

EXPENSE TRACKER APPLICATION



A PROJECT REPORT

Submitted by

HARSHITHA R (2303811710422062)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “ **EXPENSE TRACKER APPLICATION**” is the bonafide work of **HARSHITHA R (2303811710422062)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

CGB1201-JAVA PROGRAMMING
Mr. M. SARAVANAN, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

SIGNATURE

Mr. M. Saravanan, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 02.12.2024

CGB1201-JAVA PROGRAMMING
Mr. M. ARMANAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

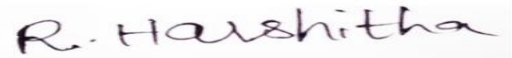
CGB1201-JAVA PROGRAMMING
Dr. R. SETHAMILSELVI, M.E., Ph.D.,
EXTERNAL EXAMINER
PROFESSOR
8138-SCE, TRICHY.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“EXPENSE TRACKER APPLICATION”** 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 **CGB1201 - JAVA PROGRAMMING**.

signature



HARSHITHA R

Place: Samayapuram

Date:02.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (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. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, 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. M. SARAVANAN, M.E.**, 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 serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.

Be an institute with world class research facilities

Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

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

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

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

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

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

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

Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

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.

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.

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

The Abstract of Expense tracker application implements a Graphical User Interface (GUI) Expense Tracker in Java using the Swing library. The application allows users to set a budget, track expenses by category (such as Food, Transportation, Entertainment, etc.), and monitor their total spending. Each expense is recorded with details like amount, category, and date, and stored in a list for easy management. Users can input a budget, add expenses, and check their budget status to see if they are within their limit or exceeding it. The program also includes a feature to view grouped expenses by category in a summarized format. Alerts are displayed if the user surpasses the budget. The interface consists of buttons, combo boxes, spinners, and labels, making it user-friendly. Additionally, the program includes safeguards like input validation and confirmation dialogs for a seamless and error-free experience. It is designed to be intuitive and efficient, offering a dynamic approach to financial tracking. The application holds significant potential for future enhancements, such as database integration, data visualization, mobile support, and advanced analytics, making it a scalable and adaptable solution for modern financial management needs. Overall, the application simplifies budget tracking and financial management in an intuitive way.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
The provided code implements a Graphical User Interface (GUI) Expense Tracker in Java using the Swing library. The application allows users to set a budget, track expenses by category (such as Food, Transportation, Entertainment, etc.), and monitor their total spending. Each expense is recorded with details like amount, category, and date, and stored in a list for easy management. Users can input a budget, add expenses, and check their budget status to see if they are within their limit or exceeding it. The program also includes a feature to view grouped expenses by category in a summarized format. Alerts are displayed if the user surpasses the budget. The interface consists of buttons, combo boxes, spinners, and labels, making it user-friendly. Additionally, the program includes safeguards like input validation and confirmation dialogs for a seamless and error-free experience. Overall, the application simplifies budget tracking and financial management in an intuitive way.	PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3	PSO1 -3 PSO2 -3 PSO3 -3

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

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
		v
	ABSTRACT	i
		i
		i
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	PROJECT METHODOLOGY	3
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	MODULE DESCRIPTION	4
	3.1 Expense management Module	4
	3.2 Budget Monitoring Module	4
	3.3 Graphical User Interface	4
	3.4 Event Handling Module	4
		4
4	CONCLUSION & FUTURE SCOPE	5
	4.1 Conclusion	5
	4.2 Future Scope	5
	REFERENCES	6
	APPENDIX A (SOURCE CODE)	7
		1
	APPENDIX B (SCREENSHOTS)	0

CHAPTER 1

INTRODUCTION

Objective

The objective of the program is to provide a user-friendly Expense Tracker application that helps users manage their finances effectively. It enables users to set a budget, record their daily expenses categorized by type, and track their total spending over time. The program alerts users if their expenses exceed the set budget and provides an organized summary of spending by category. By integrating essential features like date tracking, category selection, and budget status checks, the application aims to simplify financial monitoring and promote better budgeting habits in an intuitive graphical interface.

Overview

The Expense Tracker GUI is a Java-based application designed to help users manage their finances by setting budgets, recording expenses, categorizing them, and tracking spending in real-time. Built with Java Swing, it features an intuitive interface for adding expenses, checking budget status, and viewing categorized summaries. Using core Java concepts like OOP, data structures, and event handling, the program ensures efficiency and scalability, with future potential for enhancements like database integration and mobile support. It serves as a practical tool for promoting better financial management.

Java Programming Concepts

Basic concept of OOPS are:

- 1.Classes and Objects:** The Expense class is used to create objects that represent individual expenses, showcasing the fundamental OOP concept of working with classes and objects
- 2. Encapsulation:** The program groups related data and methods in the Expense

class, making attributes like amount, category, and date private and accessible only through public getter methods. This protects the data and ensures controlled access.

3. Abstraction: The program hides complex functionalities, such as storing and categorizing expenses, behind simple user actions like button clicks, allowing users to interact with the interface without worrying about the underlying logic.

4. Inheritance: Indirectly used through the Swing library, where classes like JFrame inherit from parent classes, providing reusable functionality for building the GUI.

5. Polymorphism: Different buttons perform unique actions using the same ActionListener interface, demonstrating polymorphism through dynamic behavior based on user input.

Project related Concepts:

1.Graphical User Interface (GUI):Java Swing is used to create an interactive interface, including buttons, labels, dropdowns, and input fields, enabling user-friendly interaction.

2.Data Storage and Organization:Expenses are stored as objects in an ArrayList, and HashMap is used for categorizing and summarizing expenses, ensuring efficient data handling.

3.Methods:

Getter Methods: Access private attributes of Expense class (amount, category, date).

Event Handlers: Handle user actions (e.g., adding expenses, setting budget).

Utility Methods: Used for date formatting (SimpleDateFormat) and dialog boxes (JOptionPane).

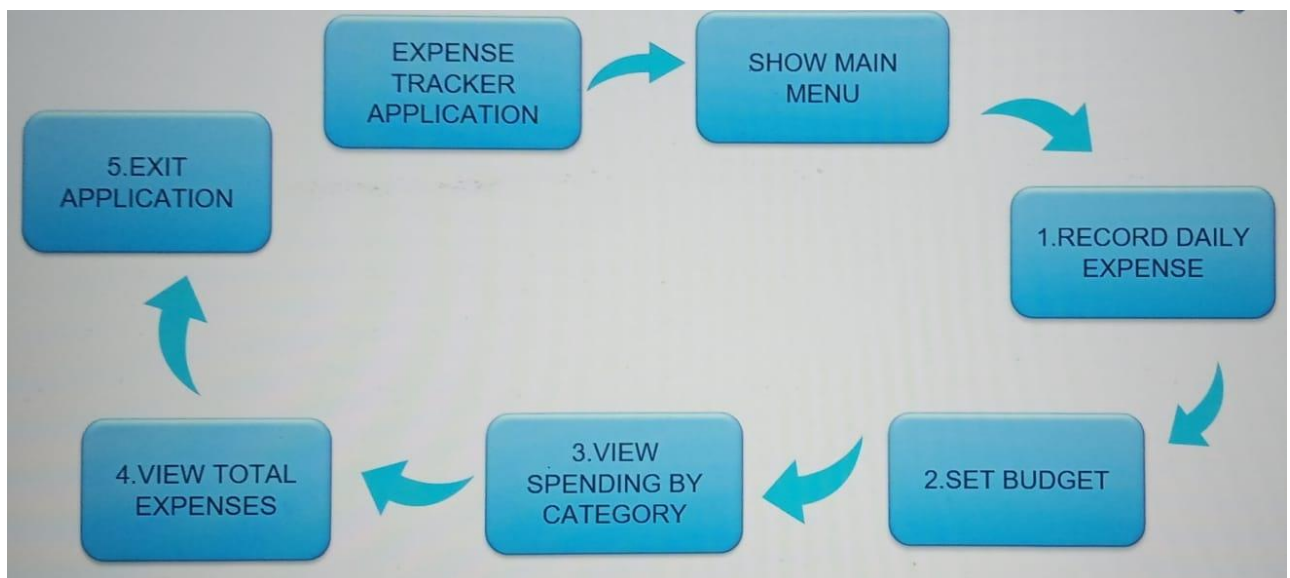
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the Expense Tracker GUI aims to enhance its functionality and user experience. Key improvements include implementing data persistence by integrating file storage or database support to retain expense records across sessions. Advanced features like recurring expense tracking, income management, and multi-currency support could make the application more versatile. Adding data visualization through graphs and charts would provide users with deeper insights into their spending patterns. Additionally, developing a mobile seamlessly across devices. Security enhancements, such as user authentication and data encryption, could ensure the privacy and safety of financial information. These upgrades would significantly improve the application's usability, scalability, and relevance for modern financial management needs.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Expense Management Module

This module focuses on the creation and management of expenses. It uses the Expense class to encapsulate expense details like amount, category, and date. These expense objects are stored in an ArrayList for easy retrieval and manipulation. The module ensures that expense data is well-organized and accessible for further processing, such as categorization and reporting.

3.2 Budget Monitoring Module

This module helps users manage their budgets effectively. Users can set a budget, and the module tracks total expenses against this budget. It dynamically updates the display and alerts users when they exceed the set budget. This module ensures that users remain informed about their financial status in real time.

3.3 Graphical user Interface(GUI)

This module is responsible for creating and managing the visual components of the application. Built using Java Swing, it includes interactive elements such as buttons (e.g., "Add Expense," "Set Budget"), labels (to display budget and expenses), dropdowns (for selecting categories), and input fields (e.g., date picker using JSpinner). The GUI provides a seamless and user-friendly platform for users to interact with the program.

3.4Event Handling Module:

This module manages the interaction between the user and the program. It captures user actions (e.g., button clicks) through ActionListener and triggers appropriate responses, such as adding expenses, setting budgets, or displaying summaries. This ensures the application remains responsive and dynamic.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In Conclusion the Expense Tracker GUI application provides an efficient and user-friendly platform for managing personal finances. By utilizing Java Swing for the graphical user interface and applying object-oriented programming principles, the program offers essential features such as expense tracking, budget monitoring, and spending analysis by category. The modular design ensures clarity and maintainability, while dynamic updates and validations enhance usability. The program demonstrates strong modularity through distinct components, such as expense management, budget monitoring, and reporting, ensuring maintainability and scalability. It validates user inputs to prevent errors and provides alerts when the budget is exceeded, helping users make informed financial decisions. Categorizing expenses into predefined groups further allows users to analyze their spending patterns effectively. Overall Project demonstrates the effective integration of programming concepts with practical functionality, making it a reliable tool for financial management

4.2 FUTURE SCOPE

The Expense Tracker GUI application has immense potential for future enhancements to improve its functionality and adaptability. Implementing data persistence through database or file storage would allow users to retain expense records across sessions. Visual analytics, such as graphs and dashboards, can provide deeper insights into spending patterns and budget usage. Developing a mobile-friendly version and integrating cloud synchronization would enhance accessibility and enable seamless cross-device usage. Security features like user authentication and data encryption could ensure the privacy of financial data. Additional functionalities such as recurring expense tracking make versatile

APPENDIX A

(SOURCE CODE)

```
import javax.swing.*;
import java.awt.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

public class ExpenseTrackerGUI {
    private double budget = 0.0;
    private double totalExpenses = 0.0;

    // Store expenses as objects
    private final ArrayList<Expense> expenseList = new ArrayList<>();

    public static void main(String[] args) {
        SwingUtilities.invokeLater(ExpenseTrackerGUI::new);
    }

    public ExpenseTrackerGUI() {
        // Create the main frame
        JFrame frame = new JFrame("Expense Tracker");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);
        frame.setLayout(new GridLayout(8, 1));
```



```

// Budget Label and Field
JLabel budgetLabel = new JLabel("Budget: $0.0");
frame.add(budgetLabel);

// Expense Label
JLabel expenseLabel = new JLabel("Total Expenses: $0.0");
frame.add(expenseLabel);

// Add Expense Panel
JPanel addExpensePanel = new JPanel(new GridLayout(1, 2));
JButton setBudgetButton = new JButton("Set Budget");
addExpensePanel.add(setBudgetButton);
JButton addExpenseButton = new JButton("Add Expense");
addExpensePanel.add(addExpenseButton);
frame.add(addExpensePanel);

// Category Selector
JLabel categoryLabel = new JLabel("Select Expense Category:");
frame.add(categoryLabel);

String[] categories = {"Food", "Transportation", "Entertainment", "Bills",
"Others"};

JComboBox<String> categoryComboBox = new
JComboBox<>(categories);
frame.add(categoryComboBox);

// Date Selector
JLabel dateLabel = new JLabel("Select Date:");
frame.add(dateLabel);

JSpinner dateSpinner = new JSpinner(new SpinnerDateModel());

```

```

    JSpinner.DateEditor dateEditor = new JSpinner.DateEditor(dateSpinner,
"yyyy-MM-dd");
    dateSpinner.setEditor(dateEditor);
    frame.add(dateSpinner);

    // Check Budget Status Button
    JButton checkStatusButton = new JButton("Check Budget Status");
    frame.add(checkStatusButton);

    // View Expenses by Category Button
    JButton viewCategoryButton = new JButton("View Expenses by
Category");
    frame.add(viewCategoryButton);

    // Exit Button
    JButton exitButton = new JButton("Exit");
    frame.add(exitButton);

    // Action Listeners
    setBudgetButton.addActionListener(e -> {
        String input = JOptionPane.showInputDialog(frame, "Enter budget
amount:");
        try {
            budget = Double.parseDouble(input);
            if (budget < 0) {
                JOptionPane.showMessageDialog(frame, "Budget cannot be
negative.", "Error", JOptionPane.ERROR_MESSAGE);
            } else {
                budgetLabel.setText("Budget: " + budget);
            }
        }
    });

```

```

    }
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(frame, "Invalid input. Please enter
a number.", "Error", JOptionPane.ERROR_MESSAGE);
    }
});

addExpenseButton.addActionListener(e -> {
    String input = JOptionPane.showInputDialog(frame, "Enter expense
amount:");
    try {
        double expense = Double.parseDouble(input);
        if (expense < 0) {
            JOptionPane.showMessageDialog(frame, "Expense cannot be
negative.", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        String category = (String) categoryComboBox.getSelectedItem();
        Date date = (Date) dateSpinner.getValue();

        totalExpenses += expense;
        expenseList.add(new Expense(expense, category, date));
        expenseLabel.setText("Total Expenses: " + totalExpenses);

        // Check if the user has exceeded the budget
        if (totalExpenses > budget) {
            JOptionPane.showMessageDialog(frame, "Warning: You have
exceeded your budget!", "Alert", JOptionPane.WARNING_MESSAGE);

```

```

    }
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(frame, "Invalid input. Please enter
a number.", "Error", JOptionPane.ERROR_MESSAGE);
    }
});

```

```

checkStatusButton.addActionListener(e -> {
    if (totalExpenses > budget) {
        JOptionPane.showMessageDialog(frame, "You are over budget by:
quot; + (totalExpenses - budget), "Status",
JOptionPane.INFORMATION_MESSAGE);
    } else {
        JOptionPane.showMessageDialog(frame, "You are within budget.
Remaining: quot; + (budget - totalExpenses), "Status",
JOptionPane.INFORMATION_MESSAGE);
    }
});

```

```

viewCategoryButton.addActionListener(e -> {
    // Group expenses by category
    Map<String, Double> categoryTotals = new HashMap<>();
    for (Expense expense : expenseList) {
        categoryTotals.put(expense.getCategory(),
            categoryTotals.getDefault(expense.getCategory(), 0.0) +
expense.getAmount());
    }
}

```

```

// Build a string to display the expenses by category

```

```

        StringBuilder message = new StringBuilder("Expenses by
Category:\n");
        for (Map.Entry<String, Double> entry : categoryTotals.entrySet()) {
            message.append(entry.getKey()).append(":
            quot;).append(entry.getValue()).append("\n");
        }

        // Show the message in a dialog
        JOptionPane.showMessageDialog(frame, message.toString(),
"Expenses by Category", JOptionPane.INFORMATION_MESSAGE);
    });

    exitButton.addActionListener(e -> {
        int confirm = JOptionPane.showConfirmDialog(frame, "Are you sure
you want to exit?", "Confirm Exit", JOptionPane.YES_NO_OPTION);
        if (confirm == JOptionPane.YES_OPTION) {
            System.exit(0);
        }
    });

    // Display the frame
    frame.setVisible(true);
}

// Expense class to store details
static class Expense {
    private final double amount;
    private final String category;
    private final Date date;

```

```

public Expense(double amount, String category, Date date) {
    this.amount = amount;
    this.category = category;
    this.date = date;
}

public double getAmount() {
    return amount;
}

public String getCategory() {
    return category;
}

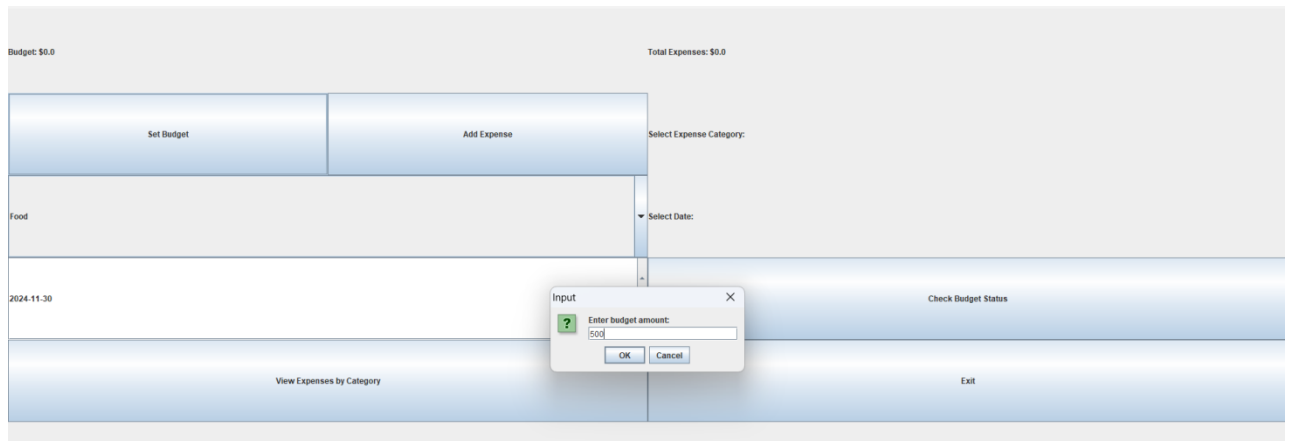
public Date getDate() {
    return date;
}

@Override
public String toString() {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    return "Amount: " + amount + ", Category: " + category + ", Date: "
+ sdf.format(date);
}
}

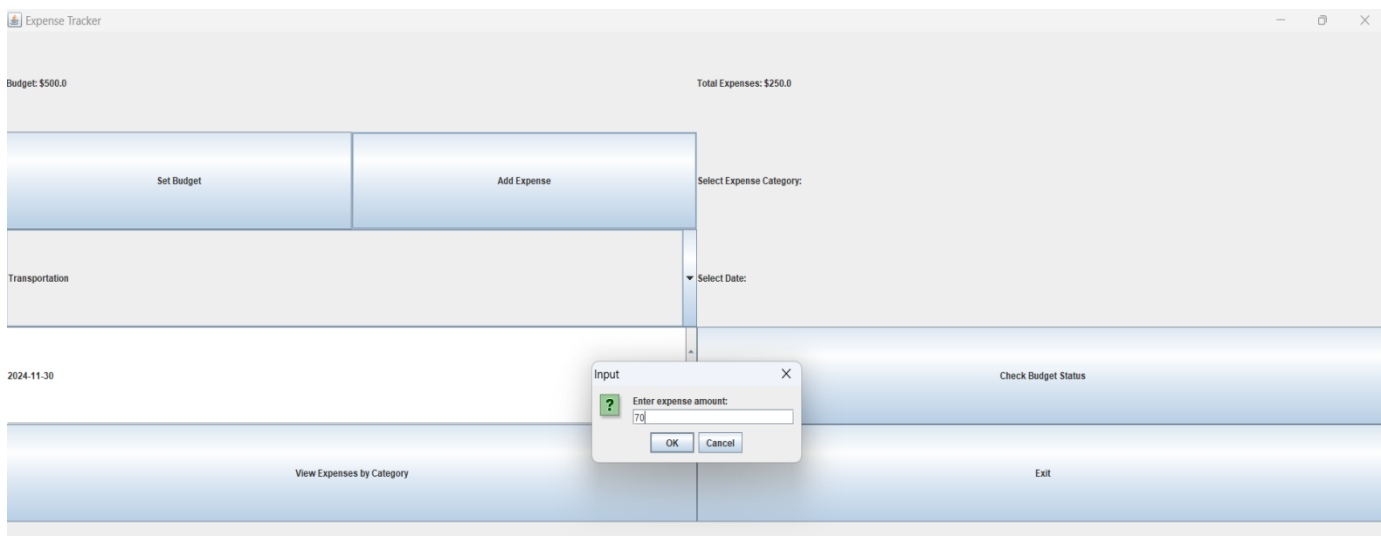
```

APPENDIX B (SCREENSHOTS)

SET BUDGET



2.ADD EXPENSE



3.VIEW EXPENSE BY CATEGORY

Budget: \$500.0

Total Expenses: \$320.0

Set Budget

Add Expense

Transportation

Select Expense Category:

2024-11-30

Select Date:

View Expenses by Category

Check Budget Status

Exit

Expenses by Category

Expenses by Category:
Transportation: \$70.0
Food: \$250.0

OK

4.CHECK BUDGET STATUS

Budget: \$500.0

Total Expenses: \$320.0

Set Budget

Add Expense

Transportation

Select Expense Category:

2024-11-30

Select Date:

View Expenses by Category

Check Budget Status

Exit

Status

You are within budget. Remaining: \$180.0

OK

REFERENCES

JAVA BOOKS:

1.Core Java Volume I: Fundamentals" by Cay S. Horstmann

A classic book for Java developers that dives deep into Java fundamentals, including object-oriented programming, and provides clear examples and explanations.

2.Java Swing" by Robert Eckstein, Marc Loy, and Dave Wood

A focused book on building user interfaces with Java Swing, perfect for those looking to create graphical applications.

REFERENCE FROM WEBSITES:

1.GeeksforGeeks - Java Programming Language

GeeksforGeeks offers tutorials, articles, and examples on various Java topics, from basic syntax to advanced concepts like event handling and Java Swing.

<https://www.geeksforgeeks.org/java/>

2.TutorialsPoint - Java Programming

TutorialsPoint is another great resource for learning Java, with detailed explanations, examples, and a section specifically for Java Swing and GUI programming.

<https://www.tutorialspoint.com/java/>

REFERENCE FROM YOUTUBE LINKS:

1.Java Swing Tutorial by CodeAcademy

A tutorial focused on Java Swing to help you learn how to build graphical user interfaces (GUIs) using Swing.

Java Swing GUI Tutoria

URL:<https://www.youtube.com/users/codeAcademy>