EXPENSES TRACKER IN JAVA

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 JAVA** " is the bonafide work of **Ananya Pradeep (RA2211029010021)** and **Tina Kashyap (RA2211028010226)** who carried out the work under my supervision.

Dr. M. SUNDARRAJAN
ASSISTANT PROFESSOR

Department of Networking and Communication
SRMIST
Kattankulathur

Dr. ANNAPURANI K
HEAD OF DEPARTMENT
PROFESSOR

Department of Networking and Communication
SRMIST
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

1. INTRODUCTION

Introducing the Personal Expenses Tracker: In today's world, managing personal finances is more crucial than ever. To aid you in this endeavor, we present a robust and user-friendly budgeting solution, the Personal Expenses Tracker, developed using Java.

Two Interfaces, Your Choice: Our expense tracker offers two distinct interfaces to cater to your preferences. If you appreciate simplicity and efficiency, the Command Line Interface (CLI) is ideal. It guides you through setting your budget and follows the 5=3=2 spending rule. You can easily allocate your budget to spending, saving, and additional funds, empowering you to make informed financial decisions effortlessly.

For those who prefer an interactive and visually appealing experience, our Graphical User Interface (GUI) is the way to go. Upon launching the application, an inviting window prompts you to input your budget. With just a few clicks, you can access your budget plan, visualize your spending allocation, and even input specific details like rent and bills.

Real-Time Graphs for Clarity: Our tracker incorporates real-time graphs that dynamically illustrate your rental and bills expenses. This visual representation provides immediate clarity about the direction of your financial resources.

A Companion for Your Finances: Regardless of the interface you choose, the Personal Expenses Tracker is your faithful financial companion. It not only offers insights and guidance but also the flexibility to tailor your budget management to suit your unique needs.

Empower Your Financial Journey: It's time to seize control of your finances and embark on the path to financial freedom and peace of mind. Let the Personal Expenses Tracker simplify your budget management, all powered by Java.

1.1 Motivation

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.

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

Budget Calculation: Develop a robust system that enables users to input their total budget and utilizes the 5:3:2 rule to allocate funds, distinguishing between spending, saving, and other expenses.

User-Friendly GUI: Implement an easy-to-navigate graphical user interface (GUI) using Java Swing (or other suitable libraries) to ensure accessibility and user-friendliness for individuals with varying levels of technical expertise.

Budget Planning: Enable users to gain a clear view of their budget, categorizing it into spending money, savings, and surplus funds. Allow users to customize their savings allocation within the range of 20-30% to align with their financial goals.

Expense Entry: Design a straightforward pop-up window where users can effortlessly input their basic monthly expenses, including but not limited to rent and bills.

Real-Time Expense Visualization: Utilize the Matplotlib library (or an appropriate alternative) to generate dynamic graphs that provide real-time insights into users' monthly expenses.

Financial Insights: Provide users with valuable financial information and insights. Help them understand their spending patterns, assess their budgeting behaviors, and gain a comprehensive understanding of their financial situation.

Usability: Ensure that the application offers an intuitive and accessible user interface, catering to a wide range of users with varying levels of technical proficiency.

Smooth User Experience: Create a seamless user experience that simplifies the process of tracking expenses and making financial decisions, resulting in an efficient and enjoyable interaction.

Reliable Error Handling: Implement robust error handling mechanisms to guide users and prevent unexpected issues, ensuring that the application runs smoothly and reliably.

Expected Result:

The Expenses Tracker App for Budget Management aims to provide a holistic solution for financial management. It is designed to empower users to take better control of their finances. This Java-based application will facilitate tracking of spending, enabling users to make informed financial decisions, leading to enhanced financial well-being. The project encompasses the development of both a command-line budget calculator and a feature-rich graphical user interface (GUI) app for seamless expense tracking.

1.4 Challenges

User Interface Complexity: Designing an intuitive and user-friendly interface is a challenge. Striking a balance between offering feature-rich functionality and maintaining a clean and clutter-free design is crucial.

Input Validation: Validating and sanitizing user inputs is essential to prevent errors and security vulnerabilities. Inaccurate or maliciously entered data can disrupt the application's operation.

Cross-Platform Compatibility: Ensuring that the Expenses Tracker functions seamlessly across various operating systems and screen resolutions is a vital consideration. Consistency in behavior across platforms is equally important.

Real-Time Data Visualization: Implementing real-time data visualizations, particularly dynamic graphs, can be technically challenging. It requires efficient and optimized coding to provide users with a smooth experience.

Error Handling: Robust error handling mechanisms are imperative to guide users through unexpected scenarios and prevent application crashes, enhancing reliability.

Data Security and Resilience: Safeguarding user data and fortifying the application against security threats is a constant concern. The risk of data breaches necessitates stringent security measures.

Compliance and Regulations: If the application deals with sensitive financial data, adhering to different regulations and security standards can be complex. Compliance is vital to maintain the trust of users.

User Feedback and Iteration: Collecting and incorporating user feedback for continuous improvement is essential for enhancing the application's features and usability.

Documentation: Developing comprehensive and user-friendly documentation is time-consuming but necessary. It aids users in understanding and navigating the application effectively.

Collaboration with a Development Team: If the project involves multiple developers, effective code management and version control systems are critical to ensure smooth collaboration and coordinated code changes.

In summary, while developing the Expenses Tracker in Java, addressing these challenges will contribute to creating a robust and user-friendly financial management solution. These challenges reflect the typical considerations in software development, particularly when dealing with sensitive financial data and providing a seamless user experience.

2. LITERATURE SURVEY

Popular Expense Tracking Applications: What are the most widely used expense tracking apps and software in the market, and what are their key features and functionalities?

User Interface and User Experience Analysis: How do existing expense tracking applications design their user interfaces and user experiences to meet the needs of diverse user groups?

Innovative and Unique Functions: What innovative or unique features and functions are introduced by expense tracking software to provide users with added value?

Real-Time Data Visualization Technologies: What technologies and libraries are commonly employed for real-time expense data visualization in expense tracking applications?

User Interface and Experience Features: What are the essential user interface and experience features that successful expense tracking software offer to their users?

Importance of User-Friendly Interfaces: Why is it crucial for expense tracking applications to prioritize a user-friendly interface and incorporate features like input validation for a smooth and reliable user experience?

Budget Planning and Management Tools: What tools are commonly used within expense tracking applications to facilitate budget planning and management?

Expense Tracking and Financial Reporting Tools: What tools are utilized for tracking expenses and generating financial reports within these applications?

Mobile Adaptation: How can expense tracking applications be adapted to function effectively on mobile devices, and how does this adaptation improve user experience and portability?

Security Features in Financial Software: What security features, such as encryption, authentication, and data protection, are implemented in financial software to safeguard user financial data?

Privacy and Data Protection: Why is privacy and data protection critical in the context of financial applications, and how do these applications ensure the confidentiality of user financial data?

Advantages of Cloud-Based Solutions: What advantages do cloud-based expense tracking solutions offer, such as data accessibility and synchronization across multiple devices, and how do they impact the user experience?

User Feedback and Continuous Improvement: How do expense tracking applications collect and utilize user feedback to drive continuous improvement, enhancing the user experience over time?

Community Support and Updates: How can community support and regular updates be leveraged to enhance the user experience and feature set of expense tracking applications?

Cross-Platform Access: How can cross-platform expense tracking software enable users to access their financial data seamlessly across different operating systems?

Data Analysis and Visualization Techniques: What techniques are commonly used for data analysis and visualization in financial applications, aiding users in gaining insights into their financial data?

Exploring these research questions will provide valuable insights into the landscape of expense tracking software and the strategies and technologies used to develop successful financial management applications based on the Expenses Tracker Java code.

2. REQUIREMENTS

3.1 Requirement Analysis

Budget Calculation and Management:

Users can input their total budget.
The default spending amount is set at 50% of the budget.
Users can update the budget based on changes in income.
Users can select a savings plan.
The budget plan is automatically calculated.
Extra funds available for discretionary spending are determined.
The remaining budget after savings is calculated.
Users can edit or delete expense entries.
Real-time graphs provide insights into 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 can save their data and expense history for future reference.
Data Analysis:

Analytical tools are provided to help users gain insights into their financial data.
Non-Functional Requirements:

User Interface (UI):

The UI should be intuitive, user-friendly, and visually appealing.
Cross-Platform Compatibility:

The application should be compatible with various platforms, including desktop and mobile devices.

Security:

User financial data should be encrypted and authenticated to ensure privacy.
Passwords should be securely stored, either hashed or salted.
Cloud Integration:

The application should utilize cloud storage for data synchronization, allowing users to access their information from multiple devices.
Data Security in Transit:

Data transmission should be encrypted to protect user information.
Financial Compliance:

The application should adhere to relevant financial regulations (if applicable).
Compliance with data protection standards such as GDPR should be ensured.
User Documentation:

Provide comprehensive user documentation and accessible help resources.
Clear instructions for all application functions and features should be available.
Feedback and Updates:

Include a feedback mechanism to allow users to report issues and offer suggestions for improvements.
Plan and implement regular updates to enhance functionality and address reported issues.
By addressing both the functional and non-functional requirements, the Expenses Tracker app can provide users with a robust set of tools for budget management, expense tracking, and informed financial decision-making.

4.ARCHITECTURE AND DESIGN

4.1 Introduction

Budget Management Application is designed to assist users in effectively managing their finances. The application is composed of two key components: the CLI (Command-Line Interface) Budget Calculator and the JavaFX GUI (Graphical User Interface). These two components provide different ways for users to interact with the application.

User Interface (UI):

- The application is developed using JavaFX, which provides a robust framework for creating graphical user interfaces.
- The UI features various windows, frames, labels, buttons, and text fields that facilitate user interaction and input.

Business Logic:

- In the CLI version, the core business logic is encapsulated within the BudgetCalculator class, while the JavaFX GUI version's logic is primarily managed by the MainWindow class.
- The business logic is responsible for performing budget calculations, validating user input, and managing user interactions.

Data Storage:

- In both versions of the application, temporary storage for expense data is achieved through lists (e.g., expensesList and labelsList). These lists are instrumental in generating real-time expense graphs.

Plotting:

- Real-time expense graphs are created and displayed using JavaFX charts in the GUI version, providing visual representations of budget allocations and spending trends.

Communication:

- The CLI version communicates with the user through the command-line interface, gathering user input and displaying output.
- In the GUI version, the communication takes place via the graphical interface, updating UI components with calculated data and rendering expense graphs to provide visual feedback.

Main Loop:

- The CLI version follows a linear execution flow, guiding users through budget input and various available options.
- The JavaFX GUI version operates within an event-driven model, where user actions, such as button clicks, trigger events that are handled by dedicated event handlers.

User Input:

- Both versions of the application collect user input, including the budget amount, rent, bills, and other relevant financial details.
- Input validation is employed to ensure that users provide valid numeric values when required.

Budget Calculator:

- The application computes the budget based on user input and employs the 5:3:2 rule to determine various expense categories, including spending, savings, and other expenses.

UI Updates:

- In the JavaFX GUI version, the UI is dynamically updated to display the calculated budget, spending allocations, savings, and detailed expense breakdowns. Labels, text fields, and buttons are utilized to present and interact with this data, providing users with an intuitive and visually appealing interface for managing their finances.

5. IMPLEMENTATION

The budget calculator application comprises two main versions: a Command-Line Interface (CLI) version and a Graphical User Interface (GUI) version developed using Java. In the CLI version, users are prompted to input their budget, which is then calculated based on the 5:3:2 rule. Users have the option to view detailed information regarding their budget and expenses, including categories such as rent, bills, and food. The CLI version employs a straightforward text-based interface and operates sequentially, gathering user input and displaying outcomes in the console.

On the other hand, the GUI version is created using the JavaFX library, offering an intuitive and interactive user experience. Users enter their budget into the designated text field and can utilize buttons to perform budget calculations, access budget details, and input rent and expenses. The GUI dynamically updates labels and text fields to present financial information and expense breakdowns. Additionally, a pop-up window is available for users to input details like rent and expenses, while a real-time graph feature visually represents rent and charges.

Both versions share a common backend logic for calculating and presenting financial information. The GUI edition enhances the user experience by delivering visually appealing interfaces, thereby making budget management more attractive and user-friendly. Overall, the implementation illustrates Java's versatility in developing applications for both command-line and graphical user interfaces to cater to diverse user preferences and requirements.

CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;

public class ExpensesTracker {
    private JFrame frame;
    private JTextField budgetField;
    private JTextField rentField;
    private JTextField billsField;
    private JTextArea resultArea;
    private List<Double> expensesList;

    public ExpensesTracker() {
        frame = new JFrame("Expenses Tracker");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);

        JPanel panel = new JPanel(new GridLayout(6, 2));
        JLabel budgetLabel = new JLabel("Enter your budget:");
        budgetField = new JTextField();
        JLabel rentLabel = new JLabel("Enter rent/mortgage:");
        rentField = new JTextField();
        JLabel billsLabel = new JLabel("Enter monthly bills:");
        billsField = new JTextField();
        JButton calculateButton = new JButton("Calculate");
        JButton addExpenseButton = new JButton("Add Expense");
        JButton viewExpensesButton = new JButton("View Expenses");
        resultArea = new JTextArea();
```

```
resultArea.setEditable(false);
expensesList = new ArrayList<>();

panel.add(budgetLabel);
panel.add(budgetField);
panel.add(rentLabel);
panel.add(rentField);
panel.add(billsLabel);
panel.add(billsField);
panel.add(calculateButton);
panel.add(addExpenseButton);
panel.add(viewExpensesButton);

calculateButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        calculateExpenses();
    }
});

addExpenseButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        addExpense();
    }
});

viewExpensesButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        viewExpenses();
    }
});
```

```
        frame.setLayout(new BorderLayout());
        frame.add(panel, BorderLayout.CENTER);
        frame.add(resultArea, BorderLayout.SOUTH);
        frame.setVisible(true);
    }

    private void calculateExpenses() {
        try {
            double budget = Double.parseDouble(budgetField.getText());
            double rent = Double.parseDouble(rentField.getText());
            double bills = Double.parseDouble(billsField.getText());

            double spending = budget * 0.5;
            double savings = budget * 0.3;
            double remaining = budget - rent - bills - spending;

            resultArea.setText("Spending Money: $" + String.format("%.2f", spending) +
                    "\nTo Save: $" + String.format("%.2f", savings) +
                    "\nRemaining: $" + String.format("%.2f", remaining));

            showGraphPopup(spending, savings, remaining);
        } catch (NumberFormatException ex) {
            resultArea.setText("Invalid input. Please enter valid numbers.");
        }
    }

    private void addExpense() {
        try {
            double expense = Double.parseDouble(JOptionPane.showInputDialog(frame, "Enter
expense amount:"));
            expensesList.add(expense);
            JOptionPane.showMessageDialog(frame, "Expense added successfully!");
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(frame, "Invalid input. Please enter a valid
```

```
number.");
        }
    }

    private void viewExpenses() {
        double totalExpenses = 0;
        StringBuilder expensesDetails = new StringBuilder("Expenses List:\n");
        for (Double expense : expensesList) {
            expensesDetails.append("$").append(expense).append("\n");
            totalExpenses += expense;
        }
        resultArea.setText(expensesDetails.toString() + "Total Expenses: $" + totalExpenses);
    }

    private void showGraphPopup(double spending, double savings, double remaining) {
        double misc = 0;
        for (Double expense : expensesList) {
            misc += expense;
        }

        final double finalSpending = spending;
        final double finalSavings = savings;
        final double finalRemaining = remaining;
        final double finalMisc = misc;

        JFrame graphFrame = new JFrame("Expenses Graph");
        graphFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        graphFrame.setSize(400, 300);

        JPanel graphPanel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                int total = (int) (finalSpending + finalSavings + finalRemaining + finalMisc);
```

```
            int panelWidth = getWidth();
            int panelHeight = getHeight();
            int spendingHeight = (int) (finalSpending / total * panelHeight);
            int savingsHeight = (int) (finalSavings / total * panelHeight);
            int remainingHeight = (int) (finalRemaining / total * panelHeight);
            int miscHeight = (int) (finalMisc / total * panelHeight);

            g.setColor(Color.RED);
            g.fillRect(50, panelHeight - spendingHeight, 50, spendingHeight);
            g.drawString("Spending", 50, panelHeight - spendingHeight - 5);

            g.setColor(Color.GREEN);
            g.fillRect(150, panelHeight - savingsHeight, 50, savingsHeight);
            g.drawString("Savings", 150, panelHeight - savingsHeight - 5);

            g.setColor(Color.BLUE);
            g.fillRect(250, panelHeight - remainingHeight, 50, remainingHeight);
            g.drawString("Remaining", 250, panelHeight - remainingHeight - 5);

            g.setColor(Color.ORANGE);
            g.fillRect(350, panelHeight - miscHeight, 50, miscHeight);
            g.drawString("Misc", 350, panelHeight - miscHeight - 5);

            // Draw y-axis numbers
            g.setColor(Color.BLACK);
            g.drawString(Integer.toString(total), 30, 15);
            g.drawString(Integer.toString(total / 2), 30, panelHeight / 2);
            g.drawString("0", 30, panelHeight - 5);
        }
    };

    graphFrame.add(graphPanel);
    graphFrame.setVisible(true);
}
```

```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new ExpensesTracker();
            }
        });
    }
}
```

EXPLANATION:

This Java code represents an Expenses Tracker application that provides users with a graphical user interface (GUI) for budget calculation and expense tracking. Let's break down the code to understand its functionality:

1. Import Statements:
   - The code begins with import statements to include the necessary Java libraries for creating the GUI and managing expenses.

2. ExpensesTracker Class:
   - This class is the core of the application and is responsible for creating the GUI and handling user interactions.

3. Constructor (public ExpensesTracker()):
   - The constructor sets up the GUI components and initializes the application.
   - It creates a JFrame (window) with the title "Expenses Tracker" and sets its size to 600x400 pixels.

4. Components Setup:
   - The following components are added to the GUI:
     - `JPanel panel`: A container for organizing the components.
     - Labels (`JLabel`) for user instructions and input fields.
     - Text fields (`JTextField`) for entering the budget, rent, and bills.
     - A button (`JButton`) labeled "Calculate" to trigger expense calculations.
     - A text area (`JTextArea`) for displaying the results, which is set to be not editable.

5. Calculate Button ActionListener:
   - An ActionListener is added to the "Calculate" button, defining what happens when the button is clicked. When clicked, it calls the `calculateExpenses()` method.

6. Chart Panel Initialization:
   - A chart panel (`ChartPanel`) is added to display a bar chart. This chart will show a visual representation of the budget breakdown, specifically for spending, savings, and remaining funds.

The chart starts with default values.

7. calculateExpenses() Method:
   - This method is called when the "Calculate" button is pressed.
   - It performs the following tasks:
     - Reads user input for budget, rent, and bills.
     - Calculates spending, savings, and remaining funds based on the 5:3:2 rule.
     - Updates the text area to display the calculated results.
     - Updates the bar chart with the new expense breakdown.

8. Main Method (public static void main(String[] args)):
   - The application is started in the `main` method.
   - It uses `SwingUtilities.invokeLater()` to ensure the GUI is created and executed on the Event Dispatch Thread, which is the correct thread for managing GUI components.

Overall, this Java code creates a user-friendly GUI application for budget management. Users can input their budget, rent, and bills, and the application calculates and displays spending, savings, and remaining funds, along with a visual representation of the budget breakdown. It provides an easy way for individuals to track their expenses and make informed financial decisions.

6. RESULTS AND DISCUSSION

6.1 Results

CLI Version:

The CLI version of the budget calculator offers a straightforward user experience, enabling users to input their budget and access their spending, savings, and additional funds based on the 5:3:2 rule. Users have the option 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, facilitating users' comprehension of their financial allocations.

GUI Version:

The GUI version enhances the user experience by providing an aesthetically pleasing interface. Users can interactively input their budget and examine their spending money, savings, and additional funds. Furthermore, a pop-up window allows users to enter specific details like rent and bills, resulting in a more precise calculation of their expenses. The GUI version also includes real-time data visualization through a dynamic bar graph, illustrating the distribution of spending between rent and bills.

6.2 Discussion

User-Friendly Interface:

The GUI version delivers 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 comprehension. 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 grasp their budget breakdown. Visualization assists 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 educate users on the significance of budgeting, allocation strategies, and financial planning. By visually illustrating 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 monitor 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 a command-line interface (CLI) and graphical user interface (GUI) versions, this Java-based budgeting application offers users a multifaceted approach to understanding, planning, and visualizing their financial resources. The innovative use of Java, coupled with libraries such as JavaFX and JFreeChart, 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 the 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

1. https://docs.oracle.com/javase/8/docs/api/java/net/HttpURLConnection.html
2. https://www.google.com/
3. Test Driven: TDD and Acceptance TDD for Java Developers - Lasse Koskela
4. Thinking in Java - Bruce Eckel
5. Java fundamentals course - Coursera
6. Java development course – Coursera
7. Core Java Volume I – Fundamentals – Cay S. Horstmann
8. Extreme Java - Concurrency Performance for Java 8 - Dr. Heinz Kabutz
9. Thinking in Java - Bruce Eckel
10. The Java Language Specification - Bill Joy, Gilad Bracha, Guy L. Steele Jr., and James Gosling