EXPENSE TRACKER SYSTEM

A MINI PROJECT REPORT

Submitted by

SHIYAM(231001193)

SUMANTH S(231001224)

VINOTH M (231001246)

in partial fulfilment for the course

CS23333-OBJECT ORIENTED PROGRAMMING USING JAVA

for the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project report titled "**EXPENSE TRACKER SYSTEM**" is the bonafide work of **SHIYAM (231001193), SUMANTH (231001224), VINOTH (231001246)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mr Narayanan KE	**Dr.P.VALARMATHIE**
Assistant Professor	Head of The Department
Department of Information Technology	Department of Information Technology
Rajalakshmi Engineering College	Rajalakshmi Engineering College

Submitted to Project Viva Voce Examination for the course CS23333 Object Oriented Programming using Java held on________

Internal Examiner **External Examiner**

ABSTRACT

The Expense Tracker Application is an automated software solution designed to facilitate effective financial management for individuals and businesses. The system aims to enhance budgeting, expense tracking, and overall financial awareness, ensuring that users can monitor their financial activities with ease and precision.The primary features of the application include maintaining a comprehensive record of expenses, categorizing spending by type (such as income, groceries, entertainment, etc.), and generating detailed reports on financial performance. Users can effortlessly add, update, or remove transactions, capturing essential details such as date, category, description, and amount spent.Additionally, the application provides a user-friendly interface for visualizing financial data through charts and graphs, enabling users to identify spending trends and make informed decisions. Budgeting tools allow users to set spending limits for various categories, helping them stay within their financial goals. Alerts and reminders can be configured to notify users of upcoming bills or budget thresholds

TABLE OF CONTENTS

INTRODUCTION

1.1 INTRODUCTION

Introduction to the Expense Tracker Application
In an increasingly complex financial landscape, managing personal finances has become more crucial than ever. The Expense Tracker Application is designed to empower individuals and small businesses by providing a comprehensive tool for monitoring and managing their financial activities. This innovative application simplifies the process of tracking expenses, budgeting, and analyzing spending habits, enabling users to gain greater control over their financial well-being.

1.1 IMPLEMENTATION

The **Expense Tracker** project discussed here is implemented using the concepts of **JAVA SWING** and **MYSQL**.

1.2 SCOPE OF THE PROJECT

The Expense Tracker application enables users to manage their personal finances by adding, updating, and categorizing transactions as income or expenses through a user-friendly Java interface. It integrates with a database for efficient data storage and retrieval, facilitating monthly expense calculations and financial summaries.

1.3 WEBSITE FEATURES

- Add, update, and remove transactions.
- Categorize transactions as income or expenses.
- Calculate and display monthly income and expenses.
- User-friendly graphical interface using Java Swing.
- Database integration for secure data storage and retrieval.
- Display total balance, total income, and total expenses.
- Customizable transaction descriptions.
- Confirmation dialogs for critical actions (e.g., deletion and logout).

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR	:	Intel i7
MEMORY SIZE	:	24GB
HARD DISK	:	500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE	:	Java, MySQL
FRONT-END	:	Java
BACK-END	:	MySQL
OPERATING SYSTEM	:	Windows 11

SAMPLE CODE

3.1 HOME PAGE DESIGN

```
 import ExpenseTracker.src.Login;
 import java.awt.*;
 import java.awt.event.*;
 import java.text.ParseException;
 import java.text.SimpleDateFormat;
 import java.util.Calendar;
 import java.util.Date;
 import javax.swing.*;
 import javax.swing.table.DefaultTableCellRenderer;
 import javax.swing.table.DefaultTableModel;
 import java.sql.Connection;
 import java.sql.PreparedStatement;
 import java.sql.ResultSet;
 import java.sql.SQLException;
 import java.sql.Statement;

public ExpenseAndIncomeTrackerApp()
        { frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 500);
        frame.setLocationRelativeTo(null);
        frame.setUndecorated(true);
        frame.getRootPane().setBorder(BorderFactory.createMatteBorder(5, 5, 5, 5, new Color(52, 73, 94)));

        titleBar = new JPanel(); titleBar.setLayout(null);
        titleBar.setBackground(new Color(52, 73, 94));
        titleBar.setPreferredSize(new Dimension(frame.getWidth(), 30));
        frame.add(titleBar, BorderLayout.NORTH);

        titleLabel = new JLabel("AURA-THE EXPENSE TRACKER");
        titleLabel.setForeground(Color.WHITE);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 15));
        titleLabel.setBounds(10, 0, 250, 30);
        titleBar.add(titleLabel);

        closeLabel = new JLabel("x");
        closeLabel.setForeground(Color.WHITE);
        closeLabel.setFont(new Font("Arial", Font.BOLD, 17));
        closeLabel.setHorizontalAlignment(SwingConstants.CENTER);
        closeLabel.setBounds(frame.getWidth() - 50, 0, 30, 30);
        closeLabel.setCursor(new Cursor(Cursor.HAND_CURSOR));

        closeLabel.addMouseListener(new MouseAdapter()
            { @Override
            public void mouseClicked(MouseEvent e)
                { System.exit(0);
            }

            @Override
```

```
    public void mouseEntered(MouseEvent e)
        { closeLabel.setForeground(Color.red);
    }

    @Override
    public void mouseExited(MouseEvent e)
        { closeLabel.setForeground(Color.white);
    }
});

titleBar.add(closeLabel);

minimizeLabel = new JLabel("-");
minimizeLabel.setForeground(Color.WHITE);
minimizeLabel.setFont(new Font("Arial", Font.BOLD, 17));
minimizeLabel.setHorizontalAlignment(SwingConstants.CENTER);
minimizeLabel.setBounds(frame.getWidth() - 80, 0, 30, 30);
minimizeLabel.setCursor(new Cursor(Cursor.HAND_CURSOR));

minimizeLabel.addMouseListener(new MouseAdapter()
    { @Override
    public void mouseClicked(MouseEvent e)
        { frame.setState(JFrame.ICONIFIED);
    }

    @Override
    public void mouseEntered(MouseEvent e)
        { minimizeLabel.setForeground(Color.yellow);
    }

    @Override
    public void mouseExited(MouseEvent e)
        { minimizeLabel.setForeground(Color.white);
    }
});

titleBar.add(minimizeLabel);

titleBar.addMouseListener(new MouseAdapter() {
    @Override
    public void mousePressed(MouseEvent e)
        { isDragging = true;
        mouseOffset = e.getPoint();
    }

    @Override
    public void mouseReleased(MouseEvent e)
        { isDragging = false;
    }
});

titleBar.addMouseMotionListener(new MouseAdapter()
    { @Override
    public void mouseDragged(MouseEvent e) {
        if (isDragging) {
            Point newLocation = e.getLocationOnScreen();
            newLocation.translate(-mouseOffset.x, -mouseOffset.y);
            frame.setLocation(newLocation);
        }
    }
});

dashboardPanel = new JPanel();
dashboardPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 20));
dashboardPanel.setBackground(new Color(211, 211,        211));
frame.add(dashboardPanel, BorderLayout.CENTER);

buttonsPanel = new JPanel();
buttonsPanel.setLayout(new GridLayout(5, 1, 5, 5));

addTransactionButton = new JButton("Add Transaction");
addTransactionButton.setBackground(new Color(41, 128, 150));
addTransactionButton.setForeground(Color.WHITE);
addTransactionButton.setFocusPainted(false);
addTransactionButton.setBorderPainted(false);
addTransactionButton.setFont(new Font("Arial", Font.BOLD, 14));
addTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

removeTransactionButton = new JButton("Remove Transaction");
removeTransactionButton.setBackground(new Color(231, 76, 60));
removeTransactionButton.setForeground(Color.WHITE);
removeTransactionButton.setFocusPainted(false);
removeTransactionButton.setBorderPainted(false);
removeTransactionButton.setFont(new Font("Arial", Font.BOLD, 14));
removeTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

updateTransactionButton = new JButton("Update Transaction");
updateTransactionButton.setBackground(new Color(255, 192, 203)); // Pink color
updateTransactionButton.setForeground(Color.WHITE);
updateTransactionButton.setFocusPainted(false);
updateTransactionButton.setBorderPainted(false);
updateTransactionButton.setFont(new Font("Arial", Font.BOLD, 14));
updateTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

calculateMonthlyButton = new JButton("Calculate Monthly Expense");
calculateMonthlyButton.setBackground(new Color(46, 204, 113));
calculateMonthlyButton.setForeground(Color.WHITE);
calculateMonthlyButton.setFocusPainted(false);
calculateMonthlyButton.setBorderPainted(false);
calculateMonthlyButton.setFont(new Font("Arial", Font.BOLD, 14));
calculateMonthlyButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

logoutButton = new JButton("Logout");
logoutButton.setBackground(new Color(128, 0, 128));
logoutButton.setForeground(Color.WHITE);
logoutButton.setFocusPainted(false);
logoutButton.setBorderPainted(true);
logoutButton.setBorder(BorderFactory.createLineBorder(Color.white,   2));
```

```
    thickness
    logoutButton.setFont(new Font("Arial", Font.BOLD, 14));
    logoutButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
```

3.2 LOGIN PAGE DESIGN:

```
package login_page;

import javax.swing.*; import
java.awt.*;
import java.awt.event.ActionEvent; import
java.awt.event.ActionListener;
import login_page_back_end.loginPageBE;

public class Login {
  private JFrame loginFrame;
  private JTextField usernameField;
  private JPasswordField passwordField;
  private final String CORRECT_USERNAME = "system"; // Predefined username
  private final String CORRECT_PASSWORD = "rec"; // Predefined password

  public Login() {
    // Create and set up the login window
    loginFrame = new JFrame("AURA - Login");
    loginFrame.setSize(400, 300);
    loginFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    loginFrame.setLocationRelativeTo(null);
    loginFrame.setUndecorated(true); // Remove window decorations
    loginFrame.getRootPane().setBorder(BorderFactory.createMatteBorder(2, 2, 2, 2, new Color(52, 73, 94)));

    // Create main panel with a nice background color
    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(null);
    mainPanel.setBackground(new Color(211, 211, 211));

    // Create title label
    JLabel titleLabel = new JLabel("AURA EXPENSE TRACKER");
    titleLabel.setFont(new Font("Arial", Font.BOLD, 16));
    titleLabel.setBounds(90, 30, 250, 30);
    titleLabel.setForeground(new Color(52, 73, 94));

    // Create username label and field
    JLabel usernameLabel = new JLabel("Username:");
    usernameLabel.setBounds(50, 90, 80, 25);
    usernameField = new JTextField();
    usernameField.setBounds(140, 90, 200, 25);
```

```
// Create password label and field
JLabel passwordLabel = new JLabel("Password:");
passwordLabel.setBounds(50, 130, 80, 25);
passwordField = new JPasswordField();
passwordField.setBounds(140, 130, 200, 25);

// Create login button
JButton loginButton = new JButton("Login");
loginButton.setBounds(140, 180, 100, 30);
loginButton.setBackground(new Color(41, 128, 185));
loginButton.setForeground(Color.WHITE);
loginButton.setFocusPainted(false);
loginButton.setBorderPainted(false);
loginButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

// Create exit button
JButton exitButton = new JButton("Exit");
exitButton.setBounds(250, 180, 90, 30);
exitButton.setBackground(new Color(231, 80, 60));
exitButton.setForeground(Color.WHITE);
exitButton.setFocusPainted(false);
exitButton.setBorderPainted(false);
exitButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

// Add action listener to login button
loginButton.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e)
    { String username =
    usernameField.getText();
    String password = new String(passwordField.getPassword());

    if (username.equals(CORRECT_USERNAME) && password.equals(CORRECT_PASSWORD))
      { loginFrame.dispose(); // Close login window
      // Start the main application
      SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
          new ExpenseAndIncomeTrackerApp();
        }
      });
    } else {
      JOptionPane.showMessageDialog(loginFrame,
          "Invalid username or password!",
          "Login Error",
          JOptionPane.ERROR_MESSAGE);
      // Clear the password field
```

```
passwordField.setText("");
```

```
            }
        }
    });

    // Add action listener to exit button
    exitButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e)
            { System.exit(0);
        }
    });

    // Add components to the panel
    mainPanel.add(titleLabel);
    mainPanel.add(usernameLabel);
    mainPanel.add(usernameField);
    mainPanel.add(passwordLabel);
    mainPanel.add(passwordField);
    mainPanel.add(loginButton);
    mainPanel.add(exitButton);

    // Add panel to frame
    loginFrame.add(mainPanel);
    loginFrame.setVisible(true);
}
```

3.3 DASHBOARD DESIGN

```
dashboardPanel = new JPanel();
    dashboardPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 20));
    dashboardPanel.setBackground(new Color(211, 211,        211));
    frame.add(dashboardPanel, BorderLayout.CENTER);

    // Create and set up buttons panel
    buttonsPanel = new JPanel();
    buttonsPanel.setLayout(new GridLayout(5, 1, 5, 5)); // Change to 5 rows to accommodate the new button

    addTransactionButton = new JButton("Add Transaction");
    addTransactionButton.setBackground(new Color(41, 128, 150));
    addTransactionButton.setForeground(Color.WHITE);
    addTransactionButton.setFocusPainted(false);
    addTransactionButton.setBorderPainted(false);
    addTransactionButton.setFont(new Font("Arial", Font.BOLD, 14));
    addTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

    removeTransactionButton = new JButton("Remove Transaction");
    removeTransactionButton.setBackground(new Color(231, 76, 60));
    removeTransactionButton.setForeground(Color.WHITE);
    removeTransactionButton.setFocusPainted(false);
    removeTransactionButton.setBorderPainted(false);
    removeTransactionButton.setFont(new Font("Arial", Font.BOLD, 14));
```

```
        removeTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

        // Create the Update Transaction button
        updateTransactionButton = new JButton("Update Transaction");
        updateTransactionButton.setBackground(new Color(255, 192, 203)); // Pink color
        updateTransactionButton.setForeground(Color.WHITE);
        updateTransactionButton.setFocusPainted(false);
        updateTransactionButton.setBorderPainted(false);
        updateTransactionButton.setFont(new Font("Arial", Font.BOLD, 14));
        updateTransactionButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

        calculateMonthlyButton = new JButton("Calculate Monthly Expense");
        calculateMonthlyButton.setBackground(new Color(46, 204, 113));
        calculateMonthlyButton.setForeground(Color.WHITE);
        calculateMonthlyButton.setFocusPainted(false);
        calculateMonthlyButton.setBorderPainted(false);
        calculateMonthlyButton.setFont(new Font("Arial", Font.BOLD, 14));
        calculateMonthlyButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

        logoutButton = new JButton("Logout");
        logoutButton.setBackground(new Color(128, 0, 128)); // Purple
        logoutButton.setForeground(Color.WHITE); // White text
        logoutButton.setFocusPainted(false);
        logoutButton.setBorderPainted(true); // Enable border painting
        logoutButton.setBorder(BorderFactory.createLineBorder(Color.white, 2)); // Set border color and
thickness
        logoutButton.setFont(new Font("Arial", Font.BOLD, 14));
        logoutButton.setCursor(new Cursor(Cursor.HAND_CURSOR));

        // Add buttons to the panel
        buttonsPanel.add(addTransactionButton);
        buttonsPanel.add(removeTransactionButton);
        buttonsPanel.add(updateTransactionButton); // Add Update button here
        buttonsPanel.add(calculateMonthlyButton);
        buttonsPanel.add(logoutButton);
        dashboardPanel.add(buttonsPanel);
        String[] columnNames = {"ID", "Date", "Type", "Description", "Amount"};
        tableModel = new DefaultTableModel(columnNames, 0) {
            @Override
            public boolean isCellEditable(int row, int column)
                { return false;
            }
        };

        // Create the transaction table
        transactionTable = new JTable(tableModel);
        transactionTable.setBackground(Color.WHITE);
        transactionTable.setGridColor(new Color(200, 200, 200));
        transactionTable.setRowHeight(25);
        transactionTable.getTableHeader().setBackground(new Color(52, 73, 94));
        transactionTable.getTableHeader().setForeground(Color.WHITE);
        transactionTable.getTableHeader().setFont(new Font("Arial", Font.BOLD, 12));

        // Add custom cell renderer for coloring based on type
        transactionTable.setDefaultRenderer(Object.class, new DefaultTableCellRenderer()
            { @Override
            public Component getTableCellRendererComponent(JTable table, Object value,
                    boolean isSelected, boolean hasFocus, int row, int column) {
                Component c = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row,
column);

                String type = (String) table.getModel().getValueAt(row, 2); // Column 2 contains the type

                if (type.equals("Expense")) {
                    c.setForeground(new Color(231, 76, 60)); // Red color for expense
                } else if (type.equals("Income")) {
                    c.setForeground(new Color(46, 204, 113)); // Green color for income
                } else {
                    c.setForeground(Color.BLACK); // Default color
                }

                // Maintain selection highlighting
                if (isSelected) {
                    c.setBackground(table.getSelectionBackground());
                } else {
                    c.setBackground(table.getBackground());
                }

                return c;
            }
        });

        // Create scroll pane for the table
        JScrollPane scrollPane = new JScrollPane(transactionTable);
        scrollPane.setPreferredSize(new Dimension(500, 300));
        dashboardPanel.add(scrollPane);

        // Create total amount panel
        JPanel totalPanel = new JPanel();
        totalPanel.setLayout(new GridLayout(3, 1)); // Change to GridLayout with 3 rows
        totalPanel.setBackground(new Color(211, 211, 211));

        // Total Balance Label
        totalLabel = new JLabel("Total Balance: ₹" + String.format("%.2f", totalAmount));
        totalLabel.setFont(new Font("Arial", Font.BOLD, 16));
        totalLabel.setForeground(new Color(52, 73, 94));
        totalPanel.add(totalLabel);

        // Total Income Label
        incomeLabel = new JLabel("Total Income: ₹0.00");
        incomeLabel.setFont(new Font("Arial", Font.BOLD, 16));
        incomeLabel.setForeground(new Color(46, 204, 113)); // Green color for income
        totalPanel.add(incomeLabel);

        // Total Expense Label
        expenseLabel = new JLabel("Total Expense: ₹0.00");
        expenseLabel.setFont(new Font("Arial", Font.BOLD, 16));
        expenseLabel.setForeground(new Color(231, 76, 60)); // Red color for expense
```

```
    totalPanel.add(expenseLabel);

    dashboardPanel.add(totalPanel);

    // Add action listeners for buttons
    addTransactionButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e)
            { showAddTransactionDialog();
        }
    });
    updateTransactionButton.addActionListener(new ActionListener()
        { @Override
        public void actionPerformed(ActionEvent e)
            { showUpdateTransactionDialog();
        }
    });

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

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

    logoutButton.addActionListener(new ActionListener()
        { @Override
        public void actionPerformed(ActionEvent e) {
            int choice = JOptionPane.showConfirmDialog(frame,
                    "Are you sure you want to logout?",
                    "Logout Confirmation",
                    JOptionPane.YES_NO_OPTION);

            if (choice == JOptionPane.YES_OPTION)
                { frame.dispose(); // Close the current window
                new Login(); // Open the Login window
            }
        }
    });

    // Make the frame visible
    frame.setVisible(true);
}
```

3.4 REMOVE OPERATION IN DASHBOARD

```
package remove_student;

import dash_board.dashBoard; import
table_order.tableOrder;

import java.awt.*;
import java.awt.event.ActionEvent; import
java.awt.event.ActionListener; import
java.sql.Connection;
import java.sql.DriverManager; import
java.sql.PreparedStatement;

import javax.swing.*;
private void removeSelectedTransaction() {
    int selectedRow = transactionTable.getSelectedRow();
    if (selectedRow != -1) {
        // Confirm deletion
        int choice = JOptionPane.showConfirmDialog(frame,
                "Are you sure you want to delete this transaction?",
                "Confirm Deletion",
                JOptionPane.YES_NO_OPTION);

        if (choice != JOptionPane.YES_OPTION)
            { return; // User chose not to delete
        }

        // Get the transaction details
        int transactionId = (int) tableModel.getValueAt(selectedRow, 0); // Assuming ID is in column 0
        String type = (String) tableModel.getValueAt(selectedRow, 2); // Assuming Type is in column 2
        String description = (String) tableModel.getValueAt(selectedRow, 3); // Assuming Description is in
        column 3
        double amount = Double.parseDouble((String) tableModel.getValueAt(selectedRow, 4)); // Assuming
        Amount is in column 4

        // Remove from the main transaction table
        try (Connection connection = DataBaseConnector.getConnection();
             PreparedStatement psMain = connection.prepareStatement(
                     "DELETE FROM transaction_table WHERE ID = ?"))
            { psMain.setInt(1, transactionId);
            psMain.executeUpdate();
            System.out.println("Transaction removed successfully from the main table.");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(frame, "Error removing transaction from main table: " +
e.getMessage(),
                    "Database Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Remove from the category-specific table
        String categoryTableName = description.toLowerCase() + "_transactions"; // Construct table name
        try (Connection connection = DataBaseConnector.getConnection();
             PreparedStatement psCategory = connection.prepareStatement(
                "DELETE FROM " + categoryTableName + " WHERE ID = ?"))
        { psCategory.setInt(1, transactionId); // Use the same ID
        int rowsAffected = psCategory.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Transaction removed successfully from " + categoryTableName);
        } else {
            JOptionPane.showMessageDialog(frame, "No transaction found in " + categoryTableName + "
        with ID: " + transactionId,
                    "Deletion Warning", JOptionPane.WARNING_MESSAGE);
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(frame, "Error removing transaction from category table: " +
      e.getMessage(),
                "Database Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    // Update the total amount based on transaction type
    if (type.equals("Income")) {
        totalAmount -= amount; // Subtract income amount from total
        double currentIncome = Double.parseDouble(incomeLabel.getText().replace("Total Income: ₹",
      ""));
        currentIncome -= amount; // Subtract from total income
        incomeLabel.setText("Total Income: ₹" + String.format("%.2f", currentIncome));
    } else { // Expense
        totalAmount += amount; // Add expense amount back to total
        double currentExpense = Double.parseDouble(expenseLabel.getText().replace("Total Expense: ₹",
      ""));
        currentExpense -= amount; // Subtract from total expense
        expenseLabel.setText("Total Expense: ₹" + String.format("%.2f", currentExpense));
    }

    // Update the total balance label
    totalLabel.setText("Total Balance: ₹" + String.format("%.2f", totalAmount));

    // Remove the row from the table
    tableModel.removeRow(selectedRow);

    // Inform the user that the transaction was successfully removed
    JOptionPane.showMessageDialog(frame, "Transaction successfully removed.", "Success",
    JOptionPane.INFORMATION_MESSAGE);
  } else {
    JOptionPane.showMessageDialog(frame, "Please select a transaction to remove!",
        "No Selection", JOptionPane.WARNING_MESSAGE);
  }
}
private void showAddTransactionDialog() {
  JDialog dialog = new JDialog(frame, "Add Transaction", true);
  dialog.setLayout(new GridLayout(5, 2, 10, 10));
  dialog.setSize(400, 250);
  dialog.setLocationRelativeTo(frame);

  String[] types = {"Expense", "Income"};
  JComboBox<String> typeCombo = new JComboBox<>(types);
```

```
// JComboBox for predefined descriptions
String[] descriptions = {"Medical", "Shopping", "Entertainment", "Transport", "Grocery", "Others"};
JComboBox<String> descriptionCombo = new JComboBox<>(descriptions);

JTextField amountField = new JTextField();

// Create date spinners
Calendar now = Calendar.getInstance();
JSpinner yearSpinner = new JSpinner(new SpinnerNumberModel(now.get(Calendar.YEAR), 1900, 2100,
1));
JSpinner monthSpinner = new JSpinner(new SpinnerNumberModel(now.get(Calendar.MONTH) + 1, 1,
12, 1));
JSpinner daySpinner = new JSpinner(new SpinnerNumberModel(now.get(Calendar.DAY_OF_MONTH),
  1, 31, 1));

JSpinner.NumberEditor yearEditor = new JSpinner.NumberEditor(yearSpinner, "#");
yearSpinner.setEditor(yearEditor);

Dimension spinnerSize = new Dimension(60, 25);
yearSpinner.setPreferredSize(spinnerSize);
monthSpinner.setPreferredSize(new Dimension(45, 25));
daySpinner.setPreferredSize(new Dimension(45, 25));

JPanel datePanel = new JPanel();
datePanel.add(yearSpinner);
datePanel.add(new JLabel("-"));
datePanel.add(monthSpinner);
datePanel.add(new JLabel("-"));
datePanel.add(daySpinner);

JButton submitButton = new JButton("Add");

dialog.add(new JLabel("Date:"));
dialog.add(datePanel);
dialog.add(new JLabel("Type:"));
dialog.add(typeCombo);
dialog.add(new JLabel("Description:"));
dialog.add(descriptionCombo);
dialog.add(new JLabel("Amount:"));
dialog.add(amountField);
dialog.add(new JLabel(""));
dialog.add(submitButton);

submitButton.addActionListener(new ActionListener()
   { @Override
   public void actionPerformed(ActionEvent e)
      { try {
         // Validate amount input
         if (amountField.getText().trim().isEmpty())
            { JOptionPane.showMessageDialog(dialog, "Please enter an amount!");
            return;
         }

         double amount = Double.parseDouble(amountField.getText());
```

```
// Validate description selection
String description = (String) descriptionCombo.getSelectedItem();
if (description == null || description.trim().isEmpty()) {
   JOptionPane.showMessageDialog(dialog, "Please select a description!");
   return;
}

// Get selected date
Calendar calendar = Calendar.getInstance();
calendar.set(Calendar.YEAR, (Integer) yearSpinner.getValue());
calendar.set(Calendar.MONTH, (Integer) monthSpinner.getValue() - 1);
calendar.set(Calendar.DAY_OF_MONTH, (Integer) daySpinner.getValue());
Date selectedDate = calendar.getTime();

// Get transaction type
String type = (String) typeCombo.getSelectedItem();

// Update totals based on transaction type
if (type.equals("Income")) {
   totalAmount += amount;
   double currentIncome = Double.parseDouble(incomeLabel.getText().replace("Total Income:
   ₹", ""));
   currentIncome += amount;
   incomeLabel.setText("Total Income: ₹" + String.format("%.2f", currentIncome));
} else { // Expense
   totalAmount -= amount;
   double currentExpense = Double.parseDouble(expenseLabel.getText().replace("Total Expense:
   ₹", ""));
   currentExpense += amount;
   expenseLabel.setText("Total Expense: ₹" + String.format("%.2f", currentExpense));
}

totalLabel.setText("Total Balance: ₹" + String.format("%.2f", totalAmount));

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
String dateString = dateFormat.format(selectedDate);

// Store expenses as negative values
if (type.equals("Expense")) {
   amount = -amount; // Store expenses as negative values in the database
}

// Add to database and get the new ID
int newId = addTransaction(typeCombo, descriptionCombo, amountField, yearSpinner,
monthSpinner, daySpinner);

// Check if the transaction was added successfully
if (newId != -1) {
   tableModel.addRow(new Object[]{newId, dateString, type, description, String.format("%.2f",
Math.abs(amount))}); // Use absolute value for display
} else {
   JOptionPane.showMessageDialog(dialog, "Failed to add transaction.",
      "Error", JOptionPane.ERROR_MESSAGE);
}
```

```
                dialog.dispose(); // Close the dialog after processing
            } catch (NumberFormatException ex)
                { JOptionPane.showMessageDialog(dialog, "Please enter a valid amount!",
                    "Error", JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {
                // Catch any other exceptions that might occur
                ex.printStackTrace();
                JOptionPane.showMessageDialog(dialog, "An unexpected error occurred: " + ex.getMessage(),
                    "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    dialog.getRootPane().registerKeyboardAction(
        e -> dialog.dispose(),
        KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0),
        JComponent.WHEN_IN_FOCUSED_WINDOW
    );

    dialog.setVisible(true);
}
```

3.5 UPDATE DATA OPERATION IN DASHBOARD

```
package update_data;

import dash_board.dashBoard; import

javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

private void showUpdateTransactionDialog() {
    int selectedRow = transactionTable.getSelectedRow();
    if (selectedRow == -1) {
        JOptionPane.showMessageDialog(frame, "Please select a transaction to update!",
                "No Selection", JOptionPane.WARNING_MESSAGE);
        return;
    }

    // Retrieve current transaction details
    int transactionId = (int) tableModel.getValueAt(selectedRow, 0); // Assuming ID is in column 0
    String currentDate = (String) tableModel.getValueAt(selectedRow, 1); // Assuming Date is in column 1
    String currentType = (String) tableModel.getValueAt(selectedRow, 2); // Assuming Type is in column 2
    String currentDescription = (String) tableModel.getValueAt(selectedRow, 3); // Assuming Description is in
column 3
    double currentAmount = Double.parseDouble((String) tableModel.getValueAt(selectedRow, 4)); //
Assuming Amount is in column 4

    JDialog dialog = new JDialog(frame, "Update Transaction", true);
    dialog.setLayout(new GridLayout(5, 2, 10, 10));
```

```
        dialog.setSize(400, 250);
        dialog.setLocationRelativeTo(frame);

        // Fields for updating transaction
        JTextField dateField = new JTextField(currentDate);
        JComboBox<String> typeCombo = new JComboBox<>(new String[]{"Expense", "Income"});
        typeCombo.setSelectedItem(currentType);
        String[] descriptions = {"Medical", "Shopping", "Entertainment", "Transport", "Grocery", "Others"};
        JComboBox<String> descriptionCombo = new JComboBox<>(descriptions);
        JTextField amountField = new JTextField(String.valueOf(currentAmount));

        dialog.add(new JLabel("Date:"));
        dialog.add(dateField);
        dialog.add(new JLabel("Type:"));
        dialog.add(typeCombo);
        dialog.add(new JLabel("Description:"));
        dialog.add(descriptionCombo);
        dialog.add(new JLabel("Amount:"));
        dialog.add(amountField);

        JButton updateButton = new JButton("Update");
        dialog.add(new JLabel(""));
        dialog.add(updateButton);

        updateButton.addActionListener(new ActionListener()
          { @Override
          public void actionPerformed(ActionEvent e)
             { try {
                String date = dateField.getText().trim();
                String type = (String) typeCombo.getSelectedItem();
                String description = (String) descriptionCombo.getSelectedItem();
                double amount = Double.parseDouble(amountField.getText().trim());

                // Create a Transaction object to update
                Transaction transaction = new Transaction(transactionId, date, type, description, amount);
                TransactionDAO transactionDAO = new TransactionDAO();
                transactionDAO.updateTransaction(transaction); // Update in the database

                // Update the table model
                updateTableModel(transaction);

                // Update total income, total expense, and total balance
                updateTotals(transaction, currentAmount, type);

                dialog.dispose(); // Close the dialog
                JOptionPane.showMessageDialog(frame, "Transaction updated successfully.", "Success",
JOptionPane.INFORMATION_MESSAGE);
             } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(dialog, "Please enter a valid amount!", "Error",
JOptionPane.ERROR_MESSAGE);
             }
          }
        });

        dialog.setVisible(true);
```

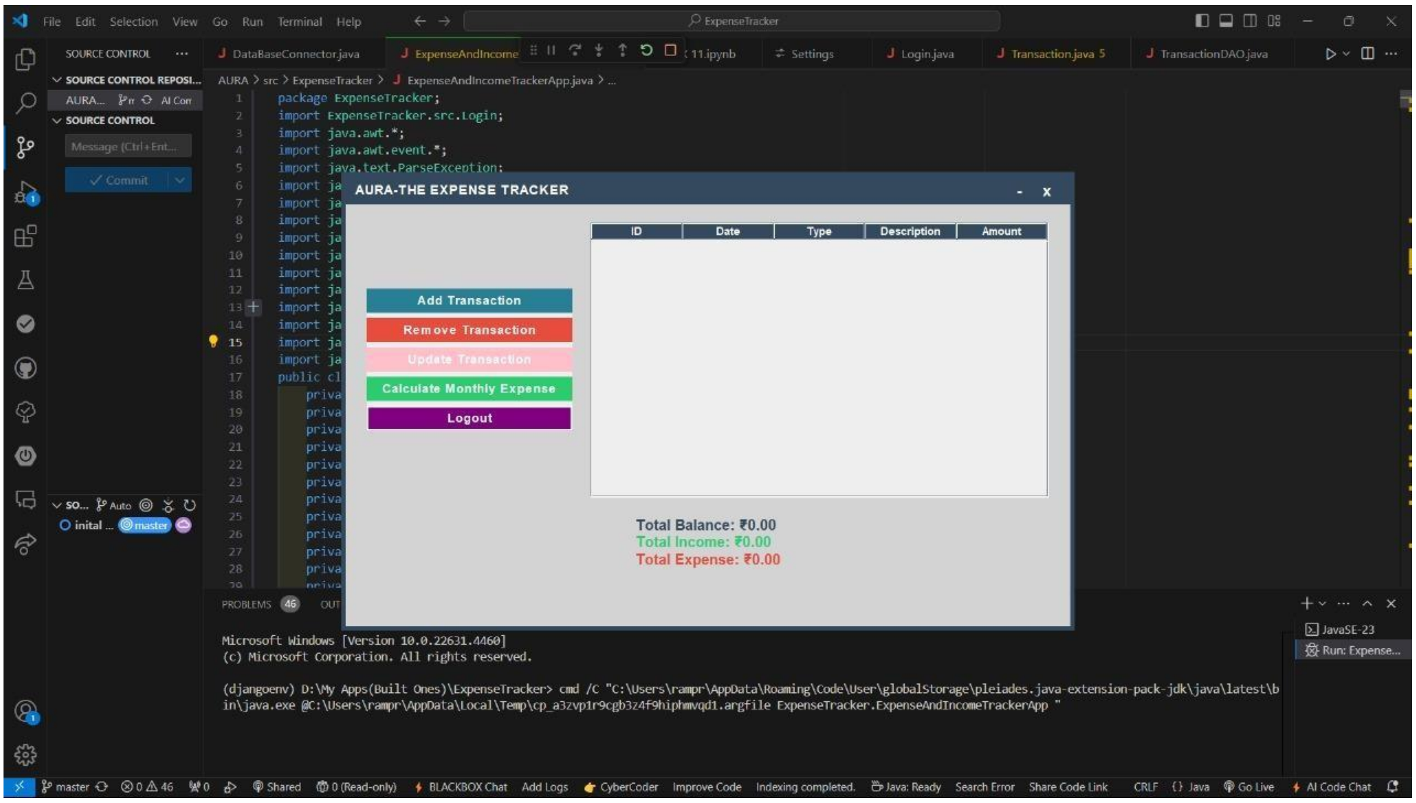
```
    }



    private void showMonthlyExpenseDialog() {
        JDialog dialog = new JDialog(frame, "Calculate Monthly Summary", true);
        dialog.setLayout(new GridLayout(3, 2, 10, 10));
        dialog.setSize(300, 150);
        dialog.setLocationRelativeTo(frame);

        JComboBox<String> monthCombo = new JComboBox<>(new
            String[]{ "January", "February", "March", "April", "May", "June",
            "July", "August", "September", "October", "November", "December"
        });
        JTextField yearField = new JTextField();
        JButton calculateButton = new JButton("Calculate");

        dialog.add(new JLabel("Month:"));
        dialog.add(monthCombo);
        dialog.add(new JLabel("Year:"));
        dialog.add(yearField);
        dialog.add(new JLabel(""));
        dialog.add(calculateButton);

        calculateButton.addActionListener(new ActionListener()
            { @Override
            public void actionPerformed(ActionEvent e)
                { try {
                    int month = monthCombo.getSelectedIndex() + 1; // January is 0, so add 1
                    int year = Integer.parseInt(yearField.getText());

                    double monthlyExpense = calculateMonthlyExpense(month, year);
                    double monthlyIncome = calculateMonthlyIncome(month, year);
                    double netAmount = monthlyIncome - monthlyExpense;

                    String message = String.format("Monthly Summary for %s %d:\n\n" +
                                        "Total Income: ₹%.2f\n" +
                                        "Total Expense: ₹%.2f\n" +
                                        "Net Amount: ₹%.2f",
                                        monthCombo.getSelectedItem(), year,
                                        monthlyIncome, monthlyExpense, netAmount);

                    JOptionPane.showMessageDialog(dialog, message, "Monthly Summary",
JOptionPane.INFORMATION_MESSAGE);
                    dialog.dispose();
                } catch (NumberFormatException ex)
                    { JOptionPane.showMessageDialog(dialog, "Please enter a valid year!",
                        "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        dialog.setVisible(true);
    }
```

3.6 DATABASE MANAGEMENT

```
package ExpenseTracker;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DataBaseConnector {
    private static final String DB_NAME = "expense_income_db";
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/" + DB_NAME;
    private static final String USER = "root";
    private static final String PASSWORD = "";

    public static Connection getConnection()
        { Connection connection = null;

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(JDBC_URL, USER, PASSWORD);
            System.out.println("Connected to the database");
        } catch (ClassNotFoundException | SQLException ex)
            { System.out.println("Connection - ClassNotFoundException: " + ex.getMessage());
        }

        return connection;
    }
}
```

SNAPSHOTS

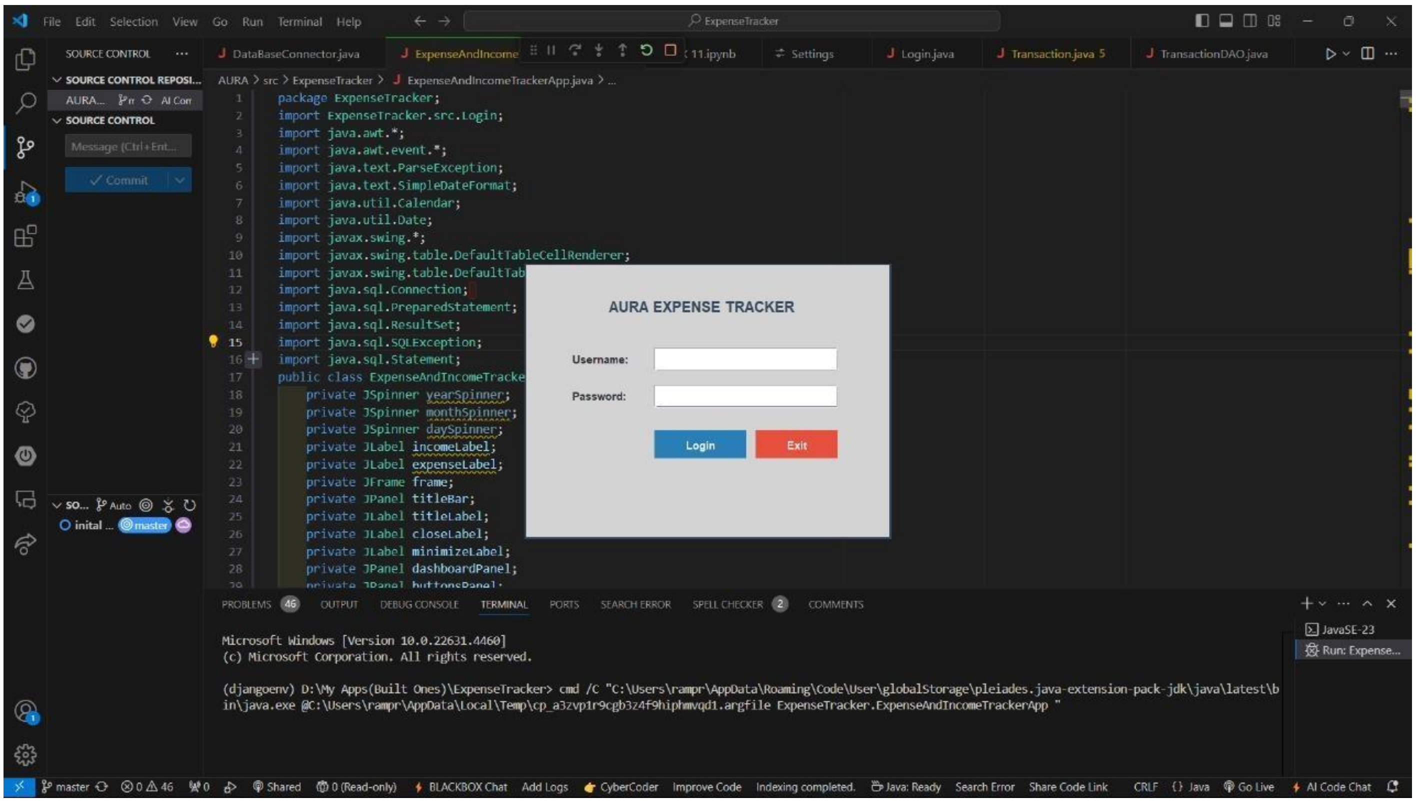
4.1 HOME PAGE



```
package ExpenseTracker;
import ExpenseTracker.src.Login;
import java.awt.*;
import java.awt.event.*;
import java.text.ParseException;
import ja
import ja
import ja
import ja
import ja
import ja
import ja
import ja
import ja
import ja
import ja
public cl
    priva
    priva
    priva
    priva
    priva
    priva
    priva
    priva
    priva
    priva
    priva
```

```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

(djangoenv) D:\My Apps(Built Ones)\ExpenseTracker> cmd /C "C:\Users\rampr\AppData\Roaming\Code\User\globalStorage\pleiades.java-extension-pack-jdk\java\latest\b
in\java.exe @C:\Users\rampr\AppData\Local\Temp\cp_a3zvp1r9cgb3z4f9hiphmvqd1.argfile ExpenseTracker.ExpenseAndIncomeTrackerApp "
```

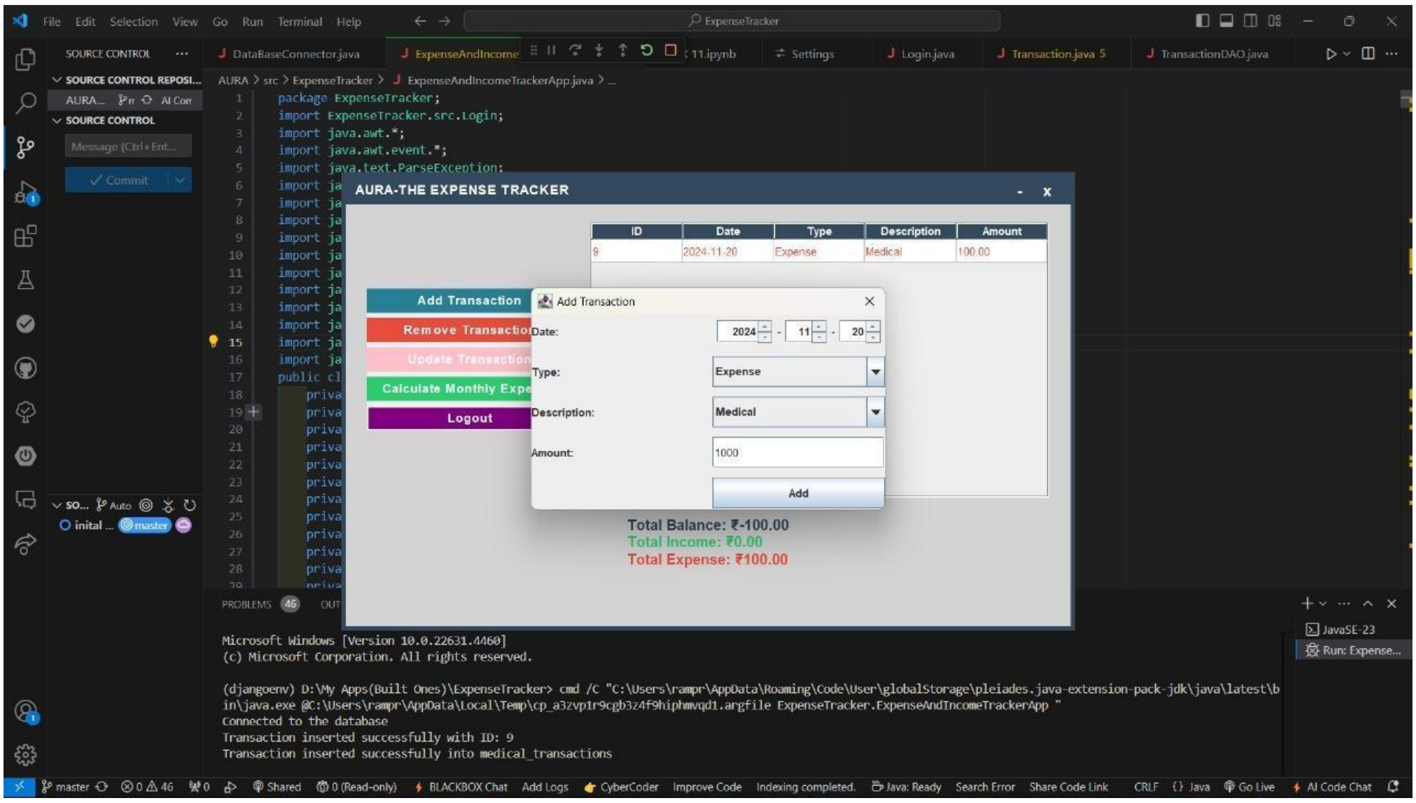
4.2 LOGIN PAGE



```
package ExpenseTracker;
import ExpenseTracker.src.Login;
import java.awt.*;
import java.awt.event.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import javax.swing.*;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTab
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class ExpenseAndIncomeTracke
    private JSpinner yearSpinner;
    private JSpinner monthSpinner;
    private JSpinner daySpinner;
    private JLabel incomeLabel;
    private JLabel expenseLabel;
    private JFrame frame;
    private JPanel titleBar;
    private JLabel titleLabel;
    private JLabel closeLabel;
    private JLabel minimizeLabel;
    private JPanel dashboardPanel;
```

```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

(djangoenv) D:\My Apps(Built Ones)\ExpenseTracker> cmd /C "C:\Users\rampr\AppData\Roaming\Code\User\globalStorage\pleiades.java-extension-pack-jdk\java\latest\b
in\java.exe @C:\Users\rampr\AppData\Local\Temp\cp_a3zvp1r9cgb3z4f9hiphmvqd1.argfile ExpenseTracker.ExpenseAndIncomeTrackerApp "
```

4.3 ADD TRANSACTION



```java
package ExpenseTracker;
import ExpenseTracker.src.Login;
import java.awt.*;
import java.awt.event.*;
import java.text.ParseException;
```

AURA-THE EXPENSE TRACKER

ID	Date	Type	Description	Amount
9	2024-11-20	Expense	Medical	100.00

Add Transaction

Date: 2024 - 11 - 20

Type: Expense

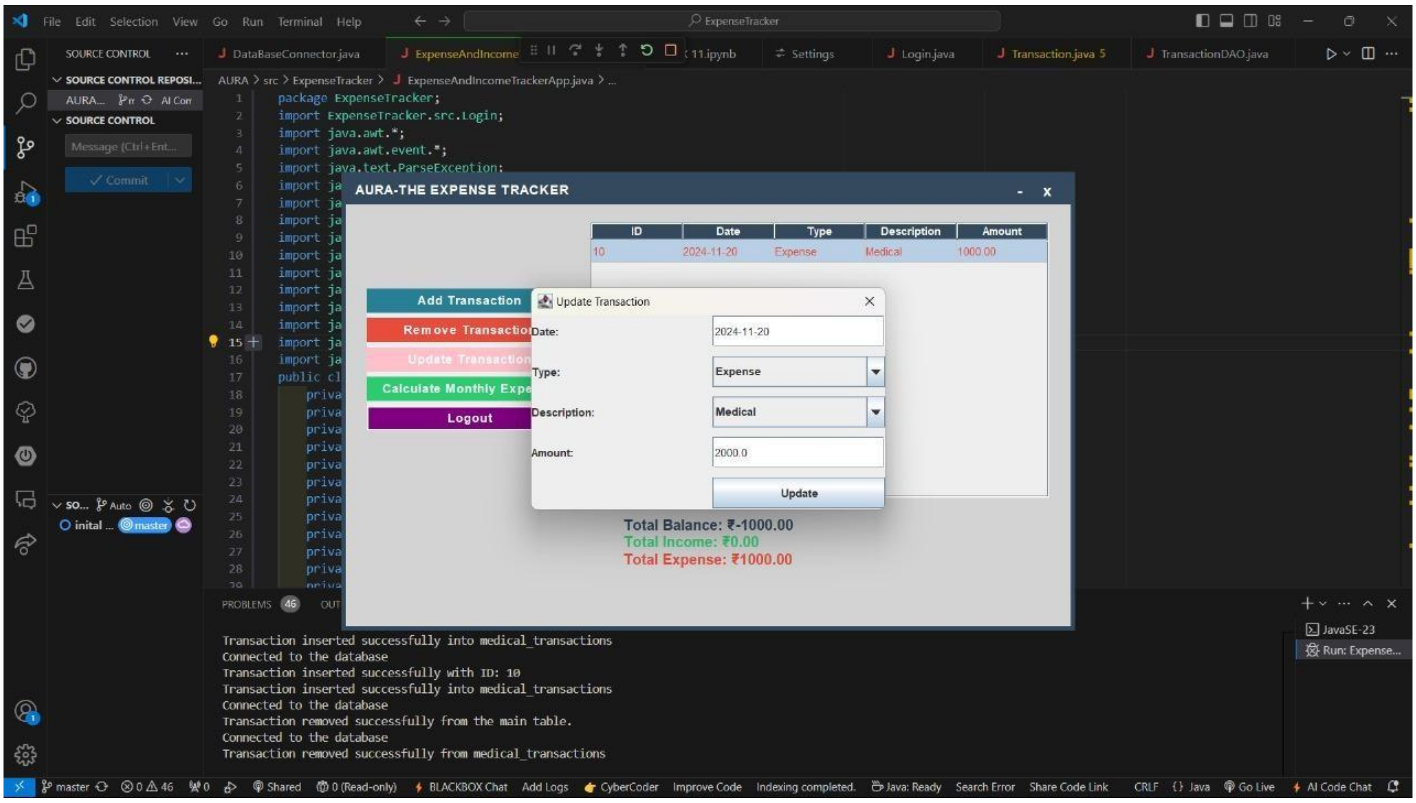
Description: Medical

Amount: 1000

Total Balance: ₹-100.00
Total Income: ₹0.00
Total Expense: ₹100.00

```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

(djangoenv) D:\My Apps(Built Ones)\ExpenseTracker> cmd /C "C:\Users\rampr\AppData\Roaming\Code\User\globalStorage\pleiades.java-extension-pack-jdk\java\latest\b
in\java.exe @C:\Users\rampr\AppData\Local\Temp\cp_a3zvp1r9cgb3z4f9hiphmvqd1.argfile ExpenseTracker.ExpenseAndIncomeTrackerApp "
Connected to the database
Transaction inserted successfully with ID: 9
Transaction inserted successfully into medical_transactions
```

4.4 UPDATE TRANSACTION



```java
package ExpenseTracker;
import ExpenseTracker.src.Login;
import java.awt.*;
import java.awt.event.*;
import java.text.ParseException;
```

AURA-THE EXPENSE TRACKER

ID	Date	Type	Description	Amount
10	2024-11-20	Expense	Medical	1000.00

Update Transaction

Date: 2024-11-20

Type: Expense

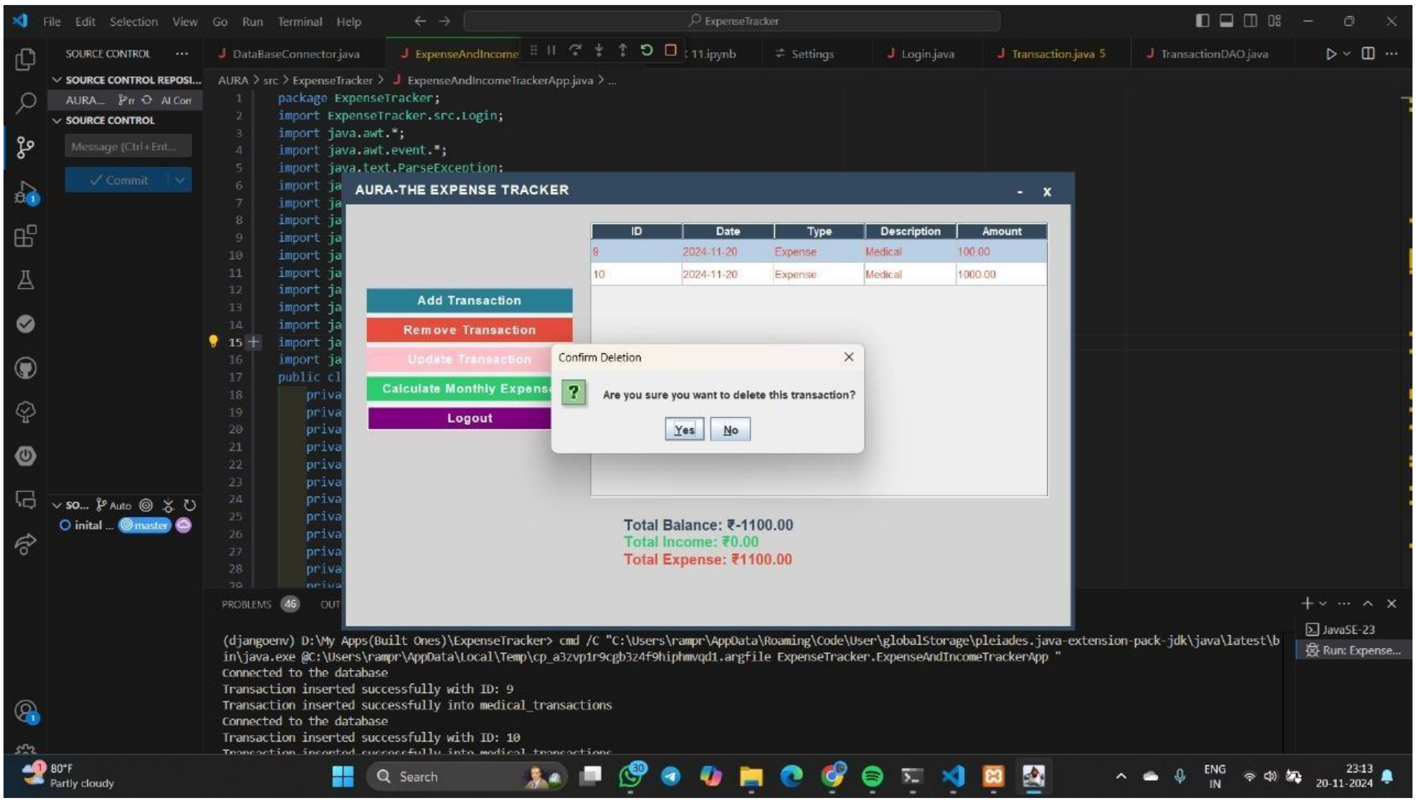
Description: Medical

Amount: 2000.0

Total Balance: ₹-1000.00
Total Income: ₹0.00
Total Expense: ₹1000.00

```
Transaction inserted successfully into medical_transactions
Connected to the database
Transaction inserted successfully with ID: 10
Transaction inserted successfully into medical_transactions
Connected to the database
Transaction removed successfully from the main table.
Connected to the database
Transaction removed successfully from medical_transactions
```

4.4 REMOVE TRANSACTION



```
package ExpenseTracker;
import ExpenseTracker.src.Login;
import java.awt.*;
import java.awt.event.*;
import java.text.ParseException;
```

AURA-THE EXPENSE TRACKER

ID	Date	Type	Description	Amount
9	2024-11-20	Expense	Medical	100.00
10	2024-11-20	Expense	Medical	1000.00

Confirm Deletion

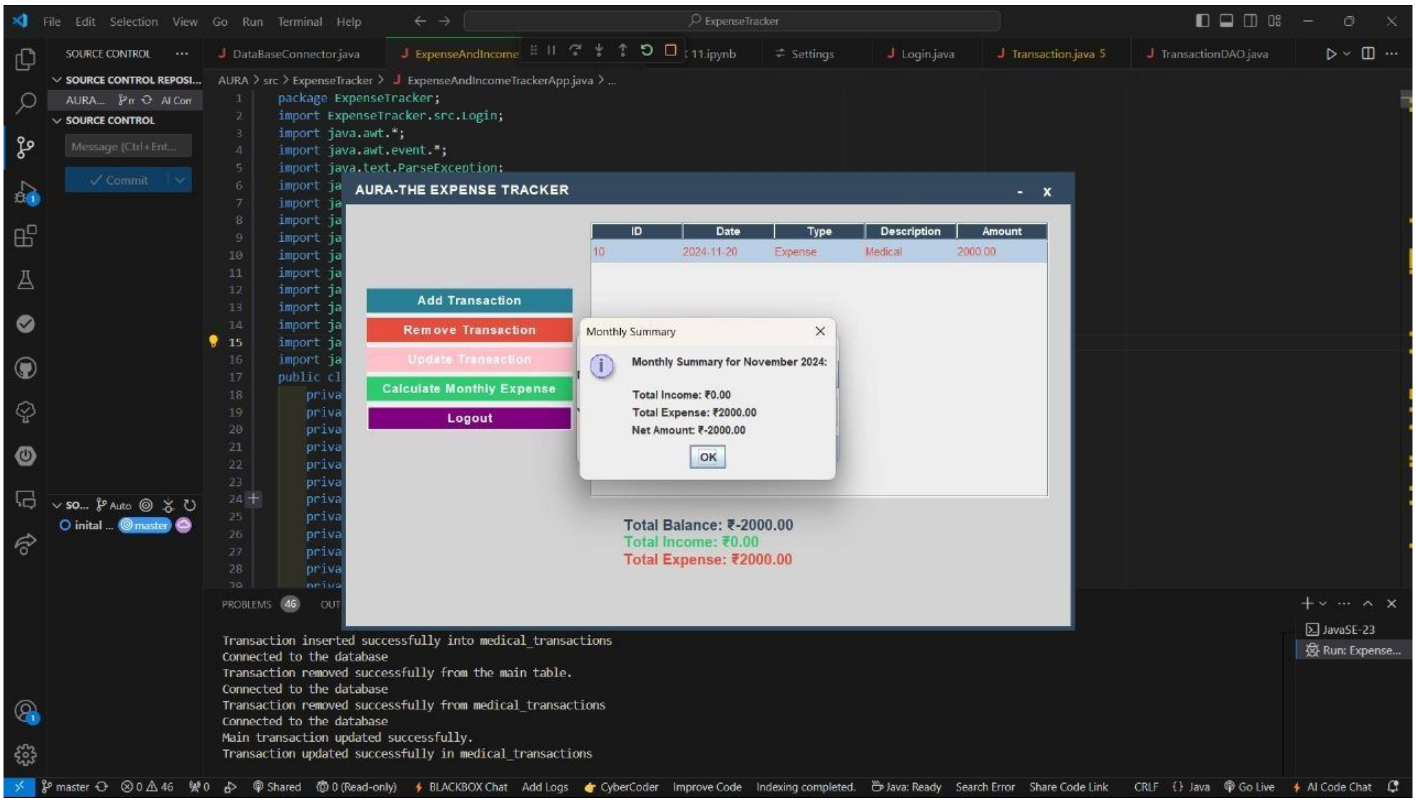
Are you sure you want to delete this transaction?

Yes No

Total Balance: ₹-1100.00
Total Income: ₹0.00
Total Expense: ₹1100.00

```
(djangoenv) D:\My Apps(Built Ones)\ExpenseTracker> cmd /C "C:\Users\rampr\AppData\Roaming\Code\User\globalStorage\pleiades.java-extension-pack-jdk\java\latest\b
in\java.exe @C:\Users\rampr\AppData\Local\Temp\cp_a3zvp1r9cgb3z4f9hiphmvqd1.argfile ExpenseTracker.ExpenseAndIncomeTrackerApp "
Connected to the database
Transaction inserted successfully with ID: 9
Transaction inserted successfully into medical_transactions
Connected to the database
Transaction inserted successfully with ID: 10
```

4.5 CALCULATE MONTHLY EXPENSE



```
package ExpenseTracker;
import ExpenseTracker.src.Login;
import java.awt.*;
import java.awt.event.*;
import java.text.ParseException;
```

AURA-THE EXPENSE TRACKER

ID	Date	Type	Description	Amount
10	2024-11-20	Expense	Medical	2000.00

Monthly Summary

Monthly Summary for November 2024:

Total Income: ₹0.00
Total Expense: ₹2000.00
Net Amount: ₹-2000.00

OK

Total Balance: ₹-2000.00
Total Income: ₹0.00
Total Expense: ₹2000.00

```
Transaction inserted successfully into medical_transactions
Connected to the database
Transaction removed successfully from the main table.
Connected to the database
Transaction removed successfully from medical_transactions
Connected to the database
Main transaction updated successfully.
Transaction updated successfully in medical_transactions
```

CONCLUSION

Upon completion of the Expense Tracker application, we are confident that the issues present in manual expense tracking systems have been effectively addressed. The "EXPENSE TRACKER" application has been designed to automate the process of managing personal finances, thereby reducing human errors and enhancing overall efficiency. The primary focus of this project is to simplify the tracking of expenses and incomes, allowing users to manage their finances with minimal effort.

All transaction records are securely stored in a database, enabling quick and easy retrieval of data. Users can navigate through their financial records seamlessly, with intuitive controls provided throughout the application. For instances where users have a large number of transactions, the application features a search functionality that allows them to quickly find specific records by entering relevant search terms, delivering results almost instantaneously.

Editing transactions has also been streamlined, as users can easily modify any required fields and update their entries with a simple click of a button. Each transaction is assigned a unique identifier, ensuring accurate access and management of financial records without the risk of confusion or error.
The main aim of this project is to empower users with the tools they need to maintain accurate financial records, analyze their spending habits, and make informed decisions regarding their finances. The Expense Tracker application stands as a comprehensive solution for individuals seeking to take control of their financial well-being.

REFERENCES

1. **https://www.javatpoint.com/java-tutorial**
2. https://www.wikipedia.org/
3. https://www.w3schools.com/sql/
4. SQL | Codecademy