we have achieved our aim. Despite not having a pre-existing dataset, our application successfully managed user input data to track expenses across different categories. The application enables users to log their financial transactions, categorise them, and track their expenses through visual representations, fulfilling the fundamental objectives of our project.

**Planning:**

In the planning stage of our project, we carefully evaluated our goals and devised a strategy for implementation. Our primary objective was to develop a budget tracker application. To achieve this, we allocated specific time slots during spring break and term time, particularly on Wednesdays, for collaborative work and progress monitoring. Each team member took on responsibilities across all project components, ensuring equitable participation and contribution. We organised our approach by dividing the code into distinct features, with each team member handling coding, report writing, and presentation for assigned features. To elaborate in more details, we did the planning like task and time allocations in the first week. We started our codes in the second week and the coding continued to the next week. During that week, we also worked on css. During the final week, we wrote our reports while amending the codes as we saw fit. This collaborative framework facilitated efficient task management and fostered a comprehensive understanding of every aspect of the project. Additionally, we drew inspiration from a variety of resources, including online tutorials and GitHub repositories, to inform our methodology and implementation strategy.

**Integrating Python with HTML:**

Initially, we encountered difficulties integrating Python with HTML for our application. However, through thorough research, we discovered Flask, a micro web framework that facilitated seamless integration between these languages. Despite the initial challenge of navigating this new software, we dedicated time to ensure smooth communication between the backend and frontend components.

**Visual Representations:**

We further enhanced the usability and interpretability of our budget tracker application by integrating visual representations such as pie charts and bar graphs. These graphical depictions offer the user a clear understanding of their expense breakdown across different categories at a glance. In order to achieve this, we utilised Matplotlib within the Flask framework, ensuring seamless integration of these visual elements into our application.

**User Experience:**

We also prioritised creating a user-friendly interface and design for our application. Emphasising simplicity and accessibility, we ensured that users with varying levels of technical expertise could navigate through the application. We further committed to improving user experience by integrating a stylesheet into the HTML code. This not only enhanced the application’s aesthetics but also improved its usability. By adhering to design principles and thoughtfully integrating visual elements, the stylesheet significantly contributed to enhancing the overall user experience and improving application navigation.

**Project limitations and challenges:**

While we successfully achieved our research goal of creating a resilient budget tracker application using Python, HTML, and Flask, we encountered and overcame several challenges along the way. Since our application relies solely on user input rather than an existing dataset, ensuring data consistency and accuracy posed initial difficulties. However, we successfully overcame this difficulty, now the recorded data persists even if the page is refreshed. Additionally the remaining budget is updated accordingly. Through this approach, as long as the same link is used to access the budget tracker, the data remains recorded, ensuring a seamless user experience. Additionally, familiarising ourselves with the Flask framework in the initial stages of development presented a learning curve. Nonetheless, by dividing research tasks among team members, we navigated this challenge effectively. Acknowledging and addressing these limitations provide valuable insights for future projects.

### 

### **References:**

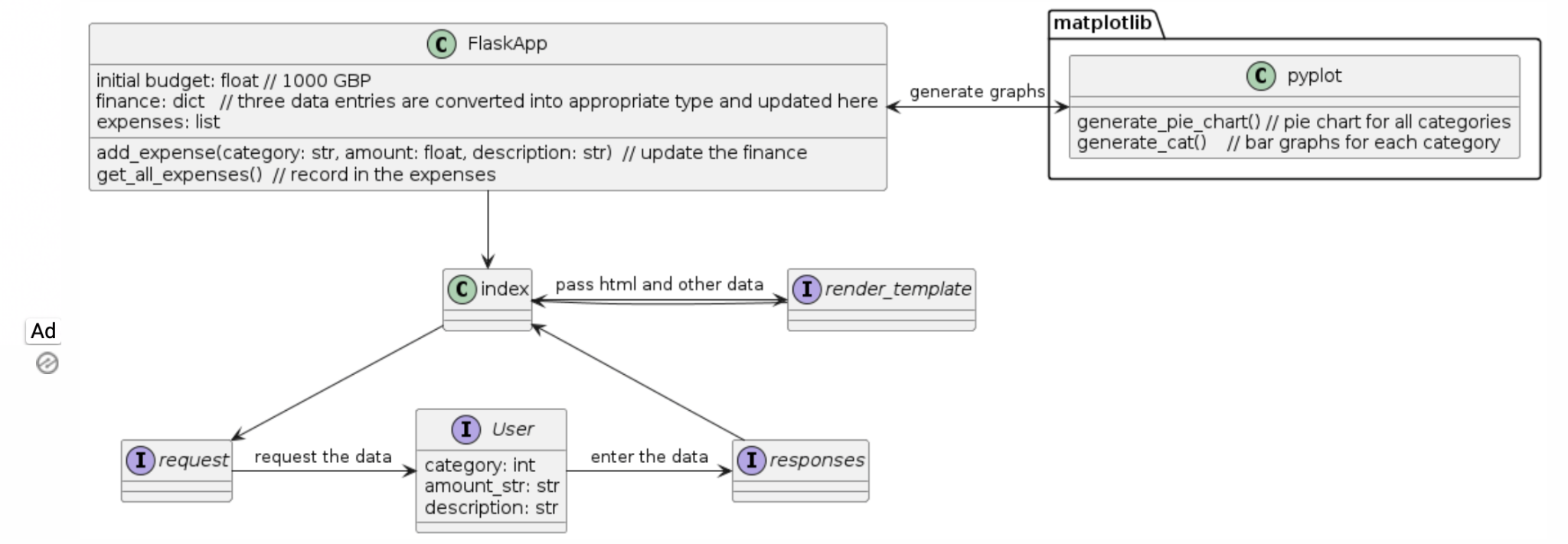
McKinney, W. (2017) *Python for Data Analysis, 2nd Edition*. O'Reilly Media, Inc.

Sweigart, A. (2015) *Automate the Boring Stuff with Python: Practical Programming for Total Beginners*. San Francisco: No Starch Press.

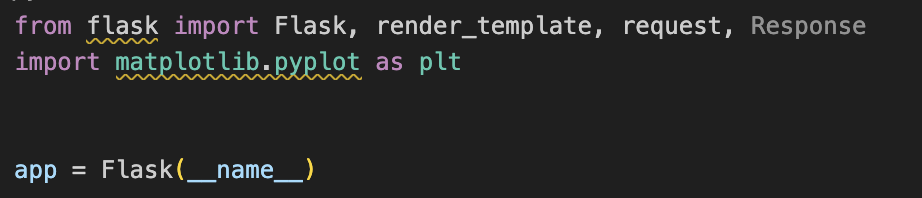
Zhu, A. (2023) 'Upgrading financial education by adding Python-based personalized financial projection: A randomized control trial', *British Journal of Educational Technology*, 55, pp. 731-750

## **Appendices :**

## **Appendix A: UML class diagram**



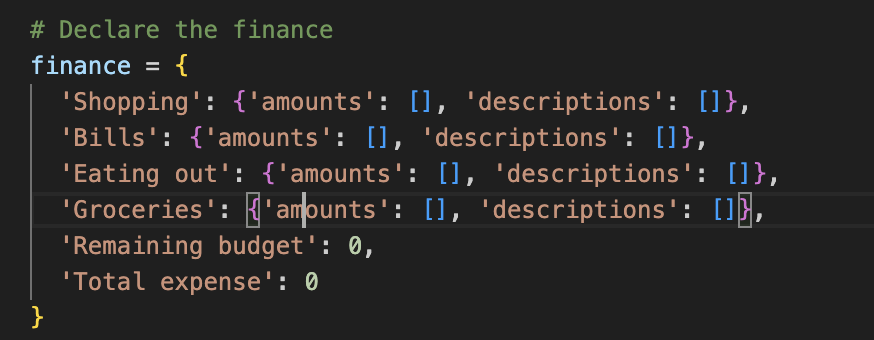
## **Appendix B: Code explanation**



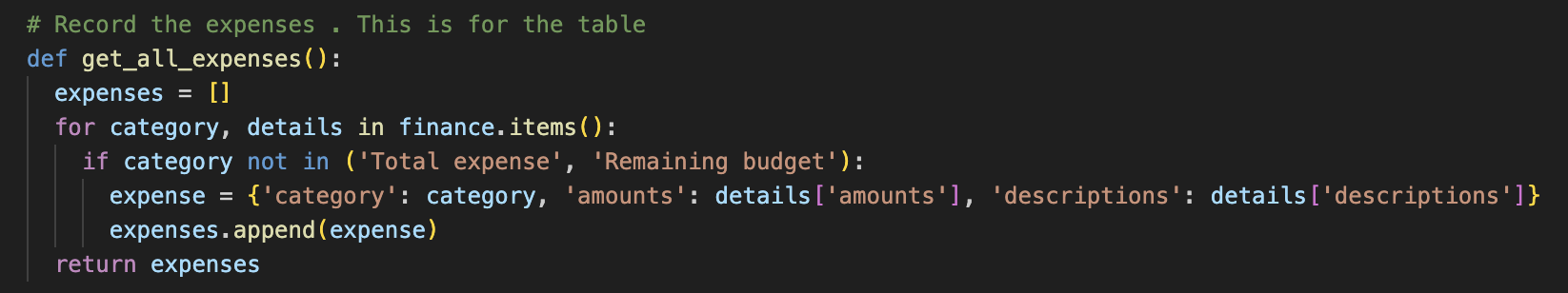
First, we imported the flask and matplotlib libraries and initialised the Flask app.

Then finance is declared as a dictionary.

For four key categories, the values are also assigned as a dictionary so we can record in more detail.



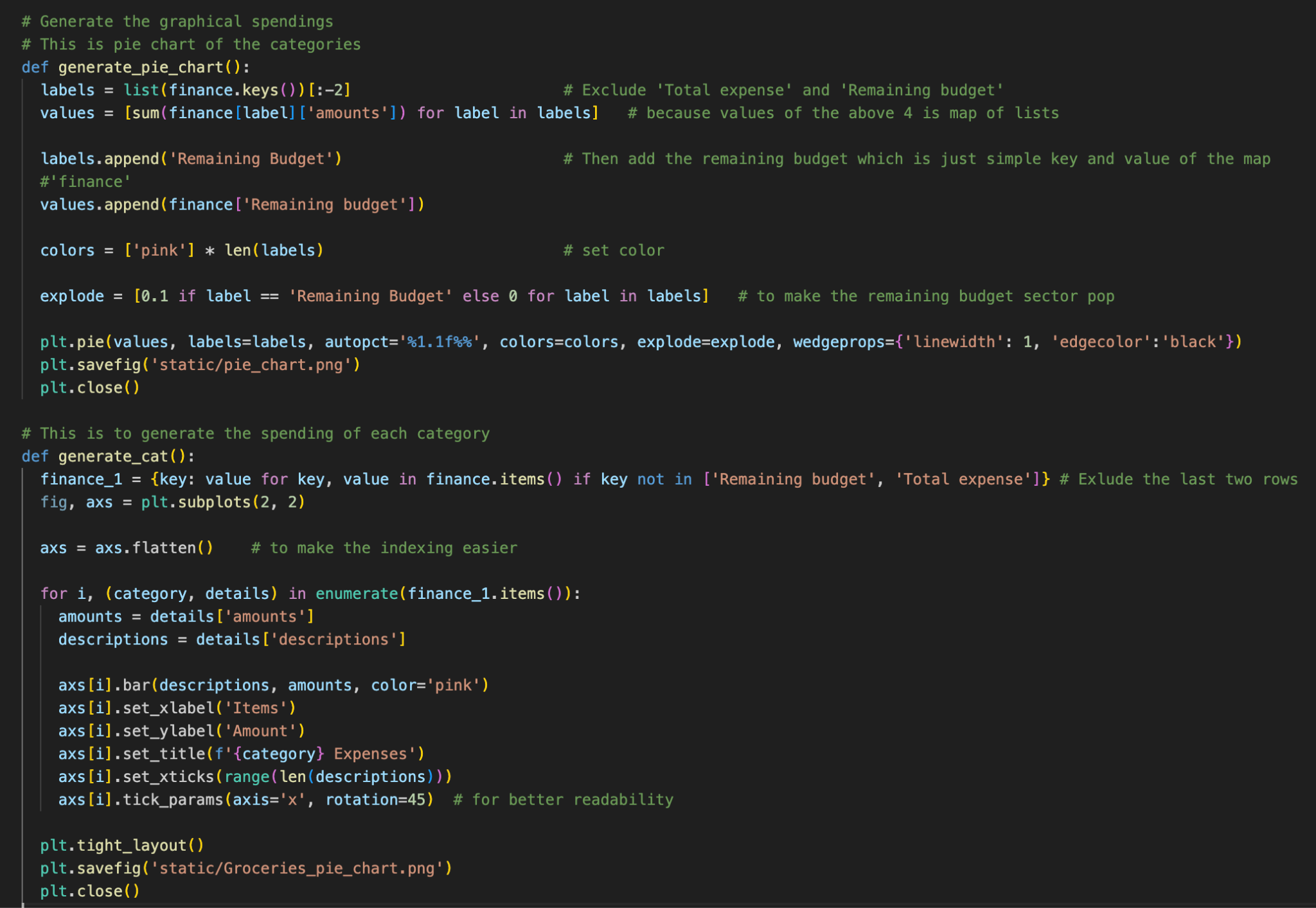
After that, we defined four functions. (see the comments in the codes for its function)



Here, since the ‘amounts’ and ‘description’ are lists, ‘append’ is used.

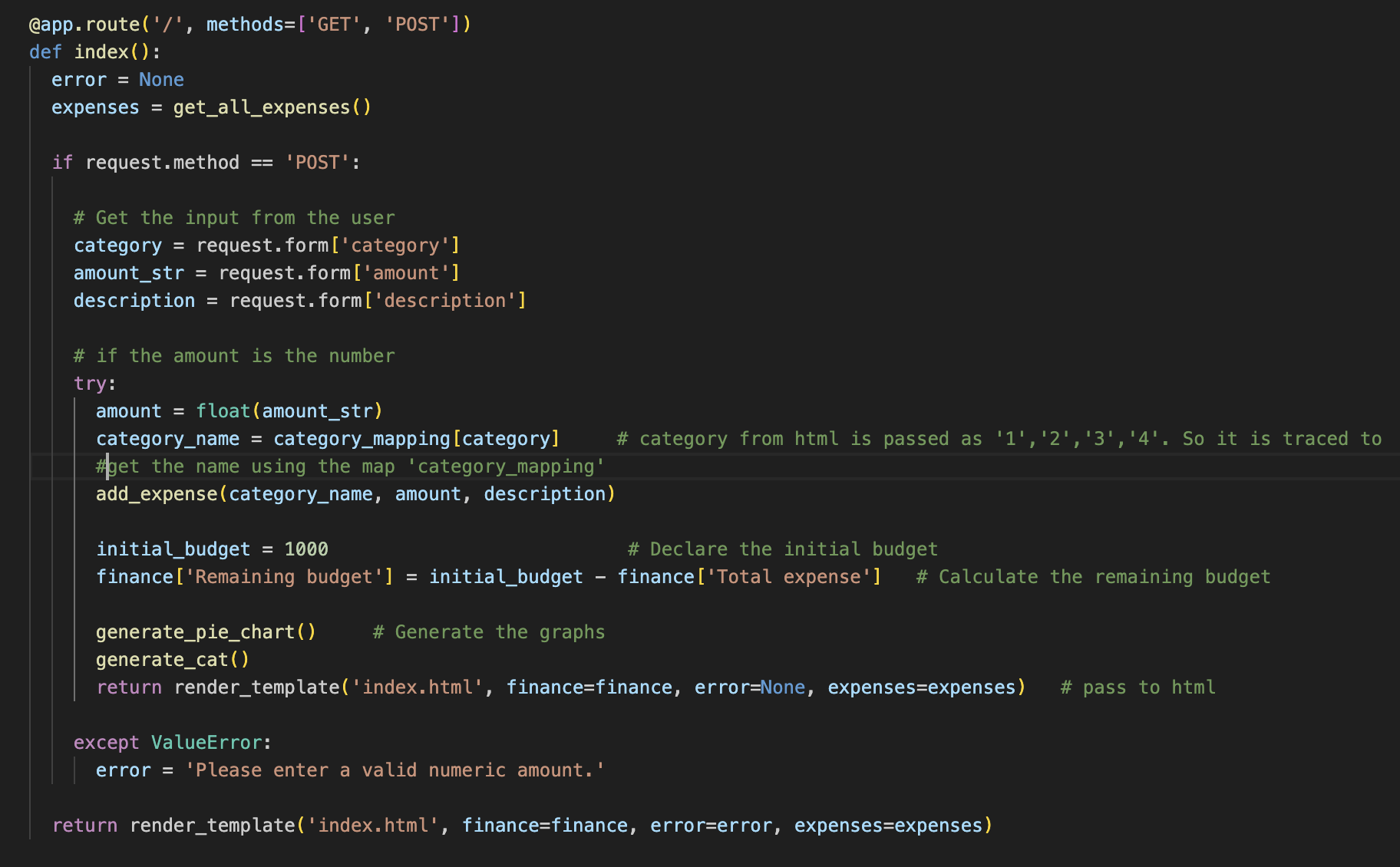


The last two functions are for generating the graphs.



In the index, we get the data entries from the request form and apply the right methods to update the finance and expenses as well as to generate the graphs. Try and except was also employed in case the user entered a string in the ‘amount’ section.

Then we return “render\_template” by passing html files and other variables that should be displayed in the html file.



In the main, we made the category\_mapping to link the category from the request form to the category in the index. (categories are passed 1, 2, 3, 4 from request form and then converted into respective categories using this dictionary ‘category\_mapping”).

And the app was run.

