EXPENSES TRACKER IN PYTHON

MINOR PROJECT REPORT

By

ANANYA PRADEEP WARRIER (RA2211029010021)
TINA KASHYAP (RA2211028010226)

Under the guidance of

Dr. M. SUNDARRAJAN

Assistant Professor. Department of Networking and Communications

In partial fulfilment for the Course

of

21CSC203P – ADVANCED PROGRAMMING PRACTICE

in **Department of Networking and Communications**

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC203P ADVANCED PROGRAMMING PRACTICE** entitled in " **EXPENSES TRACKER IN PYTHON**" is the bonafide work of **ANANYA PRADEEP WARRIER (RA2211029010021)** and **TINA KASHYAP (RA2211028010226)** who carried out the work under my supervision.

Dr. M. SUNDARRAJAN
ASSISTANT PROFESSOR

Department of Networking
and Communications,
SRM Institute of Science and
Technology
Kattankulathur

Dr. ANNAPURANI K
HEAD OF DEPARTMENT
PROFESSOR
Department of Networking
and Communications,
SRM Institute of Science and
Technology
Kattankulathur

ABSTRACT

The Simple Budget Calculator with Tkinter GUI is a user-friendly and efficient tool designed to empower individuals in managing their finances effectively. Built using Python's Tkinter library, this application provides an intuitive graphical interface for users to create, track, and visualize their budgets effortlessly. Users can input income, allocate funds to various expense categories, and set savings goals with ease. The interactive GUI allows dynamic adjustments, providing real-time updates on available funds and remaining budget. The application offers visual representations of budget allocations through charts and graphs, aiding users in understanding their financial priorities at a glance. With its simplicity and accessibility, this budget calculator is ideal for individuals, students, and small households looking to take control of their finances without the complexity of extensive financial tools. It promotes financial literacy and responsible spending while encouraging users to make informed financial decisions for a secure future.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.
We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.
We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to completemy course project We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systemsand Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our Course project Faculty **Dr.M .SUNDARRAJAN, Assistant Professor, Department of Networking and Communications ,**for his/her assistance, timely suggestion and guidance throughout the duration of this course project.
We extend my gratitude to our **Dr. Annapurani K., Professor and Head, Department of Networking and Communications** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

CHAPTER NO	CONTENTS	PAGE NO
1	**INTRODUCTION**	**10**
	1.1 Motivation	
	1.2 Objective	
	1.3 Problem Statement	
	1.4 Challenges	
2	**LITERATURE SURVEY**	**15**
3	**REQUIREMENT ANALYSIS**	**20**
4	**ARCHITECTURE & DESIGN**	**21**
5	**IMPLEMENTATION**	**23**
6	**EXPERIMENT RESULTS & ANALYSIS**	**24**
7	**CONCLUSION**	**40**
8	**REFERENCES**	**42**

1. INTRODUCTION

In this day and age where managing your personal finances is more important than ever, having access to the right tools can be the difference between success and failure. Enter your personal expense tracker, a powerful yet easy-to-use budgeting solution that will help you stay on top of your finances.

This expense tracker comes with two distinct interfaces. Whether you prefer the simplicity of a CLI or the beauty of a GUI, this tracker has it all. The CLI version is ideal for those who like a straightforward, no-nonsense experience. Run the program, enter your budget and let the app take you through the complexities of the 5:3:2 spending rule. Break your budget down into spending, saving, and additional funds and make smart financial decisions with ease.
In contrast, the GUI version provides an interactive and visually appealing experience when it comes to expense tracking. When you first open the application, you'll be presented with an attractive window that invites you to input your budget. With a few simple clicks, you can view your budget plan, display your spending budget and even enter specific information such as rent and bills.

The real-time graph function adds a dynamic element to your expense tracking. It gives you an immediate visual representation of your rental and bills, making it easier to understand where your money is headed.

No matter which interface you choose, the Personal Expenses Tracker will be your financial companion, providing you with insights, guidance and the freedom to customize your budget management to fit your specific needs. It's time to take charge of your finances and start your journey towards financial freedom and peace of mind.

1.1 Motivation

Are you prepared to take control of your financial life and make smart, data-based decisions about your finances?

Your personal expenses tracker, presented in a dynamic combination of CLI and GUI, is here to help you on your way to financial freedom. We know that managing your personal finances can be challenging at times, but this revolutionary tool aims to make it easier for you.

The CLI version offers a simple, text based interface that follows the 5: 3: 2 rule of spending. By making informed decisions, you can easily find the right balance between spending money, saving money, and having some extra cash in your wallet. Or, you can opt for the GUI version, which offers an eye-catching experience that combines functionality with visual appeal. Enter your budget and view your financial situation with a few simple clicks. Whether you're planning your budget, keeping track of your expenses, or looking into the details of your rentals and bills, The GUI version has everything you need to know about your finances. In addition, it even has a real time graph feature that helps you visualize your rentals and bills in real-time, adding an expenses tracker. It's not just about tracking your expenses, it's about achieving financial freedom, financial stability, and financial peace of mind. Understanding where your money goes allows you to make better decisions and plan for your financial future in a way that works for you.

Are you ready to take the first step toward financial control, financial freedom, and financial freedom? With your Personal Expenses Tracker, you'll have the tools and support you need to make better financial decisions, budget by budget.

Start your financial transformation today.

1.2 Objective

Key features and goals:

Budget calculation: Take user input for your total budget and allocate it using the 5::3:2 rule.
UI: Design a graphical user interface (GUI) with the help of tkinter library to make it easy to interact with your application.

Budget plan: Show users your budget plan, including your allocated spending money, saved money, and extra money.

Expense entry: Create a pop-up that allows users to enter their expenses like rent and bills.
Food and other expenses: Calculate your remaining funds based on your user input.
Real-Time expense visualization: Use matplotlib to generate graphs that show your expenses in real-time.

Clear financial insights: Make sure your users have a clear understanding of your financial situation so they can make better decisions.

Usability: Focus on an intuitive and easy-to-use design to make it easy for everyone, even if you have little to no technical experience.

Make it easy to quit the app when you're done budgeting or tracking expenses.
Use error handling to help guide users and avoid unexpected problems.
The Expenses Tracker app will help you manage your finances better, make better financial choices, and get a better understanding of your spending and saving habits. It's a great tool for anyone who wants to take charge of their finances.

1.3 Problem Statement

Our goal is to create a comprehensive Expenses Tracker App that provides the following features and goals:

Budget calculation: Design a system that lets you input your total budget, allocate funds according to the 5:3/2 rule, divide resources into spending, saving, and other expenses
UI: Create an easy-to-use graphical user interface (GUI) with tkinter, making it accessible and user-friendly for users of all levels of expertise

Budget planning: Let you view and plan your budget clearly, break it down into spending money, saving money, and surplus money, allowing you to choose between saving 20-30% of your budget to meet your financial goals

Expense entry: Create a simple pop-up to let you enter your basic monthly expenses, such as rent and bills
Use Matplotlib to create dynamic graphs that show your monthly expenses in real-time
Financial insights: Provide users with useful financial information.
Help users understand their spending habits, assess their budgeting behavior, and gain a better understanding of their financial situation.
Make the app easy to use and accessible for a wide range of users.
Easily exit the app.
Provide a smooth user experience.
Provide reliable error handling mechanisms to help guide users and avoid unexpected issues.

Expected Result:
The Expenses Tracker App for Budget Management is ready to provide a complete financial management solution that will empower users to better manage their finances.
It will help users track their spending, make better financial decisions, and improve their financial health. We are developing a command-line budget calculator as well as a feature-rich graphical user interface (GUI) app for expense tracking.

1.4 Challenges

The Expenses Tracker app has a good starting point, but like all software projects, it can face several issues in terms of function, user experience and development. Here are some of the issues that you may face while developing and using your Expenses Tracker.
User interface complexity: It can be difficult to create an easy-to-use and user-friendly interface. It is important to balance the need for feature-rich features with a clean, clutter-free design

Input validation: It is essential to validate and sanitize user inputs to avoid errors and security risks. Inaccurately or maliciously entering data can disrupt your application.
Cross-platform compatibility: It is important to ensure that your Expenses Tracker is compatible with different OSes and screen resolutions. It is also important to make sure that the app behaves consistently across platforms.
Real-time data visualization: Real-time data visualizations, especially dynamic graphs, can be difficult to implement in an efficient and optimized way.
Error handling: It is crucial to have robust error handling mechanisms in place to guide users in unexpected situations and avoid application crashes.
Protecting user data and resilience against security threats: Data breaches are constant concern.

Maintaining compliance and regulations: If the application deals with sensitive financial data, compliance with different regulations and security standards can be difficult.
User feedback and iteration: Collecting and integrating user feedback for continual improvement is essential.

Documentation: Developing easy-to-read and comprehensive documentation to assist users in understanding and navigating the application can be time-consuming.
Cooperating with a development team: Managing code changes, if needed, can be difficult without a proper version control system

2. LITERATURE SURVEY

2.1 Literature

1. What is Budgeting and Expense Management?

Budgeting and expense management refer to the process of planning, allocating, and controlling financial resources to achieve specific goals. It involves tracking income, managing expenses, and ensuring that spending aligns with financial objectives.

2. Importance of Budgeting and Expense Management:

Financial Stability: Effective budgeting ensures financial stability by avoiding overspending and accumulating debt.
Goal Achievement: It helps in achieving short-term and long-term financial goals, such as saving for a house or retirement.
Debt Reduction: Proper budgeting can help in reducing and managing debts effectively.
Emergency Preparedness: Budgeting enables building an emergency fund for unexpected expenses.
3. Common Budgeting Methods:

Zero-Based Budgeting: Every dollar of income is allocated to expenses, savings, or debt payments, leaving no money unassigned.
Envelope System: Cash is divided into envelopes for different expenses, ensuring that once the envelope is empty, no more money can be spent in that category.
50/30/20 Rule: Allocating 50% of income to needs, 30% to wants, and 20% to savings and debt payments.

4. Technology and Budgeting:

Budgeting Apps: Various mobile apps help individuals track expenses, set financial goals, and provide insights into spending patterns.
Online Tools: Web-based budgeting tools offer features like automatic transaction categorization, real-time tracking, and financial reports.
5. Challenges in Budgeting and Expense Management:

Lack of Financial Literacy: Many individuals lack knowledge about effective budgeting techniques and financial planning.
Unexpected Expenses: Sudden medical emergencies or car repairs can disrupt budgets, leading to financial stress.
Changing Income Levels: Irregular incomes, common among freelancers and self-employed individuals, pose challenges in consistent budgeting.
6. Future Trends in Budgeting:

AI-Powered Financial Advisors: Artificial Intelligence-based financial advisors analyze spending patterns and provide personalized budgeting recommendations.

Cryptocurrency Budgeting: With the rise of digital currencies, new budgeting methods specific to cryptocurrencies are emerging.

2.2 Methodology

The development of the real-time expenses tracker involved a systematic approach that combined both Python programming and Tkinter GUI toolkit. The methodology for creating this application can be outlined as follows:

1. Problem Definition and Requirement Analysis:
The initial step involved understanding the problem at hand, which was to create an interactive expenses tracker allowing users to input their budget, view budget plans, and monitor real-time.

2. Choosing the Programming Language and Libraries:
Python was chosen as the programming language due to its simplicity, versatility, and extensive support for various libraries. Matplotlib, a popular data visualization library, was selected to create real-time graphs for visualizing expenses.

3. Designing the User Interface (UI):
The Tkinter library, a standard Python interface to the Tk GUI toolkit, was employed for designing the graphical user interface. The UI was conceptualized to be user-friendly, featuring input fields for budget, rent, bills, and buttons for various actions such as calculating budget, viewing budget plans, and inputting rent and bills.

4. Implementation of Budget Calculation Logic:
The core logic of the expenses tracker was implemented in Python. Classes and functions were created to handle budget calculations, including budget planning, spending allocation, and real-time expense visualization. Input validation and error handling mechanisms were integrated to enhance user experience and ensure data integrity.

5. Integration of Real-time Graphical Representation:
Matplotlib was utilized to generate real-time graphical representations of expenses. The 'drawRealTimeGraph' function was designed to update and display the expenses graph dynamically based on user inputs for rent and bills. Bar charts were chosen as the visualization format for clear and intuitive representation.

6. User Testing and Iterative Development:
The application was subjected to rigorous user testing to identify potential bugs, usability issues, and areas of improvement. Feedback from users was valuable in making iterative enhancements to the UI, functionality, and overall user experience.

7. Documentation and Code Refinement:
Comprehensive documentation was prepared to explain the functionality of different modules, classes, and functions. Code comments and explanations were added to ensure readability and maintainability. The code was refined, adhering to best practices and coding standards.

8. Deployment and User Training:
The final step involved deploying the application for users. User training materials, including tutorials and guides, were developed to assist users in navigating the application's features effectively.

2.3 Outcome

The Expenses Tracker project resulted in the creation of an intuitive and interactive financial management tool. This application provides users with a seamless experience to monitor their budget, allocate spending, and visualize expenses in real-time. The project achieved several key outcomes:

1. User-Friendly Interface:
The Expenses Tracker features a user-friendly interface developed using Tkinter, allowing users to input their budget, view budget plans, and track expenses effortlessly. The intuitive design enhances user engagement and simplifies the financial management process.

2. Budget Planning and Spending Allocation:
The application implements the 5:3:2 rule, automatically calculating spending allocations based on the user's budget. Users can visualize their spending money, savings, and remaining balance, empowering them to make informed financial decisions.

3. Real-Time Expense Visualization:
One of the significant achievements of the project is the implementation of real-time graphical representation using Matplotlib. Users can input their rent and monthly bills, and the application generates a dynamic bar chart, providing a clear visual representation of expenses. This feature enhances users' understanding of their spending patterns.

4. Error Handling and Validation:
The application includes robust error handling and input validation mechanisms, ensuring that users provide accurate and valid data. Error messages guide users in correcting input errors,

enhancing the overall user experience.

5. Interactive User Engagement:

The project's outcome fosters interactive user engagement by allowing users to dynamically explore their budget plans and expenses. The application encourages users to actively manage their finances and make adjustments based on their needs and priorities.

6. Empowering Financial Decision-Making:

By providing users with a comprehensive overview of their budget, spending, and savings, the Expenses Tracker empowers users to make informed financial decisions. It promotes financial literacy and encourages responsible financial management practices.

3. REQUIREMENTS

3.1 Requirement Analysis

Budget calculation and management: Users can enter their total budget, the default spending amount is 50% of the budget, the budget can be updated based on income changes, users can select a savings plan, the budget plan is calculated, the extra funds are available for spending, the remaining budget after savings is calculated, users can edit or delete the expense entries, real-time graphs focus on specific categories such as rent and bills, data validation: Input fields must validate the data to ensure it is in the correct format, data persistence: Users should save their data and expense history for future reference, data analysis: What are the non-functional requirements?

UI: The user interface should be intuitive, easy to use, and aesthetically pleasing

Cross-platform compatibility: The application should work on desktop, mobile, and other operating systems

Security: User financial data should be encrypted and authenticated, and passwords should be securely stored (hashed or salted)

Cloud integration: The application should use cloud storage to synchronize data and make it accessible from multiple devices

Data in transit: Ensure data is encrypted

Financial compliance: Ensure that the application complies with financial regulation (if applicable), data protection standards (GDPR, etc.), and other relevant regulations.

User Documentation: Provide comprehensive user documentation, and help resources , Clear instructions for all functions and features

Feedback & Updates: Include a feedback mechanism so that users can report issues or suggestions for improvements

Regular updates: Plan and implement updates to improve functionality and address reported issues

By taking care of both the functional and the non-functional needs, you can create a powerful Expenses Tracker app that gives users all the tools they need to manage their budgets, track expenses, and make financial decisions.

4. ARCHITECTURE AND DESIGN

4.1 Introduction

Budget Management Application is designed to assist users in managing their finances effectively. The application is divided into two components: the **CLI (Command-Line Interface) Budget Calculator** and the **Tkinter GUI (Graphical User Interface)**. The CLI Calculator focuses on providing a text-based budget calculation and planning tool, while the Tkinter GUI offers a more interactive and visually appealing interface for users.

4.2 User Interface (UI)

The application is built using the Tkinter library for Python, which provides the graphical user interface.
The UI consists of windows, frames, labels, buttons, and entry fields for user interaction.

4.3 Business Logic

The application's business logic is implemented in the **Budget** class in the command-line version and the **MainWindow** class in the GUI version.
The business logic handles budget calculations, input validation, and user interaction.

4.4 Data Storage

In both versions of the application, lists (expenses_list and labels_list) are used to store expense data temporarily. These lists are used for plotting the real-time graph

4.5 Plotting

Matplotlib is used to create and display real-time expense graphs.
The drawRealTimeGraph function handles the creation and display of these graphs.

4.6 Communication

The command-line version communicates with the user through the console, taking user input and displaying output.
The GUI version communicates with the user through the graphical interface, updating the UI elements with calculated data and displaying graphs.

4.7 Main Loop

In the command-line version, the application follows a linear flow from budget input to various options.
In the GUI version, the tkinter library provides an event-driven model where user actions trigger events (e.g., button clicks), which are handled by event handlers.

4.8 User Input

User input is collected for the budget amount, rent, bills, and other options.
Input validation is used to ensure that the user enters valid numeric values where required.

4.9 Budget Calculator

The budget is calculated based on the user's input, and various expenses are derived according to the 5:3:2 rule.

4.10 UI Updates

In the GUI version, the UI is updated to display the calculated budget, spending, savings, and expenses.

5. IMPLEMENTATION

The budget calculator application's application consists of a command-line interface (CLI) version and a graphical user interface (GUI) version built using Python. In the CLI version the application asks the user for their budget and calculates the cost based on 5:3:2 rule Users can choose to view details of their budget or expenses, including rent, bills and food. The CLI version uses a simple text interface and runs sequentially, collecting user input and displaying results in the console.

The GUI rendering created using the Tkinter library provides an intuitive and interactive experience. Users enter their budget through the entry field and can click buttons to calculate the budget, view the budget, and enter rent and fees. The GUI dynamically updates labels and text boxes to display financial information and expenses. A pop-up window allows users to input rent and charges, and a real-time graph feature visualizes rent and charges.

Both versions share common backend logic for calculating and displaying cost information. The GUI definition enhances the user experience by providing visually appealing interfaces, making budgets more attractive and user-friendly Overall, the implementation demonstrates Python's flexibility in command-line and developing graphical applications to meet various user preferences and needs

Code for CLI:

```
import os
import sys
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

expenses_list = []
labels_list = []

class Budget(object):
    def __init__(self):
        os.system('cls')
        self.budget = float(input('How much is your budget?\n'))
        self.spending = self.budget * 0.5
        os.system('cls')
        self.main()

    def main(self):
        print('This calculator uses 5:3:2 rule of spending by default.')
        print('\nYour budget: $', '{:.2f}'.format(self.budget))
        main_option = int(input('\nWhat do you want to do?\n1) View Budget Plan\n2) View
Spending Budget\n3) Exit\n'))
        if main_option == 1:
            self.budget_plan()
        elif main_option == 2:
            self.spending_budget()
        else:
            quit

    def budget_plan(self):
        os.system('cls')
        option = int(input('How much do you want to save?\n1) 20%\n2) 30%\n'))
        if option == 1:
            self.saving = 0.2
```

```
        elif option == 2:
            self.saving = 0.3
        else:
            print('Please select only 1 or 2')
        self.final_saving = self.budget * self.saving
        self.extra = self.budget-self.spending-(self.budget*self.saving)
        print('\nSpending: $', '{:.2f}'.format(self.spending), '\nTo Save: $',
'{:.2f}'.format(self.final_saving), '\nExtra: $', '{:.2f}'.format(self.extra))
        os.system('pause')
        os.system('cls')
        self.main()

    def spending_budget(self):
        os.system('cls')
        print('Spending Budget: $', '{:.2f}'.format(self.spending))
        rent = float(input('\nHow much is your rent/mortgage?\n'))
        bills = float(input('\nHow much are your monthly bills?\n'))
        food = self.spending - rent - bills
        print('\nEXPENSES:\nRent: $','{:.2f}'.format(rent),'\nBills: $','{:.2f}'.format(bills),'\nFood:
$','{:.2f}'.format(food))
        os.system('pause')
        os.system('cls')
        self.main()

if __name__ == '__main__':
    sys.exit(Budget())
```

Explanation:

1. Import Statements:

os: This module provides a way to use operating system dependent functionality.

Certainly! The os module in Python provides a way to interact with the operating system. It allows you to perform various operating system-related tasks, such as working with files and directories, managing processes, and accessing environment variables. Here are some key functionalities provided by the os module:

1. File and Directory Operations
2. Path Manipulation
3. Process Management.

sys: This module provides access to some variables used or maintained by the Python interpreter.

The sys module in Python provides access to some variables used or maintained by the Python interpreter and to functions that interact strongly with the interpreter. Here are some key functionalities provided by the sys module:

1. Command Line Arguments.
2. System-specific Parameters and Functions:
3. Functionality Related to Python Path:
4. Memory Management and Garbage Collection

matplotlib.pyplot: This module provides a MATLAB-like interface for creating plots and visualizations.

Certainly! matplotlib is a popular Python library for creating static, interactive, and animated visualizations in Python. It provides an object-oriented API for embedding plots into applications that use general-purpose GUI toolkits, such as Tkinter, wxPython, Qt, or GTK. Here's an explanation of the key components and functionalities of the matplotlib library:

1. Figure and Axes
2. Types of Plots
3. Customization and Styling
4. Subplots
5. Saving and Showing Plots
6. Interactive Features:

FuncAnimation: A class from the matplotlib.animation module that facilitates creating animations.

FuncAnimation is a class in the matplotlib.animation module that allows you to create animated visualizations in Python using the matplotlib library. It is particularly useful for generating dynamic and interactive plots, such as real-time data visualization, simulations, and scientific animations. Here's an explanation of how FuncAnimation works and its key components:

1. Initialization
2. Update Function
3. Animation Frames
4. Rendering

2. Global Variables:

expenses_list and labels_list are empty lists defined at the global scope.

3. Class Definition (Budget):

Defines a class named Budget.

4. Constructor:

The constructor method initializes the Budget object. It clears the console screen, takes user input for the budget, calculates the spending based on the 5:3:2 rule, and calls the main() method.

5. main() Method:

Displays options to the user: viewing budget plan, viewing spending budget, or exiting.
Based on user input, it calls budget_plan() or spending_budget() methods, or exits the program.

6. budget_plan() Method:

Asks the user how much they want to save (20% or 30% of the budget).
Calculates the final saving amount, extra amount, and displays the spending, saving, and extra

money.

7. spending_budget() Method:

Asks the user for rent/mortgage and monthly bills.
Calculates the remaining budget for food after deducting rent and bills, and displays the expenses breakdown.

8. Conditional Statements and User Input:

The code contains various conditional statements (if, elif, else) to handle user input and make decisions based on the input.

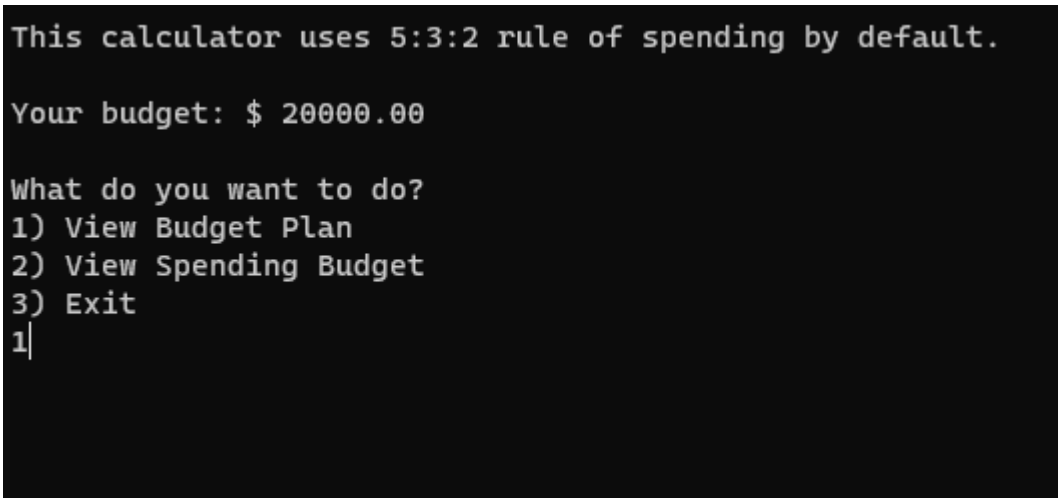
9. Console Clearing and Pausing:

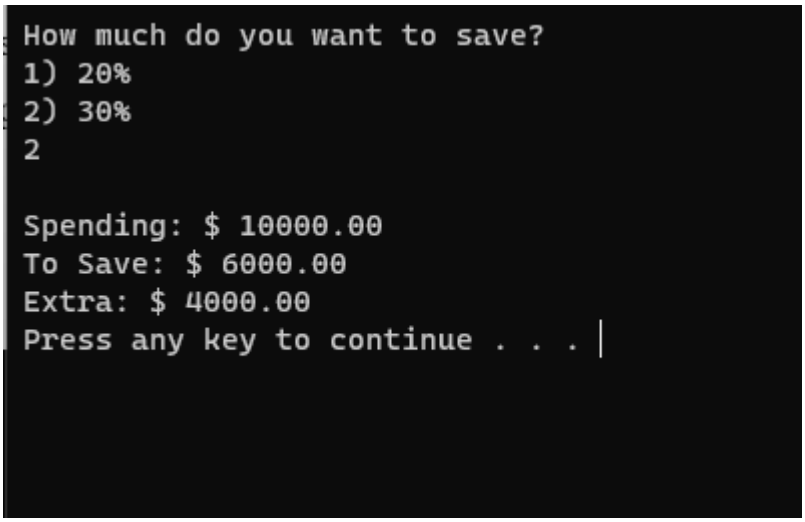
os.system('cls'): Clears the console screen (works on Windows).
os.system('pause'): Pauses the program and waits for the user to press a key (works on Windows).

10. Program Execution:

If the script is run directly, it creates an instance of the Budget class and starts the budget planning process. The sys.exit() function is used to exit the program gracefully.
Please note that the code is interactive and relies on user input for its functionality. Users can input their budget, make choices, and view budget plans and spending budgets based on their inputs. The code also includes console clearing commands to enhance the user experience.

Output:


This calculator uses 5:3:2 rule of spending by default.
Your budget: $ 20000.00
What do you want to do?
1) View Budget Plan
2) View Spending Budget
3) Exit
1


How much do you want to save?
1) 20%
2) 30%
2
Spending: $ 10000.00
To Save: $ 6000.00
Extra: $ 4000.00
Press any key to continue . . .

Code for GUI:

```
import tkinter as tk
import matplotlib.pyplot as plt

expenses_list = []
labels_list = []


class MainWindow:
    def __init__(self):
        self.rent = 0  # Initialize rent attribute
        self.bills = 0  # Initialize bills attribute
        self.root = tk.Tk() # Initiate new tkinter object
        self.root.geometry("920x540") # Set dimension size
        self.root.resizable(width=False, height=False) # Lock app dimension
        self.root.title("Budget Tool") # Window title
        self.style = ('Monospace', 18)
        self.draw('Welcome to the Budget App!\n\nThis calculator uses the 5:3:2 rule of spending by default.\n\nPlease enter your budget to continue...')
        self.buttons()
        self.root.mainloop() # Keep UI alive

#   WIDGETS   #

    def draw(self, msg):
        self.frameText = tk.Frame(self.root, height=19, width=50)
        self.textBox = tk.Text(self.frameText, height=19, width=50, relief='flat', bg='light yellow', font=self.style)
        self.textBox.insert(1.0, msg)
        self.textBox.config(state='disabled')
        self.frameText.grid(row=0, column=0)
        self.textBox.pack(padx=3, pady=3)
```

```
    def buttons(self):
        self.frame = tk.Frame(self.root, height=1, width=50)
        self.frame.grid(row=0, column=1)
        self.label = tk.Label(self.frame, text='Your Budget:')
        self.entry = tk.Entry(self.frame, font=12)
        self.calBudget = tk.Button(self.frame, text='Calculate Budget',
command=self.calculateBudget, width=20)
        self.viewBudget = tk.Button(self.frame, text='View Budget Plan',
command=self.viewBudgetPlan, width=20)
        self.viewSpending = tk.Button(self.frame, text='View Spending Budget',
command=self.popUp, width=20)
        items = [self.label, self.entry, self.calBudget, self.viewBudget, self.viewSpending]
        for item in items: item.pack(padx=10, pady=20)

    def popUp(self):
        self.popUp = tk.Tk()
        self.popUp.geometry("200x200")
        self.popUp.title("Rent & Bills")
        self.popUp.resizable(width=False, height=False)
        self.rentLabel = tk.Label(self.popUp, text="Rent/Mortgage:")
        self.rentEntry = tk.Entry(self.popUp, width=25)
        self.billsLabel = tk.Label(self.popUp, text="Total Monthly Bills:")
        self.billsEntry = tk.Entry(self.popUp, width=25)
        self.done = tk.Button(self.popUp, text='Done', command=self.calculateSpending)
        items = [self.rentLabel, self.rentEntry, self.billsLabel, self.billsEntry, self.done]
        for item in items: item.pack(pady=5)

#   FUNCTIONS   #

    def calculateBudget(self):
        self.budget = float(self.entry.get() or 0)
        self.spending = self.budget * 0.5
        self.draw(f"Total Budget: ${'{:.2f}'.format(self.budget)}\nSpending Money:
${'{:.2f}'.format(self.spending)}")
```

```
    def viewBudgetPlan(self):
        self.budget = float(self.entry.get() or 0)
        self.spending = self.budget * 0.5
        self.savings = self.budget * 0.3
        self.extra = self.budget - self.spending - self.savings
        self.draw(f"Total Budget\t\t: ${'{:.2f}'.format(self.budget)}\nSpending Money\t\t:
${'{:.2f}'.format(self.spending)} \
            \nTo Save\t\t: ${'{:.2f}'.format(self.savings)}\nExtra\t\t: ${'{:.2f}'.format(self.extra)}")

    def calculateSpending(self):
        self.rent = float(self.rentEntry.get() or 0)
        self.bills = float(self.billsEntry.get() or 0)
        self.popUp.destroy()
        self.budget = float(self.entry.get() or 0)
        self.draw(f"### TOTAL BUDGET ###\nTotal\t\t: ${'{:.2f}'.format(self.budget)}\nSpending
Money\t\t: ${'{:.2f}'.format(self.budget*0.5)} \
            \n\n### EXPENSES ###\nRent\t\t: ${'{:.2f}'.format(self.rent)}\nBills\t\t:
${'{:.2f}'.format(self.bills)}")
        self.drawRealTimeGraph()

    def drawRealTimeGraph(self):
        # Draw real-time graph for Rent and Bills only
        plt.figure(figsize=(8, 5))
        categories = ["Rent", "Bills"]
        amounts = [self.rent, self.bills]
        plt.bar(categories, amounts, color='skyblue')
        plt.xlabel('Expense Categories')
        plt.ylabel('Amount')
        plt.title('Real-time Expenses Graph')
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

if __name__ == '__main__':
    MainWindow()
```

Explanation:

1. Import Statements:

tkinter: A standard Python interface to the Tk GUI toolkit.
tkinter is the standard Python interface to the Tk GUI toolkit. It is widely used for creating graphical user interfaces (GUIs) for desktop applications in Python. Tkinter provides a set of widgets (such as buttons, labels, entry fields, and canvas) that developers can use to build interactive and visually appealing applications. Here's an explanation of key concepts and functionalities provided by the tkinter module:

matplotlib.pyplot: A module from the matplotlib library used for creating visualizations and plots.

2. Global Variables:

expenses_list and labels_list are empty lists defined at the global scope.

3. Class Definition (MainWindow):

Defines a class named MainWindow.

4. Constructor (__init__ method):

Initializes the main window for the budget application using tkinter.
Creates various widgets such as labels, entry fields, and buttons for user interaction.
The calculateBudget(), viewBudgetPlan(), calculateSpending(), and drawRealTimeGraph() methods handle different aspects of budget calculation and visualization.

5. Widget Creation Methods:

draw(self, msg): Creates a text box for displaying messages to the user.
buttons(self): Creates buttons and entry fields for user input.
popUp(self): Creates a pop-up window for entering rent and bills information.

6. Event Handling Methods:

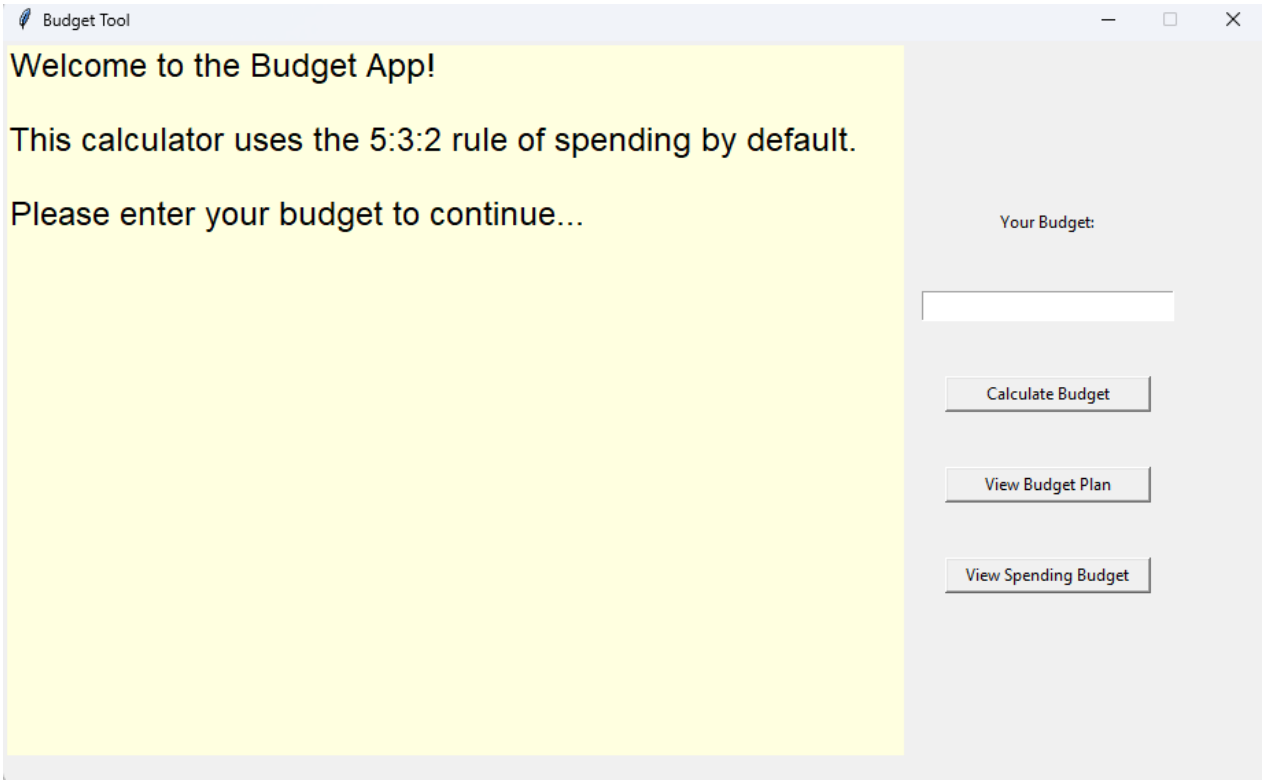
calculateBudget(self): Calculates the budget and spending money based on user input.
viewBudgetPlan(self): Calculates and displays total budget, spending money, savings, and extra money based on user input.
calculateSpending(self): Handles user input for rent and bills, updates the main window with the

entered expenses, and triggers the real-time graph drawing.
drawRealTimeGraph(self): Draws a real-time bar graph for rent and bills expenses using matplotlib.

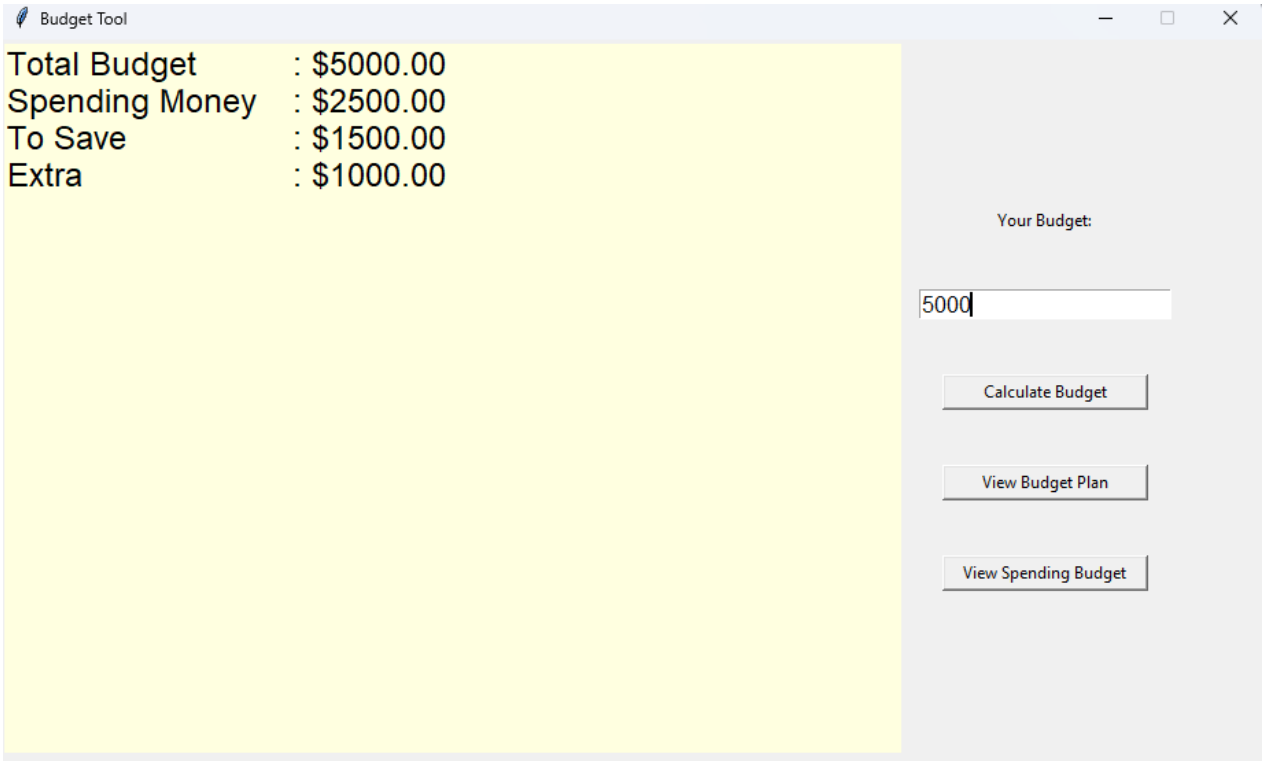
7. Main Program Execution:

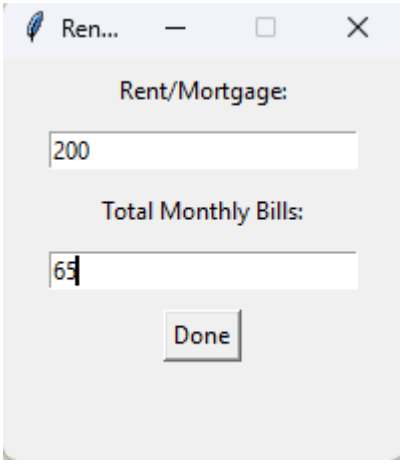
MainWindow()
If the script is run directly, it creates an instance of the MainWindow class, initiating the budget application and the graphical user interface**.**

Output:




Budget Tool
Total Budget: $5000.00
Spending Money: $2500.00
Your Budget:
5000
Calculate Budget
View Budget Plan
View Spending Budget


Budget Tool
Total Budget : $5000.00
Spending Money : $2500.00
To Save : $1500.00
Extra : $1000.00
Your Budget:
5000
Calculate Budget
View Budget Plan
View Spending Budget


Ren...
Rent/Mortgage:
200
Total Monthly Bills:
65
Done

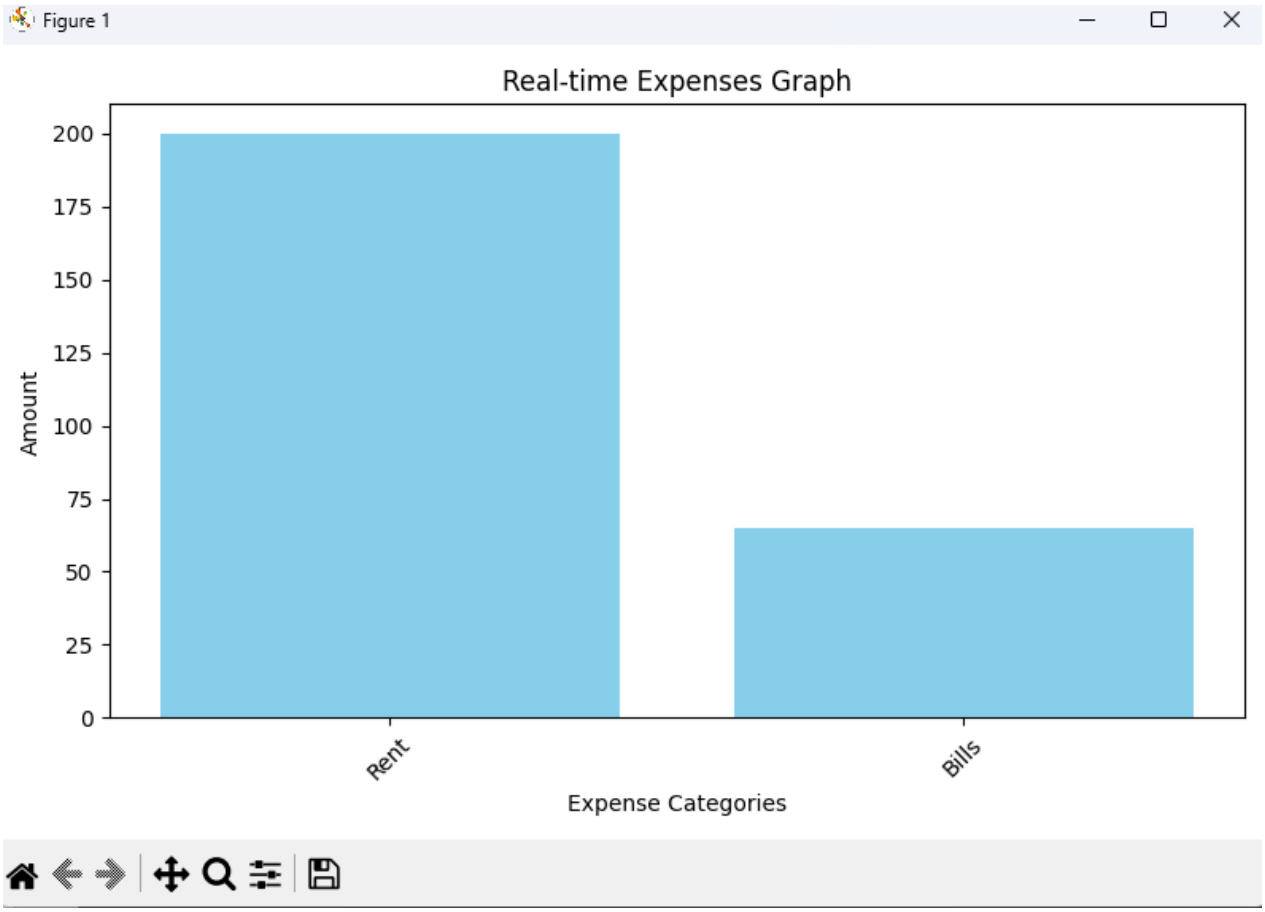
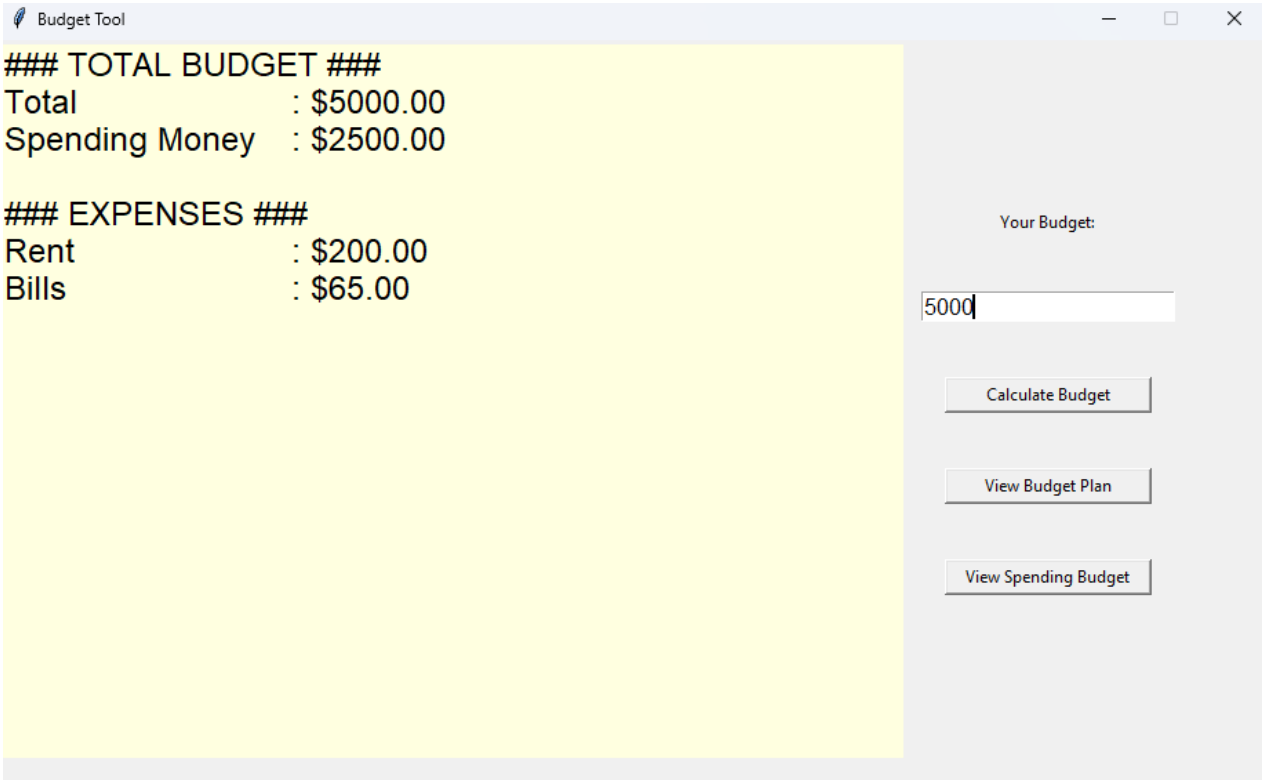


Figure 1
Real-time Expenses Graph
Amount
200
175
150
125
100
75
50
25
0
Rent
Bills
Expense Categories


Budget Tool
TOTAL BUDGET
Total : $5000.00
Spending Money : $2500.00
EXPENSES
Rent : $200.00
Bills : $65.00
Your Budget:
5000
Calculate Budget
View Budget Plan
View Spending Budget

6. RESULTS AND DISCUSSION

6.1 Results

CLI Version:

The CLI version of the budget calculator provides a straightforward user experience, allowing users to input their budget and view their spending, savings, and extra funds based on the 5:3:2 rule. Users can choose to view a detailed budget plan or analyze their spending budget by entering their rent, bills, and other monthly expenses. The CLI version generates clear and concise text-based outputs, helping users understand their financial allocations easily.

GUI Version:

The GUI version enhances the user experience by providing a visually appealing interface. Users can interactively input their budget and view their spending money, savings, and extra funds. Additionally, a pop-up window allows users to input specific details such as rent and bills, leading to a more accurate calculation of their expenses. The GUI version further includes real-time data visualization through a dynamic bar graph, presenting the distribution of spending between rent and bills.

6.2 Discussion

User-Friendly Interface:

The GUI version offers a more user-friendly experience, especially for individuals who may find command-line interfaces intimidating. The visually appealing design and interactive features enhance user engagement and understanding. The pop-up window for specific expense inputs simplifies the process, ensuring accurate calculations.

Data Visualization:

The dynamic bar graph provides users with an intuitive and graphical representation of their expenditure, making it easier to comprehend their budget breakdown. Visualization aids in identifying patterns and trends, empowering users to make informed financial decisions.

Educational Value:

Both versions of the budget calculator serve an educational purpose. They teach users the importance of budgeting, allocation strategies, and financial planning. By visually presenting the impact of rent, bills, and other expenses on the budget, users gain valuable insights into their spending habits and can adjust their financial priorities accordingly.

Versatility and Adaptability:

The availability of both CLI and GUI versions caters to a diverse user base. While some users may prefer the simplicity and efficiency of a command-line interface, others may find the graphical interface more appealing and intuitive. This versatility ensures that the budget calculator can reach a broader audience, regardless of their technical expertise.

Future Enhancements:

To further enhance the application, future iterations could include additional features such as expense categories (e.g., entertainment, transportation) and savings goals. Integrating a savings tracker or financial goal planner could empower users to set specific objectives and track their progress over time, fostering long-term financial discipline.

7. CONCLUSION

In the digital age, where managing personal finances is both a necessity and a challenge, the implementation of budgeting tools like the one presented in this project serves as a beacon of financial empowerment. Through a seamless integration of command-line interface (CLI) and graphical user interface (GUI) versions, this budgeting application offers users a multifaceted approach to understanding, planning, and visualizing their financial resources. The innovative use of Python, coupled with libraries such as tkinter and matplotlib, has resulted in an intuitive and interactive platform that caters to users of diverse technical backgrounds.

User-Friendly Experience:

One of the standout features of this budgeting tool is its user-friendly interface. The CLI version, with its simple text-based interactions, provides a quick and efficient way for users to input their budget and receive essential financial insights. On the other hand, the GUI version elevates the user experience significantly. By incorporating visually appealing elements and interactive widgets, it bridges the gap for users less accustomed to command-line interfaces. The GUI's real-time data visualization through dynamic bar graphs not only enriches the user experience but also aids in understanding expenditure patterns.

Educational Value:

Beyond its practical functionality, this budgeting application serves an educational purpose. It imparts financial knowledge by adhering to the popular 5:3:2 rule of spending, promoting the allocation of budget into essential expenses, discretionary spending, and savings. Users are not only able to view their budget breakdown but also gain insights into the importance of specific expense categories such as rent, bills, and discretionary spending. The application essentially acts as a virtual financial advisor, guiding users toward responsible financial planning.

Versatility and Adaptability:

The dual existence of both CLI and GUI versions showcases the versatility of the application. Users with varying technical preferences can comfortably choose the interface that best aligns with their comfort levels. This adaptability ensures that the tool caters to a broad user base,

spanning from tech-savvy individuals to beginners in the realm of personal finance.

Future Enhancements:

While the current version of the budgeting tool is robust and functional, there are avenues for future enhancements. Incorporating additional expense categories, such as entertainment, transportation, and healthcare, would offer a more comprehensive overview of an individual's financial landscape. Furthermore, the application could evolve into a holistic financial planning tool by integrating features like savings goal tracking, investment planning, and debt management. Such enhancements would transform the budgeting tool into an all-encompassing financial companion, guiding users through every facet of their financial journey.

Conclusion:

In conclusion, this budgeting application exemplifies the marriage of technology and financial literacy. By offering an accessible, interactive, and educational platform, it empowers individuals to take control of their finances and make informed decisions. Financial literacy is not merely a skill; it is an essential life tool that influences one's quality of life and future prospects. This project stands as a testament to the power of programming in fostering financial literacy, enabling individuals to navigate the complexities of personal finance with confidence and clarity. As users engage with this application, they embark on a journey toward financial resilience, armed with knowledge and understanding, ultimately shaping a more secure and prosperous future.

REFERENCES

Python Official Documentation: The official Python documentation
(https://www.python.org/doc/) is
an excellent resource for learning Python programming. It provides comprehensive guides, tutorials,
and references for Python developers of all levels.
Tkinter Documentation: For those specifically interested in GUI programming with Tkinter, the official Tkinter documentation (https://docs.python.org/3/library/tkinter.html) offers detailed information about the Tkinter module in Python.
Books:
"Python Crash Course" by Eric Matthes: This book provides a hands-on, project-based introduction
to Python programming and covers topics essential for game development.
"Invent Your Own Computer Games with Python" by Al Sweigart: This book introduces programming concepts through game development using Python.
"Python GUI Programming with Tkinter" by Alan D. Moore: This book focuses on GUI programming with Tkinter and is suitable for readers interested in creating graphical applications.
Online Courses and Tutorials:
Coursera (https://www.coursera.org/): Coursera offers various Python programming courses, including those specific to game development and GUI programming.
Udemy (https://www.udemy.com/): Udemy provides a wide range of Python courses, including game development and Tkinter tutorials.
Game Development Frameworks:
Pygame (https://www.pygame.org/): Pygame is a popular Python library for game development, providing tools and modules for creating games with graphics, sound, and user input.