

BUDGET CALCULATOR



A PROJECT REPORT

Submitted by

EFFIN STRAYA J (8115U23EC024)

in partial fulfillment of requirements for the award of the course

EGB1201 - JAVA PROGRAMMING

in

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

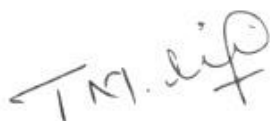
DECEMBER - 2024

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING
(AUTONOMOUS)**

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “ **BUDGET CALCULATOR**” is the bonafide work of **EFFIN STRAYA J (8115U23EC024)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



SIGNATURE

Dr. T. M. NITHYA, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

ASSOCIATE PROFESSOR

Department of CSE

K.Ramakrishnan College of Engineering
(Autonomous)

Samayapuram-621112.



SIGNATURE

Mr.V.KUMARARAJA, M.E.,(Ph.D.),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE


K.Ramakrishnan College of Engineering
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 06/12/24



INTERNAL EXAMINER



EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**BUDGET CALCULATOR**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **EGB1201 - JAVA PROGRAMMING**.

Signature

00

Effin Straya J (6/12/2024)

EFFIN STRAYA J

Place: Samayapuram

Date:06/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Engineering (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. D. SRINIVASAN, B.E, M.E., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. M. NITHYA, M.E.,Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR.V.KUMARARAJA, M.E., (Ph.D.)**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To achieve a prominent position among the top technical institutions.

MISSION OF THE INSTITUTION

- M1: To bestow standard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.
- M3: To provide education for developing high-quality professionals to transform the society.

VISION OF DEPARTMENT

To create eminent professionals of Computer Science and Engineering by imparting quality education.

MISSION OF DEPARTMENT

M1: To provide technical exposure in the field of Computer Science and Engineering through state of the art infrastructure and ethical standards.

M2: To engage the students in research and development activities in the field of Computer Science and Engineering.

M3: To empower the learners to involve in industrial and multi-disciplinary projects for addressing the societal needs.

PROGRAM EDUCATIONAL OBJECTIVES

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.
- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

A budget calculator is a vital tool for managing personal finances effectively, allowing users to allocate and track their income and expenses systematically. This project aims to develop a budget calculator using Abstract Window Toolkit (AWT), a part of Java's graphical user interface (GUI) library, to provide a simple and interactive solution for financial planning. The application enables users to input income sources and categorize expenses such as rent, groceries, utilities, and savings, facilitating a clear overview of their financial status. By leveraging AWT's lightweight components, the budget calculator delivers a user-friendly interface that includes text fields, labels, buttons, and choice menus for seamless navigation and data input. The core functionality of the budget calculator revolves around its ability to calculate the total income, aggregate expenses, and determine the balance. Users can enter data dynamically, which is instantly reflected in the calculations, ensuring real-time updates. Additionally, the calculator offers features like graphical representation of financial data, including bar charts and pie charts, to provide visual insights into spending patterns. The use of AWT simplifies the development process, offering platform independence and lightweight performance suitable for standalone applications. However, the project also addresses AWT's limitations, such as limited aesthetic flexibility, by focusing on functional utility and ease of use. The budget calculator supports a broad audience, from students managing their allowances to professionals handling complex financial schedules. The intuitive design makes it accessible to users with minimal technical knowledge, while the inclusion of advanced features like exportable reports in CSV format caters to more experienced users. The application also emphasizes security by restricting data storage to the user's local system, alleviating concerns about privacy and unauthorized access.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>This project implements a Budget Calculator using Java's Abstract Window Toolkit (AWT) for a user-friendly graphical interface. It enables users to input income, expenses, and savings goals, dynamically calculating budget summaries and providing visual insights. Designed with simplicity and functionality, the application ensures efficient financial tracking and planning. Utilizing AWT components like text fields, buttons, and labels, it delivers a lightweight yet robust solution for personal finance management.</p>	<p>PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3</p>	<p>PSO1 -3 PSO2 -3</p>

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	PROJECT METHODOLOGY	4
	2.1 Proposed Work	4
	2.2 Block Diagram	5
3	MODULE DESCRIPTION	6
	3.1 User Interface Layout and Components	6
	3.2 Expense Addition and Handling	6
	3.3 Budget Calculation	7
	3.4 Expense List Management	7
	3.5 Clear Functionality and Window Management	8
4	CONCLUSION & FUTURE SCOPE	9
	4.1 Conclusion	9
	4.2 Future Scope	10
	APPENDIX A (SOURCE CODE)	12
	APPENDIX B (SCREENSHOTS)	17
	REFERENCES	18

LIST OF FIGURES

FIG.NO	FIGURES	PAGE NO
2.1	Block Diagram of Budget Calculator	5
1	Output Part 1(Login Page)	17
2	Output Result	18

CHAPTER 1

INTRODUCTION

1.1 Objective

The Budget Calculator using Abstract Window Toolkit (AWT) provides a user-friendly desktop application that helps individuals or households effectively manage their finances. This application leverages AWT, a Java-based graphical user interface (GUI) toolkit, to create an intuitive and interactive interface for users to input, organize, and analyze their income and expenses. It aims to simplify the budgeting process by enabling users to categorize expenses, set budget limits, and track their financial performance over time. The application seeks to enhance financial literacy by offering features such as real-time calculation of remaining budget, graphical representation of spending patterns, and alerts for exceeding budgetary thresholds. The tool ensures data accuracy and reliability by implementing robust input validation and error handling. Overall, the Budget Calculator aims to empower users to make informed financial decisions, avoid overspending, and achieve their financial goals efficiently. Its core design focuses on simplicity, functionality, and adaptability, catering to a broad audience seeking a straightforward solution for budget management. Additionally, it emphasizes cross-platform compatibility and lightweight performance, making it accessible to users with varying technical expertise and hardware configurations.

1.2 Overview

The Budget Calculator using Abstract Window Toolkit (AWT) is a desktop-based financial management tool designed to assist users in planning and tracking their budgets. Built using Java's AWT framework, the application provides a simple and interactive interface for managing personal or household finances. It allows users to input income, categorize expenses, and allocate budgets for specific categories. The tool performs real-time calculations to display insights, such as total income, total

expenses, and remaining budget, enabling users to monitor their financial health effectively. One of the primary features of the Budget Calculator is its ability to generate visual representations, such as charts or graphs, offering a clear view of spending patterns and budget utilization. Additionally, it includes options to set alerts when expenses exceed defined limits, promoting better financial discipline. The application is designed to be lightweight, cross-platform, and user-friendly, ensuring accessibility for users with varying technical skills. The Budget Calculator prioritizes simplicity while providing essential budgeting features, making it suitable for individuals looking for a straightforward and efficient solution to manage their finances. Integrating intuitive design with practical functionality helps users achieve their financial goals and make informed decisions with minimal effort.

1.3 Java Programming Concepts

Developing a Budget Calculator using Abstract Window Toolkit (AWT) involves utilizing several Java programming concepts. The key concepts applied in creating it are:

Object-Oriented Programming (OOP):

Core principles such as encapsulation, inheritance, and polymorphism are used to design and structure the application. Classes and objects represent components like input fields, buttons, and calculation logic.

Abstract Window Toolkit (AWT):

AWT components such as Frame, Panel, Label, TextField, Button, and Checkbox are used to create the graphical user interface (GUI). Layout managers like FlowLayout, BorderLayout, or GridLayout arrange UI components. Event-handling mechanisms manage user interactions like button clicks and text input.

Event Handling:

Implementing the ActionListener interface to capture and process events like button clicks. Using classes such as(ActionEvent) to trigger specific actions in response to user interactions.

Data Handling and Processing:

Variables and data structures (e.g., arrays or lists) store and manage user input.

Basic arithmetic operations perform budget calculations.

Error Handling:

The try-catch blocks handle exceptions like invalid numeric input, ensuring a robust application.

File Handling :

Reading and writing files to save or load budget data, enabling persistent storage.

Threading:

Using threads for tasks like updating graphical elements or processing calculations without freezing the interface.

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the Budget Calculator Project involves developing a graphical user interface (GUI) application using Java's Abstract Window Toolkit (AWT). This application is designed to assist users in efficiently managing their financial activities by allowing them to record their income and expenses, categorize spending, and calculate the remaining budget. The primary objective is to provide an intuitive, interactive, and user-friendly tool for individuals or households to track and optimize their finances. The application will consist of several key components. Users can input their total income and add multiple expenses categorized into predefined types such as Food, Transport, Utilities, Entertainment, and Others. A dynamic display will list all expenses, including their categories, enabling users to review their spending habits. The program will calculate the total expenses and provide real-time updates on the remaining budget, displayed in a dedicated field.

Additionally, the tool includes functionality for clearing all inputs and calculations, allowing users to reset and start a new budgeting session. Robust input validation will ensure that only numeric values are accepted, preventing errors during calculations. The program's layout is structured using AWT components such as Frame, Panel, Label, TextField, Choice, and Button, along with event-handling mechanisms to manage user interactions seamlessly. The proposed work aims to combine simplicity with practical functionality, ensuring accessibility for users with varying technical expertise. The final product will serve as an effective tool for promoting financial discipline and helping users achieve their budgeting goals.

2.2 Block Diagram

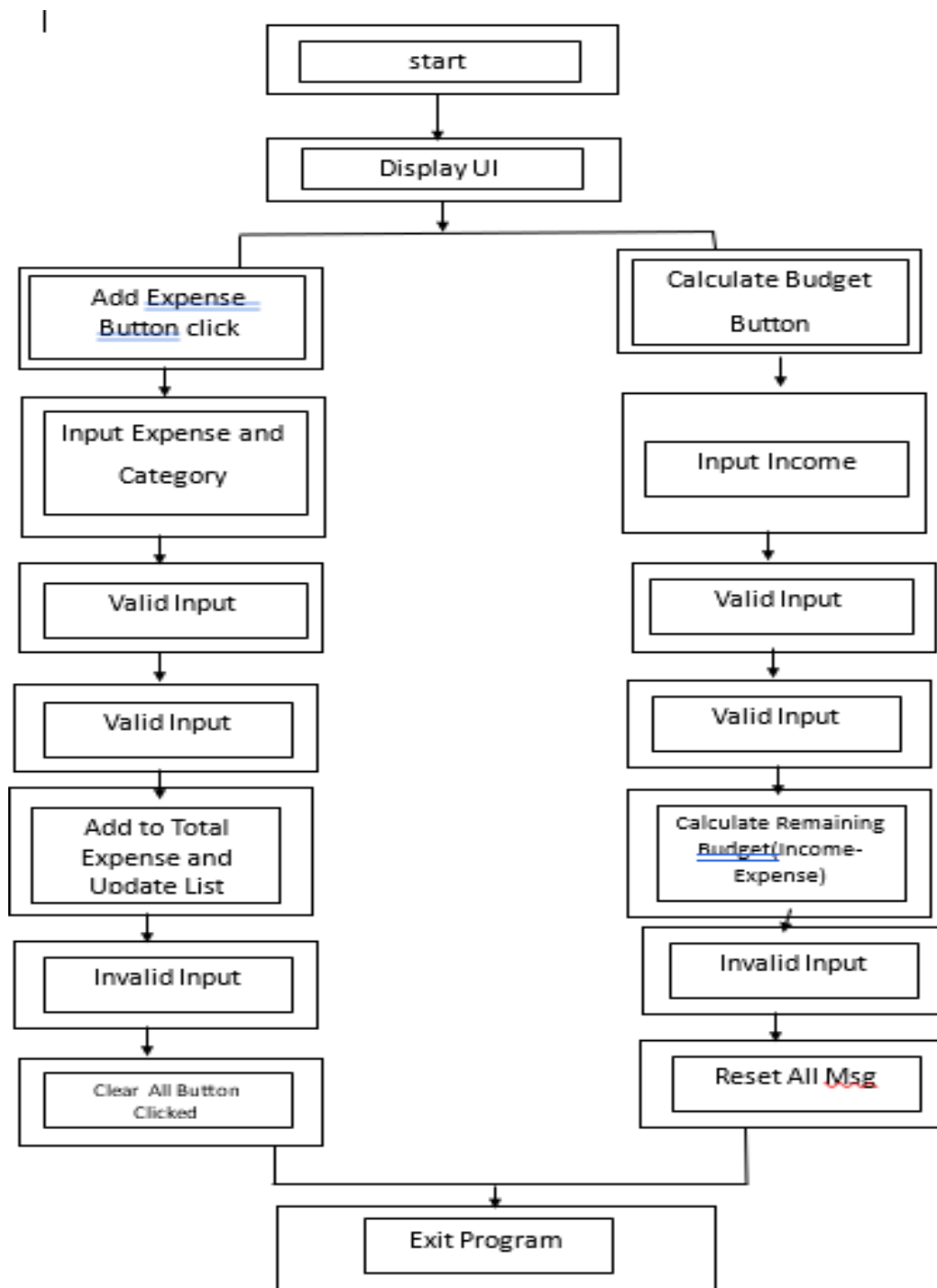


Fig 2.1 Block Diagram of Budget Calculator

CHAPTER 3

MODULE DESCRIPTION

3.1 Module 1 User Interface Layout and Components

In this module, the program sets up the graphical user interface (GUI) using AWT components. The main layout of the window is managed with a BorderLayout, dividing the window into several sections. At the top, a Label titled "Budget Calculator" acts as the header, using a large font for emphasis. The main section of the window (center) contains a Panel with a GridLayout that organizes the components into six rows and two columns. The user can input their total income using a TextField next to the "Enter Income" label. Similarly, for entering expenses, the program uses another TextField next to the "Enter Expense Amount" label. Users can select an expense category using a Choice component, which includes options like "Food", "Transport", "Utilities", "Entertainment", and "Other". Below the income and expense input fields, the "Remaining Budget" field displays the calculated remaining budget. A TextArea on the east side of the window is used to display a list of expenses added by the user, showing both the category and the amount. This structure helps the program remain organized, with distinct areas for data entry, displaying results, and listing expenses, all within a user-friendly interface.

3.2 Module 2 Expense Addition and Handling

In this module, the program allows users to add individual expenses. When the user clicks the "Add Expense" button, the actionPerformed() method listens for the event. First, the program retrieves the expense value entered in the expenseField using Double.parseDouble(). If the input is valid, it is added to the totalExpenses variable. The program also fetches the selected expense category from the categoryChoice component, which helps classify the expenses. This category and the corresponding amount are appended to the expenseList (a TextArea), allowing the user to see a running list of all expenses. If the user enters invalid data, such as a non-numeric value,

a `NumberFormatException` is caught, and the program displays an error message in the expense field. After adding an expense, the input field is cleared, and the program is ready to accept the next input. This functionality ensures that the user can continuously input and view their expenses without the application crashing due to input errors.

3.3 Module 3 Budget Calculation

The "Calculate Budget" button triggers the `calculateButton` action, where the program computes the remaining budget. When this button is clicked, the program retrieves the total income entered in the `incomeField` and tries to parse it into a double. The program then subtracts the total expenses (stored in the `totalExpenses` variable) from the total income to calculate the remaining budget. The result is formatted to two decimal places and displayed in the `resultField`, which is non-editable to ensure the result cannot be tampered with. If the income is not valid (for example, if the user enters a non-numeric value), a `NumberFormatException` is caught, and the program displays an error message in the result field, prompting the user to correct the income input. This step ensures that users can easily calculate and view their remaining budget, based on the expenses they have entered.

3.4 Module 4 Expense List Management

In this module, the program focuses on maintaining and displaying the list of expenses entered by the user. Each time a new expense is added, it is appended to the `expenseList` (a `TextArea`), which shows the expense category and the amount entered. This list gives the user a clear view of where their money is going. The list is structured to show both the category of each expense (e.g., "Food" or "Transport") and the corresponding amount spent, making it easy for users to track their spending. The `TextArea` is non-editable to prevent users from manually altering the displayed expenses. The expense list updates dynamically as new expenses are added, providing

real-time feedback. The format "Category: Amount" makes the list easy to understand. This module is essential for the user to visually track and manage multiple expenses, making it a key feature for those aiming to stay within their budget.

3.5 Module 5 Clear Functionality and Window Management

This module focuses on the program's ability to clear all fields and reset the data when needed. The "Clear All" button triggers the `clearButton` action, resetting the `totalIncome`, `totalExpenses`, and all input fields to their initial empty states. After clicking "Clear All," the income field, expense field, result field, and expense list are all cleared, allowing the user to start a new budgeting session. This functionality is essential for users who wish to begin a fresh calculation without closing and reopening the application. In addition, the `WindowListener` is used to handle the window's closing event. When the user closes the application window, the program terminates using `System.exit(0)`, ensuring that the application shuts down without leaving any unnecessary processes running in the background. The ability to reset and close the application smoothly enhances user experience and ensures the program is easy to use for repeated sessions.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, the Budget Calculator Project demonstrates how a simple yet effective budgeting tool can be created using Java's AWT framework. It serves as a solid foundation for individuals looking to manage their finances with ease, providing them with a clear picture of their income, expenses, and remaining budget. This project highlights the importance of software in daily life and showcases the ability of Java to create interactive, practical applications that can address real-world needs. As a beginner-friendly project, it also serves as an excellent learning resource for anyone looking to explore GUI development in Java. The Budget Calculator Project developed using AWT (Abstract Window Toolkit) provides a comprehensive and user-friendly solution for managing personal finances. By leveraging Java's AWT library, the application offers a graphical user interface (GUI) that allows users to input, organize, and track their income and expenses effectively. This program supports various expense categories such as "Food," "Transport," "Utilities," and more, enabling users to categorize their expenses and monitor spending patterns. Additionally, it provides the functionality to calculate the remaining budget by subtracting total expenses from the entered income, offering users a clear and actionable view of their financial status. The program is designed with simplicity and accessibility in mind, making it suitable for both beginners and more experienced users. While the program currently supports basic budgeting functionalities, it also lays the groundwork for future enhancements. For instance, it could be expanded to include the ability to save or load budget data from files, track monthly expenses, or generate graphical representations of spending. These improvements could make the program even more powerful and useful for individuals with complex financial tracking needs.

4.2 FUTURE SCOPE

The future scope of the Budget Calculator Project can encompass several enhancements and additions to improve functionality, user experience, and performance. As financial management tools become increasingly integral in personal and business settings, expanding the capabilities of this simple budget calculator can make it more comprehensive, adaptive, and efficient. Below are some potential areas for future development:

Advanced Financial Reports and Analytics: Currently, the program allows for basic income-expense tracking. Future iterations could include more advanced analytics, such as graphical reports (pie charts, bar graphs) for visualizing spending patterns. This would allow users to identify trends, track monthly or yearly performance, and set and monitor financial goals more easily.

Budget Categories and Subcategories: The ability to create custom categories and subcategories for both income and expenses could provide more granularity and flexibility. Users could organize their budgets in ways that are more meaningful to them, e.g., splitting "Entertainment" into "Movies," "Dining," and "Travel."

Recurring Expenses: Adding functionality for users to track recurring expenses (such as monthly bills or subscriptions) would improve the tool's utility. A calendar-based system could allow users to set reminders for these recurring payments, helping to avoid missed payments and better manage cash flow.

Mobile App Development: While the current tool is designed for desktop use, there is a significant market for mobile applications. Developing a mobile version of the

Budget Calculator would allow users to track their budgets on the go, adding to the convenience and accessibility of the tool.

Machine Learning for Spending Predictions: By using machine learning algorithms, the tool could predict future expenses based on historical data, helping users anticipate upcoming costs and plan accordingly. These insights could assist in making better budgeting decisions and in setting more realistic financial goals. By integrating these features, the

APPENDIX A

(SOURCE CODE)

```
package encap1;

import java.awt.*;
import java.awt.event.*;

public class budcal extends Frame implements ActionListener {
    Label incomeLabel, expenseLabel, categoryLabel, resultLabel, titleLabel;
    TextField incomeField, expenseField, resultField;
    Choice categoryChoice;
    Button addButton, calculateButton, clearButton;
    TextArea expenseList;
    double totalIncome = 0.0;
    double totalExpenses = 0.0;

    public budcal() {

        setLayout(new BorderLayout());
        setTitle("Budget Calculator");

        titleLabel = new Label("Budget Calculator", Label.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        add(titleLabel, BorderLayout.NORTH);
```

```
Panel centerPanel = new Panel(new GridLayout(6, 2, 10, 10));
```



```
incomeLabel = new Label("Enter Income:");
```

```
incomeField = new TextField();
```

```
expenseLabel = new Label("Enter Expense Amount:");
```

```
expenseField = new TextField();
```

```
categoryLabel = new Label("Select Expense Category:");
```

```
categoryChoice = new Choice();
```

```
categoryChoice.add("Food");
```

```
categoryChoice.add("Transport");
```

```
categoryChoice.add("Utilities");
```

```
categoryChoice.add("Entertainment");
```

```
categoryChoice.add("Other");
```

```
resultLabel = new Label("Remaining Budget:");
```

```
resultField = new TextField();
```

```
resultField.setEditable(false);
```

```
centerPanel.add(incomeLabel);
```

```
centerPanel.add(incomeField);
```

```
centerPanel.add(expenseLabel);
```

```
centerPanel.add(expenseField);
```

```
centerPanel.add(categoryLabel);
```

```
centerPanel.add(categoryChoice);
```

```
centerPanel.add(resultLabel);
```

```
centerPanel.add(resultField);
```

```
add(centerPanel, BorderLayout.CENTER);
```

```

Panel buttonPanel = new Panel(new FlowLayout());
addButton = new Button("Add Expense");
calculateButton = new Button("Calculate Budget");
clearButton = new Button("Clear All");

buttonPanel.add(addButton);
buttonPanel.add(calculateButton);
buttonPanel.add(clearButton);

add(buttonPanel, BorderLayout.SOUTH);

Panel expensePanel = new Panel(new BorderLayout());
Label expenseListLabel = new Label("Expenses List:");
expenseList = new TextArea();
expenseList.setEditable(false);

expensePanel.add(expenseListLabel, BorderLayout.NORTH);
expensePanel.add(expenseList, BorderLayout.CENTER);

add(expensePanel, BorderLayout.EAST);

addButton.addActionListener(this);
calculateButton.addActionListener(this);
clearButton.addActionListener(this);

// Frame settings

```

```

setSize(600, 400);
setVisible(true);

// Window close handler
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        System.exit(0);
    }
});
}

@Override
public void actionPerformed(ActionEvent ae) {
    if (ae.getSource() == addButton) {
        try {
            double expense = Double.parseDouble(expenseField.getText());
            String category = categoryChoice.getSelectedItem();

            totalExpenses += expense;
            expenseList.append(category + ": " + expense + "\n");

            expenseField.setText("");
        } catch (NumberFormatException e) {
            expenseField.setText("Invalid Input");
        }
    } else if (ae.getSource() == calculateButton) {
        try {
            totalIncome = Double.parseDouble(incomeField.getText());
            double remaining = totalIncome - totalExpenses;

```

```

        resultField.setText(String.format("%.2f", remaining));
    } catch (NumberFormatException e) {
        resultField.setText("Invalid Income");
    }
} else if (ae.getSource() == clearButton) {
    totalIncome = 0.0;
    totalExpenses = 0.0;
    incomeField.setText("");
    expenseField.setText("");
    resultField.setText("");
    expenseList.setText("");
}
}

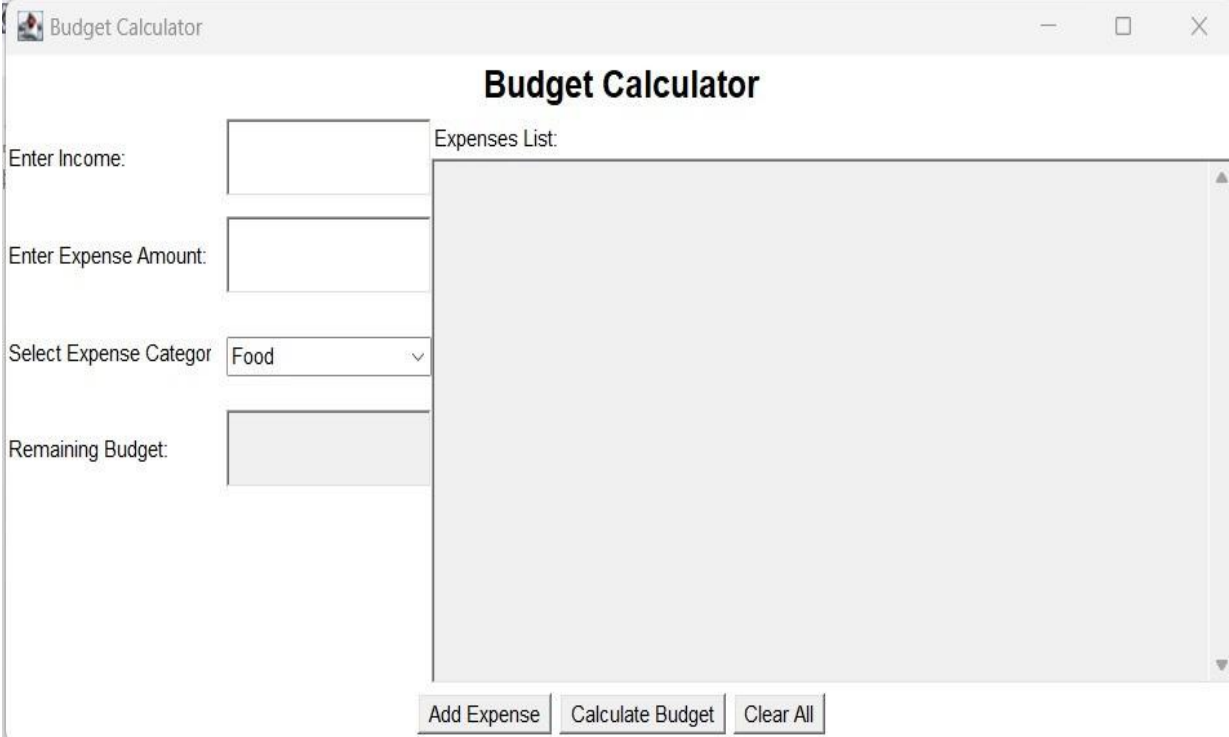
public static void main(String[] args) {
    new budcal();
}
}

```

APPENDIX B

(SCREENSHOTS)

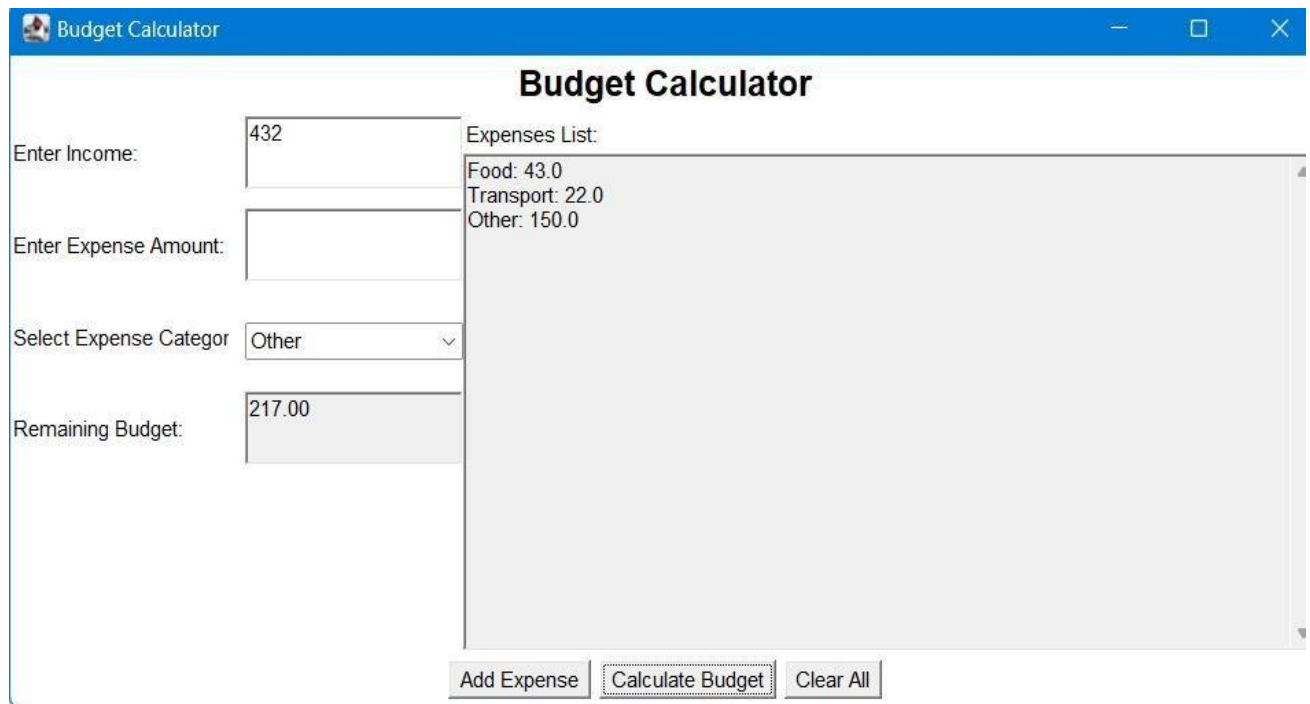
RESULT:



The screenshot shows a window titled "Budget Calculator" with a standard Windows-style title bar. The main content area is divided into two sections. On the left, there are four input fields with labels: "Enter Income:", "Enter Expense Amount:", "Select Expense Category" (with a dropdown menu showing "Food"), and "Remaining Budget:". On the right, there is a large, empty rectangular area labeled "Expenses List:". At the bottom of the window, there are three buttons: "Add Expense", "Calculate Budget", and "Clear All".

Fig 1. Output Part 1(Login Page)

RESULT:



The screenshot shows a Windows application window titled "Budget Calculator". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is divided into two columns. The left column contains input fields and labels: "Enter Income:" with a text box containing "432", "Enter Expense Amount:" with an empty text box, "Select Expense Category:" with a dropdown menu showing "Other", and "Remaining Budget:" with a text box containing "217.00". The right column contains an "Expenses List:" label above a large, empty list box. At the bottom of the window, there are three buttons: "Add Expense", "Calculate Budget", and "Clear All".

Label	Value
Enter Income:	432
Enter Expense Amount:	
Select Expense Category:	Other
Remaining Budget:	217.00

Expenses List:

- Food: 43.0
- Transport: 22.0
- Other: 150.0

Buttons: Add Expense, Calculate Budget, Clear All

Fig 2. Output Result

REFERENCES

1. How to create a simple calculator in Java using AWT - CodeSpeedy
2. Creating a Calculator using Java AWT - DEV Community
3. Budget Tracker using Java With Source Code - CodeWithCurious
4. YouTube Tutorial: How to Create Calculator in Java NetBeans Full Tutorial by DJ Oamen
5. YouTube Tutorial: Java Realtime Projects | Calculator And ATM App by Simplilearn - Search
6. Java AWT Tutorial - javatpoint
7. Java: A Step-by-Step Guide for Absolute Beginners by Daniel Bell - Aimed at newcomers, this book makes learning Java straightforward and teaches you how to navigate Eclipse for coding
8. Introduction to Programming in Java: An Interdisciplinary Approach (2nd Edition) by Robert Sedgewick - Although it is more general in terms of programming basics, it includes in-depth coverage of Java programming that can be practiced in Eclipse