Cashflow Minimizer

Aim :

The aim of the **Cashflow Minimizer** project is to design and implement a software solution in Java that optimizes cash flow management by analyzing and categorizing financial transactions. The project seeks to help users efficiently manage their income and expenses, minimize cash outflows, and ensure timely handling of financial commitments.

Coding :

```
import javax.swing.*;

import java.awt.*;

import java.util.HashMap;

import java.util.LinkedList;

import java.util.Map;

class Transaction {

    String date; // Expected format: yyyy-mm

    double amount;

    String category;

    public Transaction(String date, double amount, String category) {

        this.date = date;

        this.amount = amount;

        this.category = category;

    }

    @Override

    public String toString() {

        return "Date: " + date + ", Amount: " + amount + ", Category: " + category;

    }

}

class CategoryThreshold {

    String category;

    double threshold;
```

```
    public CategoryThreshold(String category, double threshold) {
        this.category = category;
        this.threshold = threshold;
    }
}
class ExpenseManager {
    LinkedList<Transaction> transactions = new LinkedList<>();
    LinkedList<CategoryThreshold> categoryThresholds = new LinkedList<>();
    public void addTransaction(String date, double amount, String category) {
        Transaction transaction = new Transaction(date, amount, category);
        transactions.add(transaction);
        checkThreshold(category);
    }
    public void setCategoryThreshold(String category, double threshold) {
        CategoryThreshold categoryThreshold = new CategoryThreshold(category, threshold);
        categoryThresholds.add(categoryThreshold);
    }
    public void checkThreshold(String category) {
        double totalSpent = 0.0;
        for (Transaction transaction : transactions) {
            if (transaction.category.equals(category)) {
                totalSpent += transaction.amount;
            }
        }
        for (CategoryThreshold threshold : categoryThresholds) {
            if (threshold.category.equals(category) && totalSpent > threshold.threshold) {
                JOptionPane.showMessageDialog(null, "Alert: Spending limit exceeded for " +
category + ". Current spending: " + totalSpent);
            }
        }}
```

```
    public Map<String, Map<String, Double>> getMonthlyCategoryTotals() {
        Map<String, Map<String, Double>> monthlyCategoryTotals = new HashMap<>();
        for (Transaction transaction : transactions) {
            monthlyCategoryTotals
                    .computeIfAbsent(transaction.date, k -> new HashMap<>())
                    .merge(transaction.category, transaction.amount, Double::sum);
        }
        return monthlyCategoryTotals;
    }
    public String viewTransactions() {
        StringBuilder history = new StringBuilder();
        for (Transaction transaction : transactions) {
            history.append(transaction).append("\n");
        }
        return history.toString();
    }
}
class BackgroundPanel extends JPanel {
    private Image backgroundImage;
    public BackgroundPanel(String imagePath) {
        try {
            backgroundImage = new ImageIcon(imagePath).getImage();
        } catch (Exception e) {
            System.out.println("Image not found at: " + imagePath);
        }
    }
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
```

```
        if (backgroundImage != null) {
            g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
        }
    }
}
public class PocketPlanAppGUI extends JFrame {
    ExpenseManager manager = new ExpenseManager();
    public PocketPlanAppGUI() {
        setTitle("PocketPlan Expense Manager");
        setSize(600, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        BackgroundPanel mainPanel = new BackgroundPanel("logo.png");
        mainPanel.setLayout(new BorderLayout());
        JPanel centerPanel = new JPanel();
        centerPanel.setOpaque(false);
        centerPanel.setLayout(new BoxLayout(centerPanel, BoxLayout.Y_AXIS));
        centerPanel.add(Box.createVerticalGlue());
        centerPanel.add(createButtonPanel());
        centerPanel.add(Box.createVerticalGlue());
        mainPanel.add(centerPanel, BorderLayout.CENTER);
        add(mainPanel);
    }
    private JPanel createButtonPanel() {
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.Y_AXIS));
        buttonPanel.setOpaque(false);
        JButton addCategoryButton = new JButton("Add Category & Threshold");
        JButton addTransactionButton = new JButton("Add Transaction");
```

```
    JButton viewTransactionsButton = new JButton("View Transactions");
    JButton overviewButton = new JButton("Overview");  // Overview Button
    Dimension buttonSize = new Dimension(200, 50);
    Font buttonFont = new Font("Arial", Font.BOLD, 16);
    addCategoryButton.setPreferredSize(buttonSize);
    addTransactionButton.setPreferredSize(buttonSize);
    viewTransactionsButton.setPreferredSize(buttonSize);
    overviewButton.setPreferredSize(buttonSize);
    addCategoryButton.setFont(buttonFont);
    addTransactionButton.setFont(buttonFont);
    viewTransactionsButton.setFont(buttonFont);
    overviewButton.setFont(buttonFont);
    addCategoryButton.addActionListener(e -> addCategoryAndThreshold());
    addTransactionButton.addActionListener(e -> addTransaction());
    viewTransactionsButton.addActionListener(e -> viewTransactions());
    overviewButton.addActionListener(e -> showOverview());
    buttonPanel.add(addCategoryButton);
    buttonPanel.add(Box.createVerticalStrut(20));
    buttonPanel.add(addTransactionButton);
    buttonPanel.add(Box.createVerticalStrut(20));
    buttonPanel.add(viewTransactionsButton);
    buttonPanel.add(Box.createVerticalStrut(20));
    buttonPanel.add(overviewButton);
    return buttonPanel;
}
private void showOverview() {
JFrame overviewFrame = new JFrame("Monthly Transaction Overview");
overviewFrame.setSize(600, 600);
overviewFrame.setLocationRelativeTo(this);
```

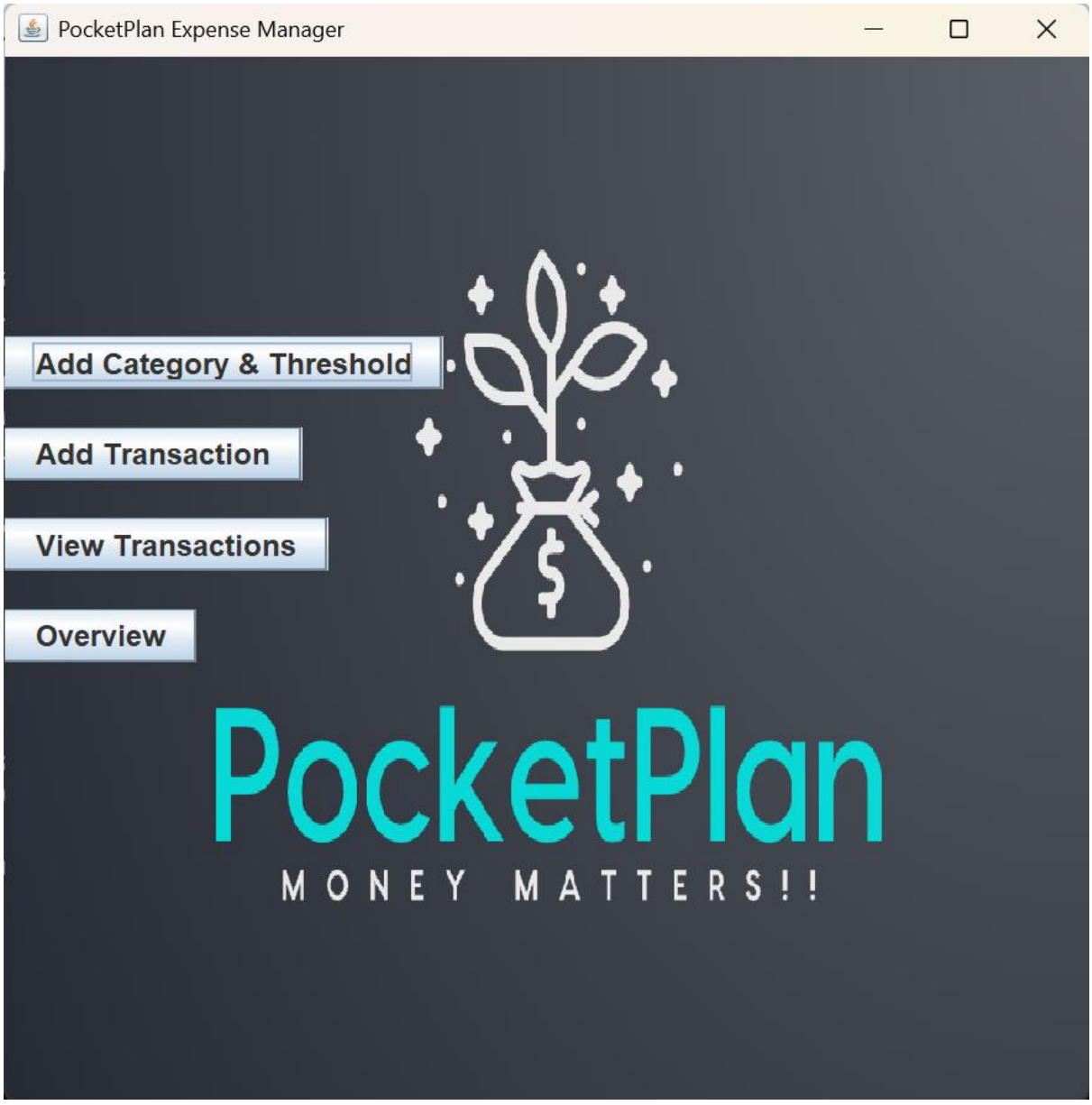
```
    JPanel mainPanel = new JPanel();
    mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y_AXIS));
    Map<String, Map<String, Double>> monthlyCategoryTotals =
manager.getMonthlyCategoryTotals();
    Color[] barColors = {Color.BLUE, Color.RED, Color.GREEN, Color.ORANGE, Color.MAGENTA,
Color.CYAN, Color.PINK, Color.YELLOW};
    for (Map.Entry<String, Map<String, Double>> entry : monthlyCategoryTotals.entrySet()) {
        String month = entry.getKey();
        Map<String, Double> categoryTotals = entry.getValue();
        JPanel chartPanel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                int x = 50;
                int width = 50;
                int maxHeight = 200;
                double maxTotal =
categoryTotals.values().stream().max(Double::compare).orElse(1.0);
                int colorIndex = 0;
                for (Map.Entry<String, Double> categoryEntry : categoryTotals.entrySet()) {
                    int barHeight = (int) ((categoryEntry.getValue() / maxTotal) * maxHeight);
                    g.setColor(barColors[colorIndex % barColors.length]);
                    g.fillRect(x, maxHeight - barHeight + 50, width, barHeight);
                    g.setColor(Color.BLACK);
                    g.drawString(categoryEntry.getKey(), x, maxHeight + 70);
                    x += width + 20;
                    colorIndex++;
                }
            }       };
```

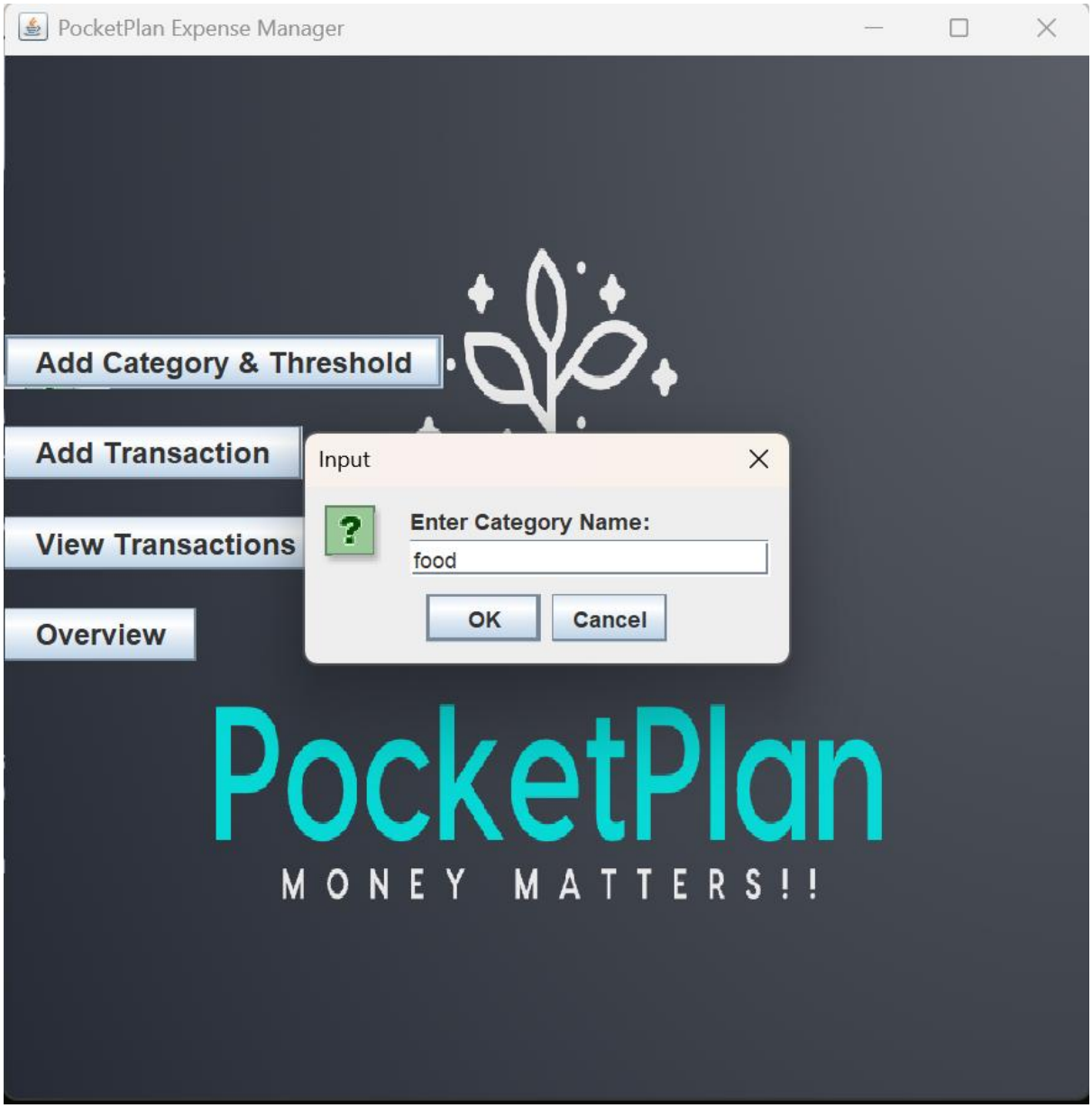
```
        chartPanel.setPreferredSize(new Dimension(500, 300));
        chartPanel.setBorder(BorderFactory.createTitledBorder("Transactions for " + month));
        mainPanel.add(chartPanel);
    }
    JScrollPane scrollPane = new JScrollPane(mainPanel);
    overviewFrame.add(scrollPane);
    overviewFrame.setVisible(true);
    }
    private void addCategoryAndThreshold() {
        String category = JOptionPane.showInputDialog(this, "Enter Category Name:");
        if (category == null || category.trim().isEmpty()) return;
        String thresholdStr = JOptionPane.showInputDialog(this, "Enter Threshold for " +
category + ":");
        if (thresholdStr == null || thresholdStr.trim().isEmpty()) return;
        try {
            double threshold = Double.parseDouble(thresholdStr);
            manager.setCategoryThreshold(category, threshold);
            JOptionPane.showMessageDialog(this, "Category " + category + " added with
threshold " + threshold);
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter a valid number for the
threshold.");
        }
    }
    private void addTransaction() {
        String[] months = {"January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"};
        JComboBox<String> monthComboBox = new JComboBox<>(months);
        JTextField yearField = new JTextField(4);
        JPanel datePanel = new JPanel();
        datePanel.add(new JLabel("Month:"));
```

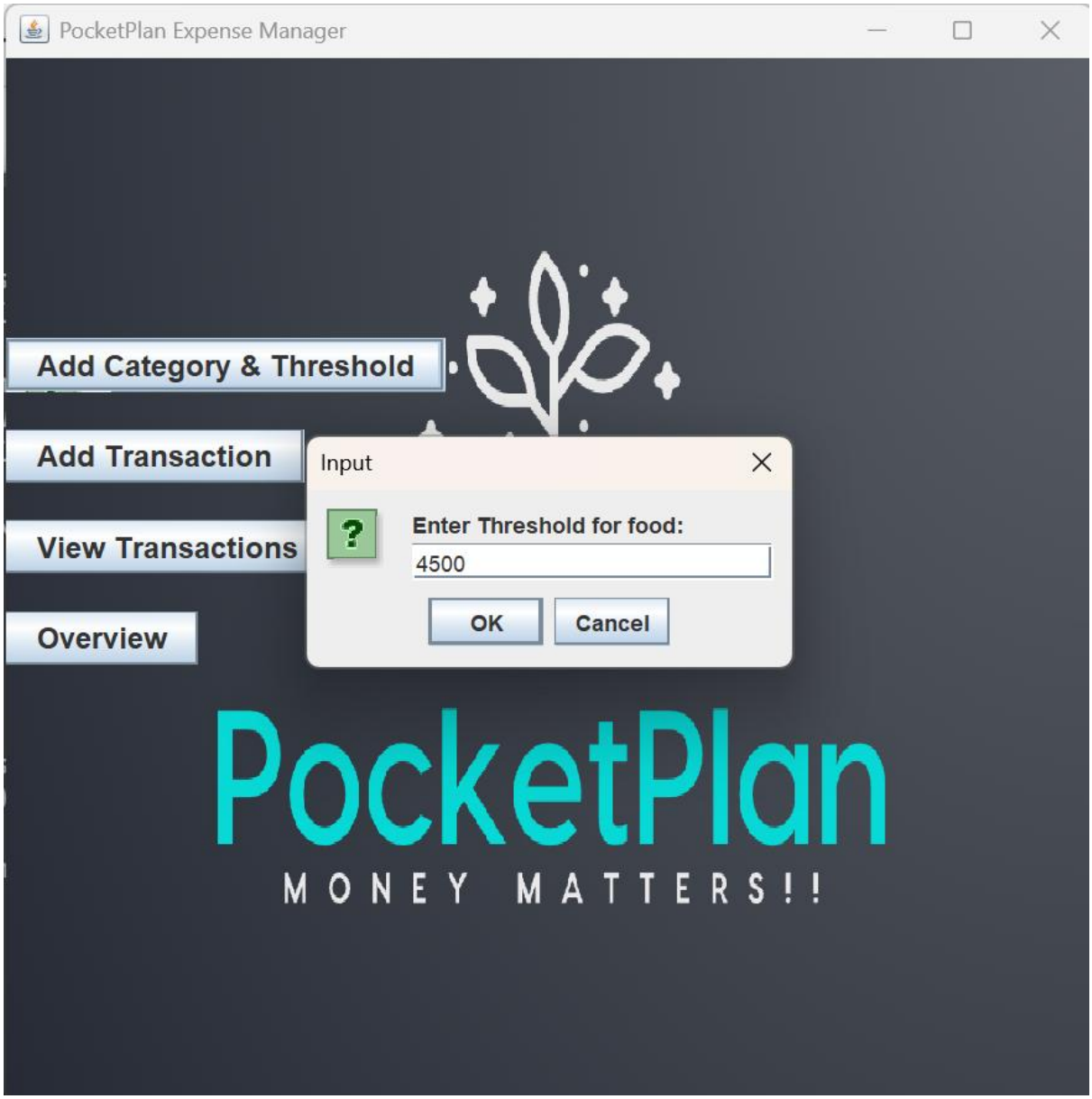
```
        datePanel.add(monthComboBox);
        datePanel.add(Box.createHorizontalStrut(15));
        datePanel.add(new JLabel("Year:"));
        datePanel.add(yearField);
        int result = JOptionPane.showConfirmDialog(this, datePanel, "Enter Date",
JOptionPane.OK_CANCEL_OPTION);
        if (result == JOptionPane.CANCEL_OPTION || yearField.getText().trim().isEmpty())
return;
        String month = (String) monthComboBox.getSelectedItem();
        String year = yearField.getText().trim();
        String date = year + "-" + (monthComboBox.getSelectedIndex() + 1);
        String amountStr = JOptionPane.showInputDialog(this, "Enter Amount:");
        if (amountStr == null || amountStr.trim().isEmpty()) return;
        try {
            double amount = Double.parseDouble(amountStr);
            String[] categories = manager.categoryThresholds.stream().map(ct ->
ct.category).toArray(String[]::new);
            if (categories.length == 0) {
                JOptionPane.showMessageDialog(this, "No categories available. Please add a
category first.");
                return;
            }
            String selectedCategory = (String) JOptionPane.showInputDialog(this, "Select
Category:", "Category", JOptionPane.QUESTION_MESSAGE, null, categories, categories[0]);
            manager.addTransaction(date, amount, selectedCategory);
            JOptionPane.showMessageDialog(this, "Transaction added successfully.");
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter a valid number for the
amount.");
        }
    }
```

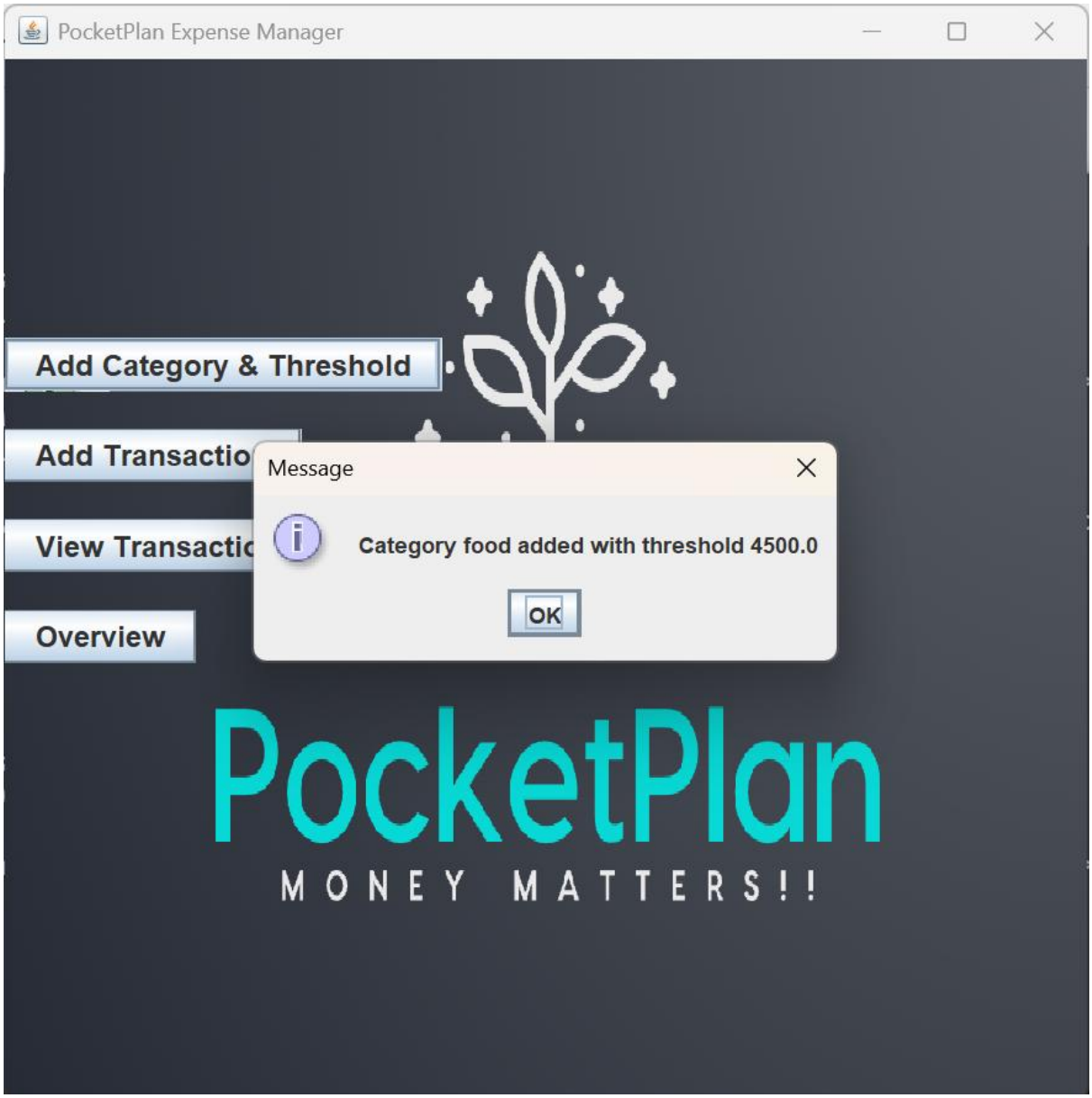
```
    private void viewTransactions() {
        JOptionPane.showMessageDialog(this, manager.viewTransactions());
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            PocketPlanAppGUI app = new PocketPlanAppGUI();
            app.setVisible(true);
        });
    }
}
```

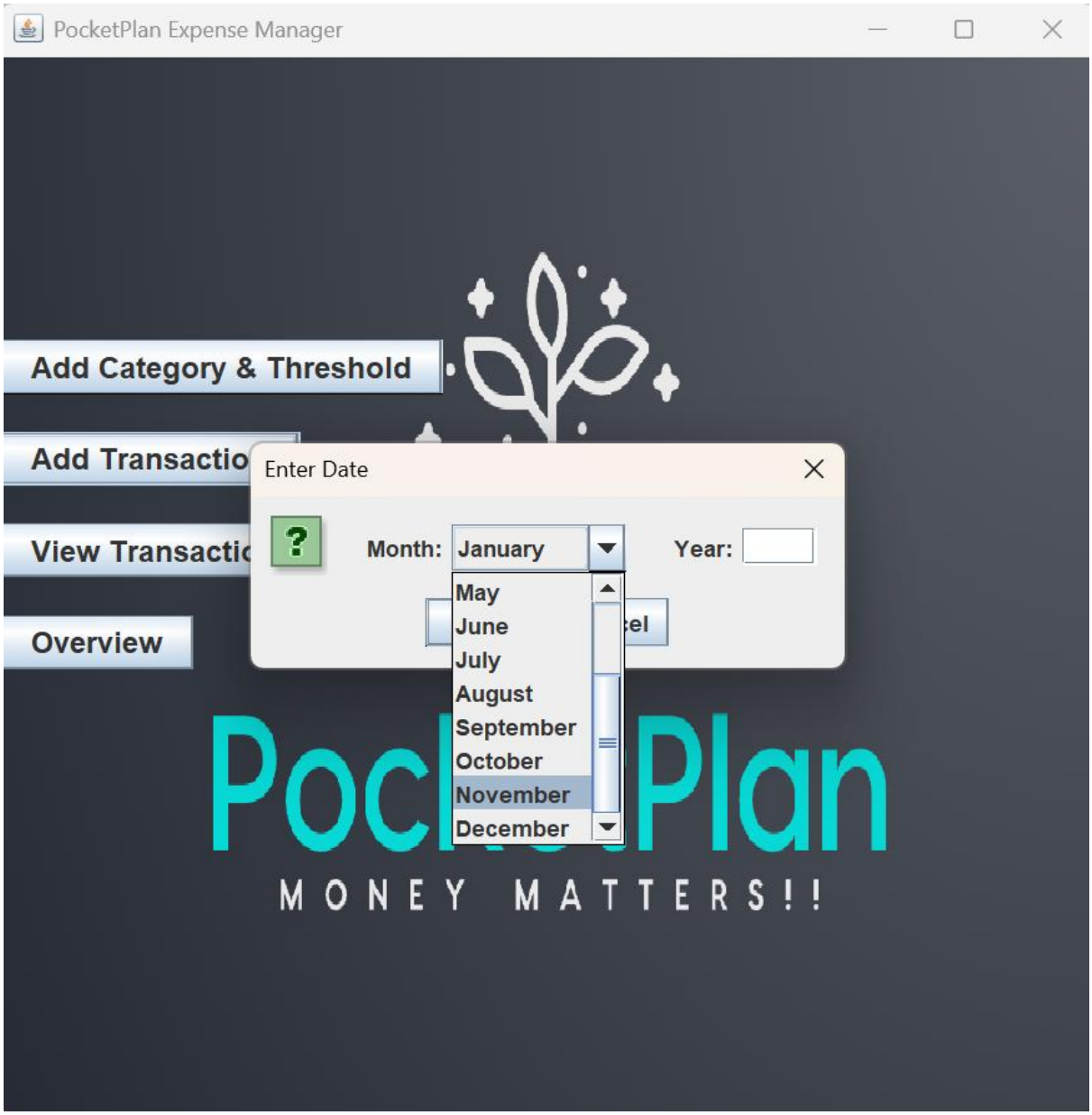
ScreenShorts :

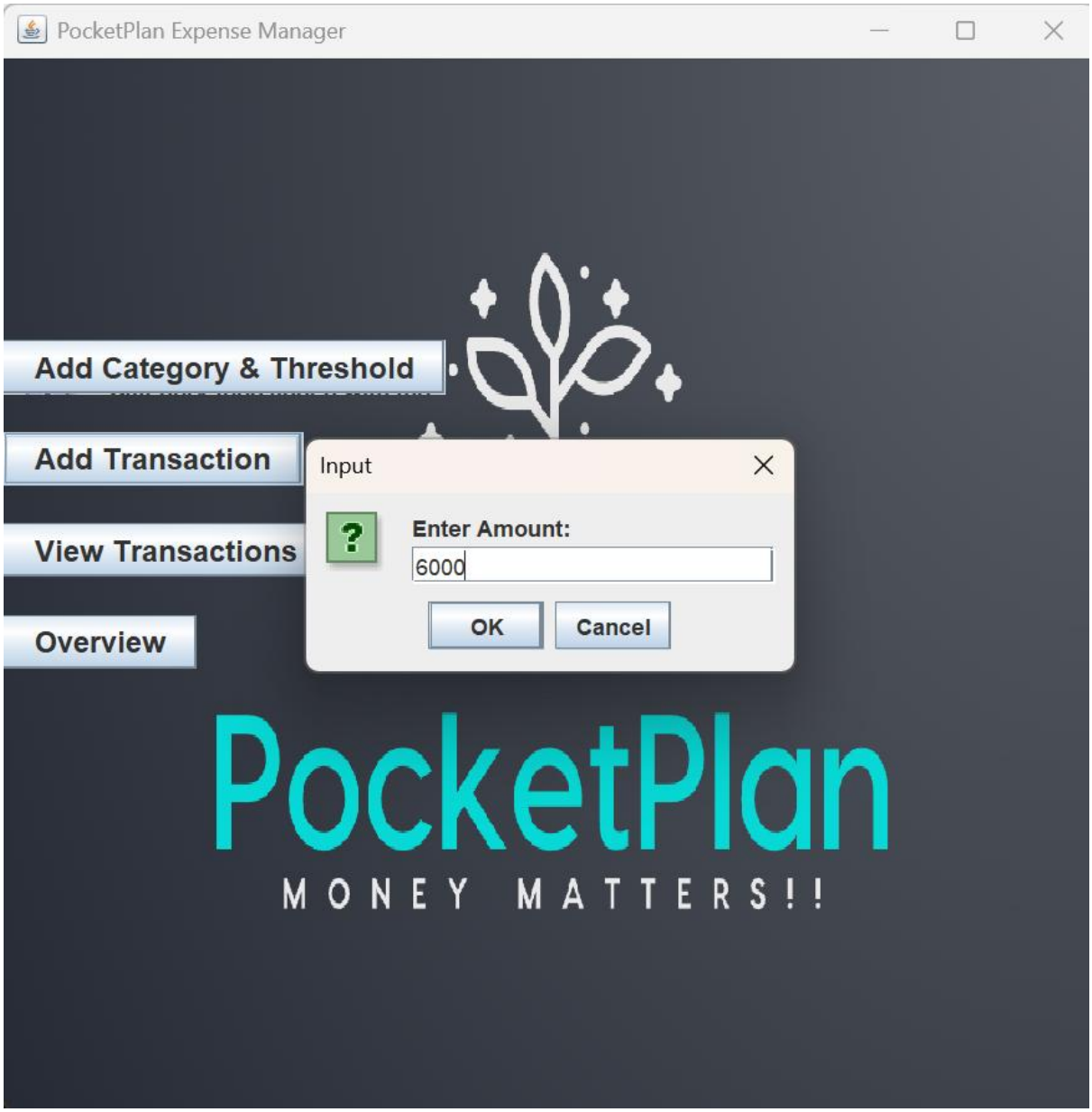

PocketPlan Expense Manager
Add Category & Threshold
Add Transaction
View Transactions
Overview
PocketPlan
MONEY MATTERS!!

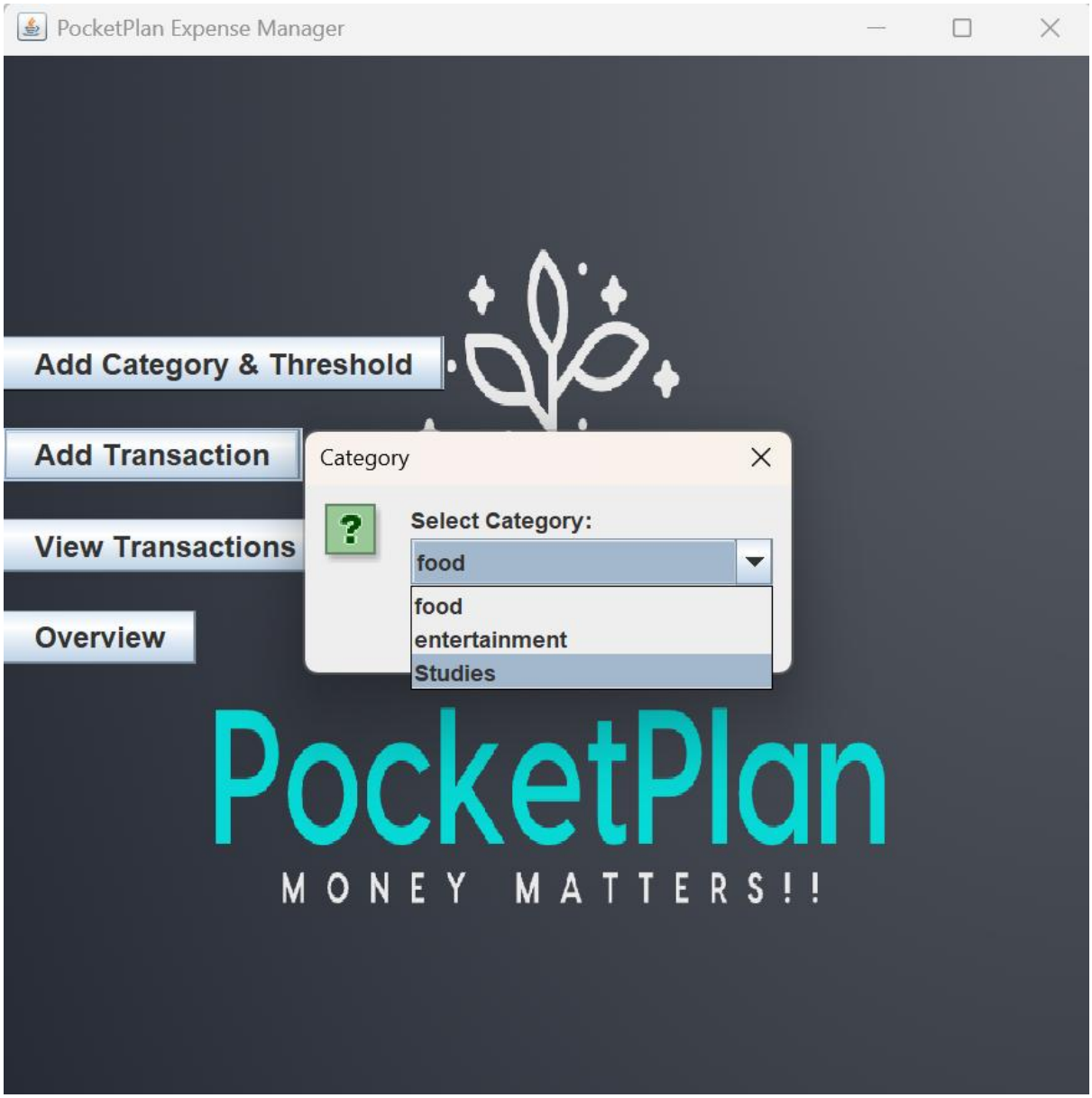

PocketPlan Expense Manager
Add Category & Threshold
Add Transaction
View Transactions
Overview
Input
Enter Category Name:
food
OK
Cancel
PocketPlan
MONEY MATTERS!!

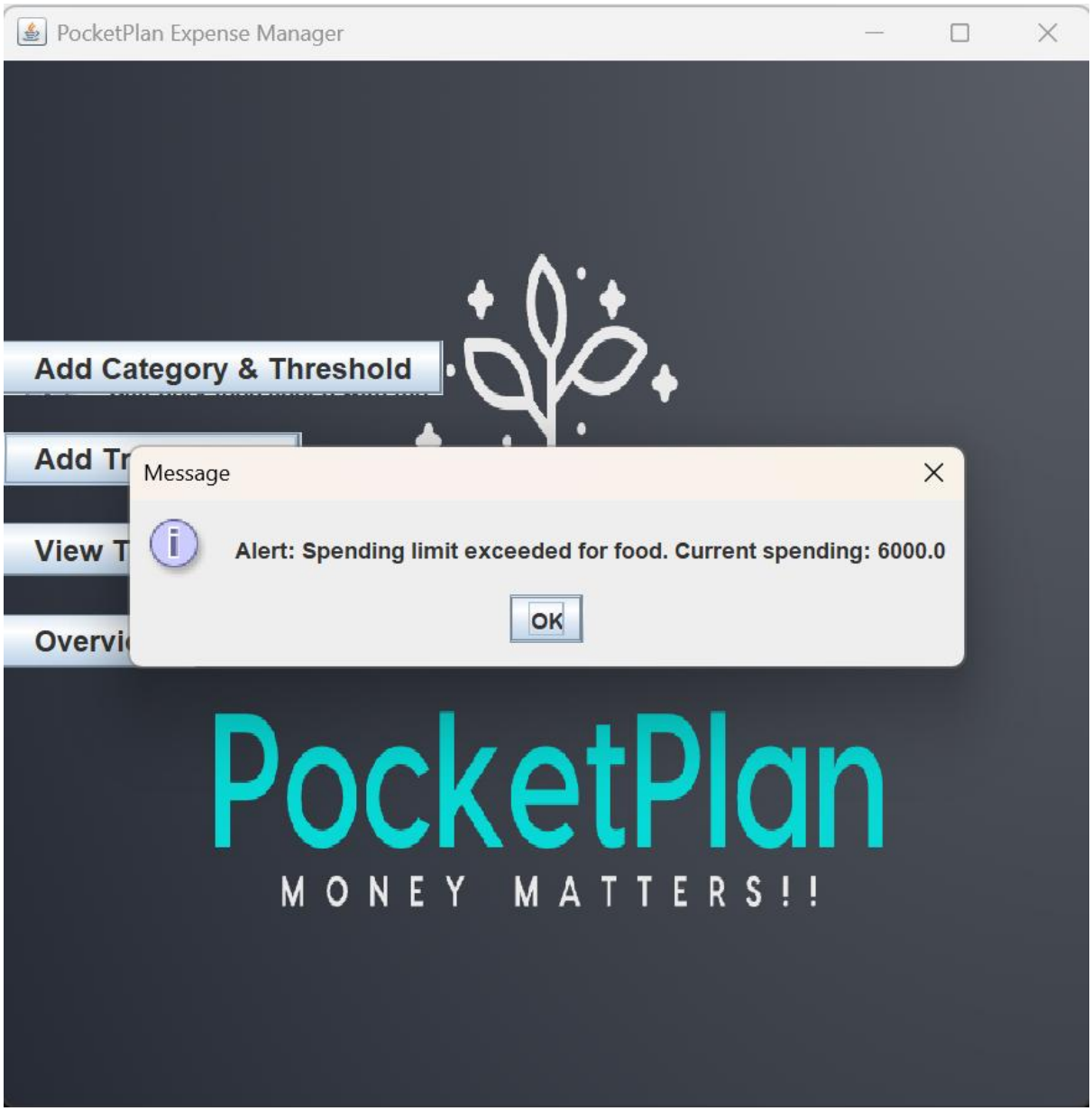

PocketPlan Expense Manager
Add Category & Threshold
Add Transaction
View Transactions
Overview
Input
Enter Threshold for food:
4500
OK
Cancel
PocketPlan
MONEY MATTERS!!

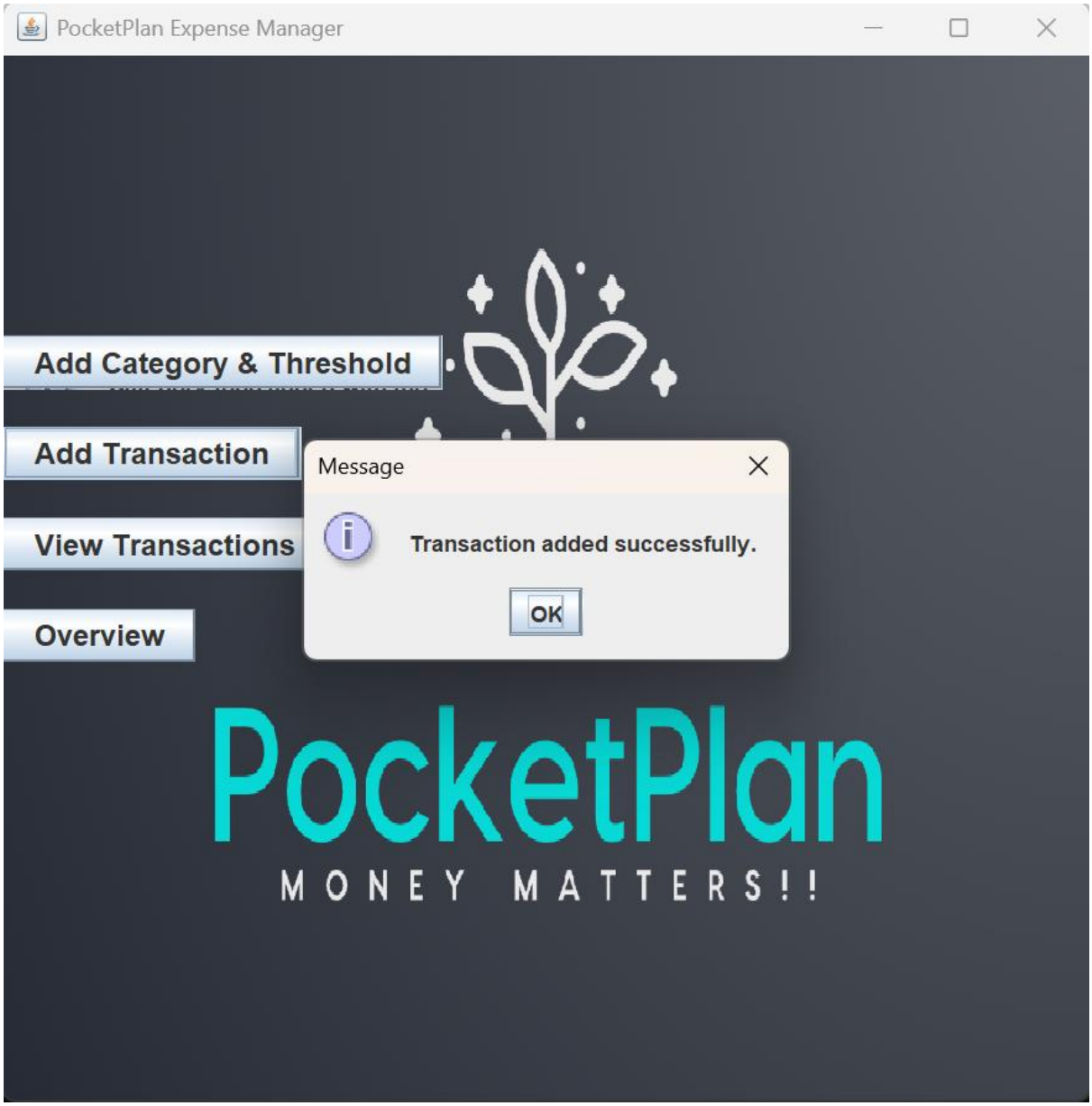

PocketPlan Expense Manager
Add Category & Threshold
Add Transactio
View Transactic
Overview
Message
Category food added with threshold 4500.0
OK
PocketPlan
MONEY MATTERS!!

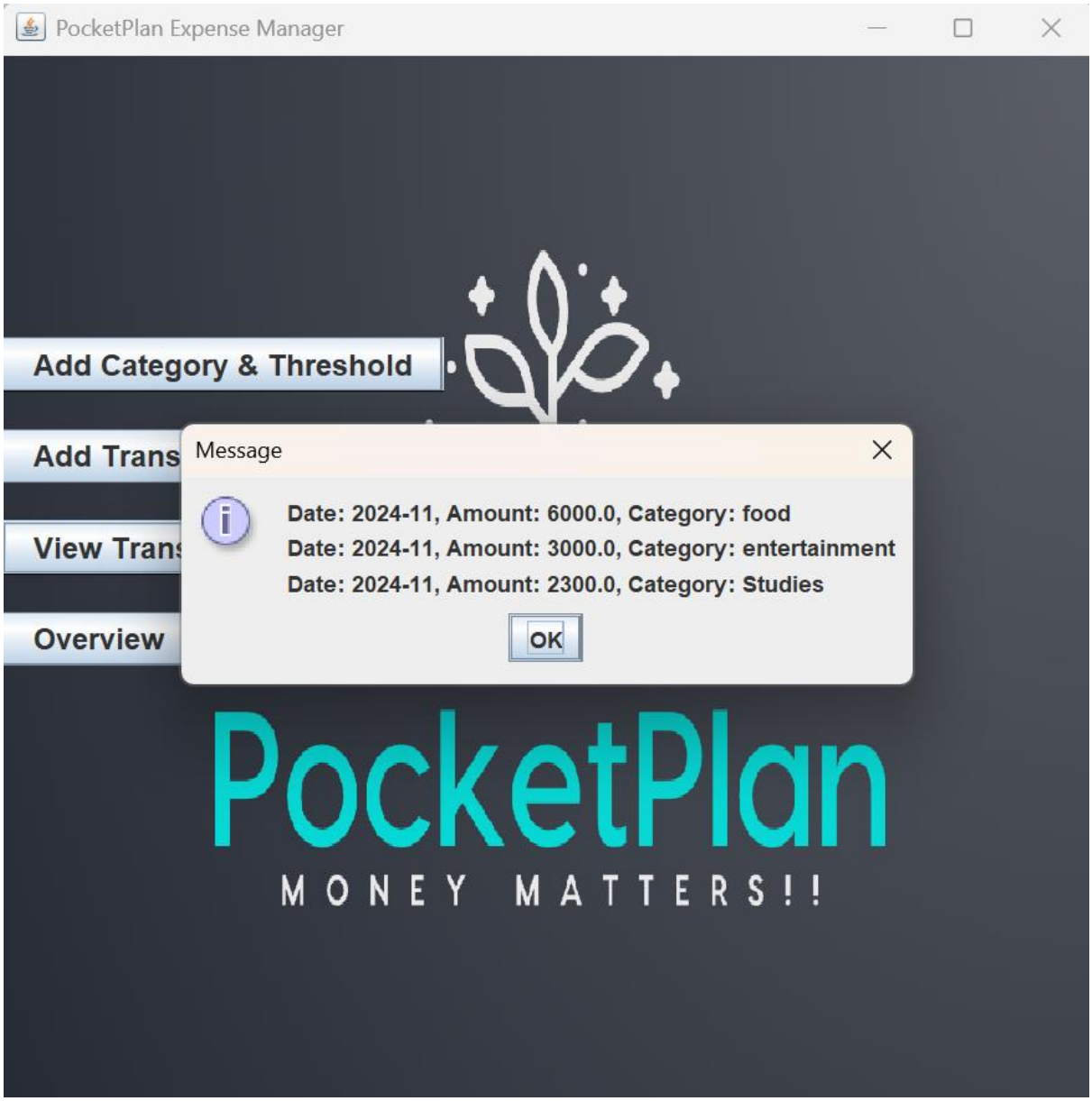

PocketPlan Expense Manager
Add Category & Threshold
Add Transactio
View Transactic
Overview
Enter Date
Month:
January
Year:
May
June
July
August
September
October
November
December
el
MONEY MATTERS!!

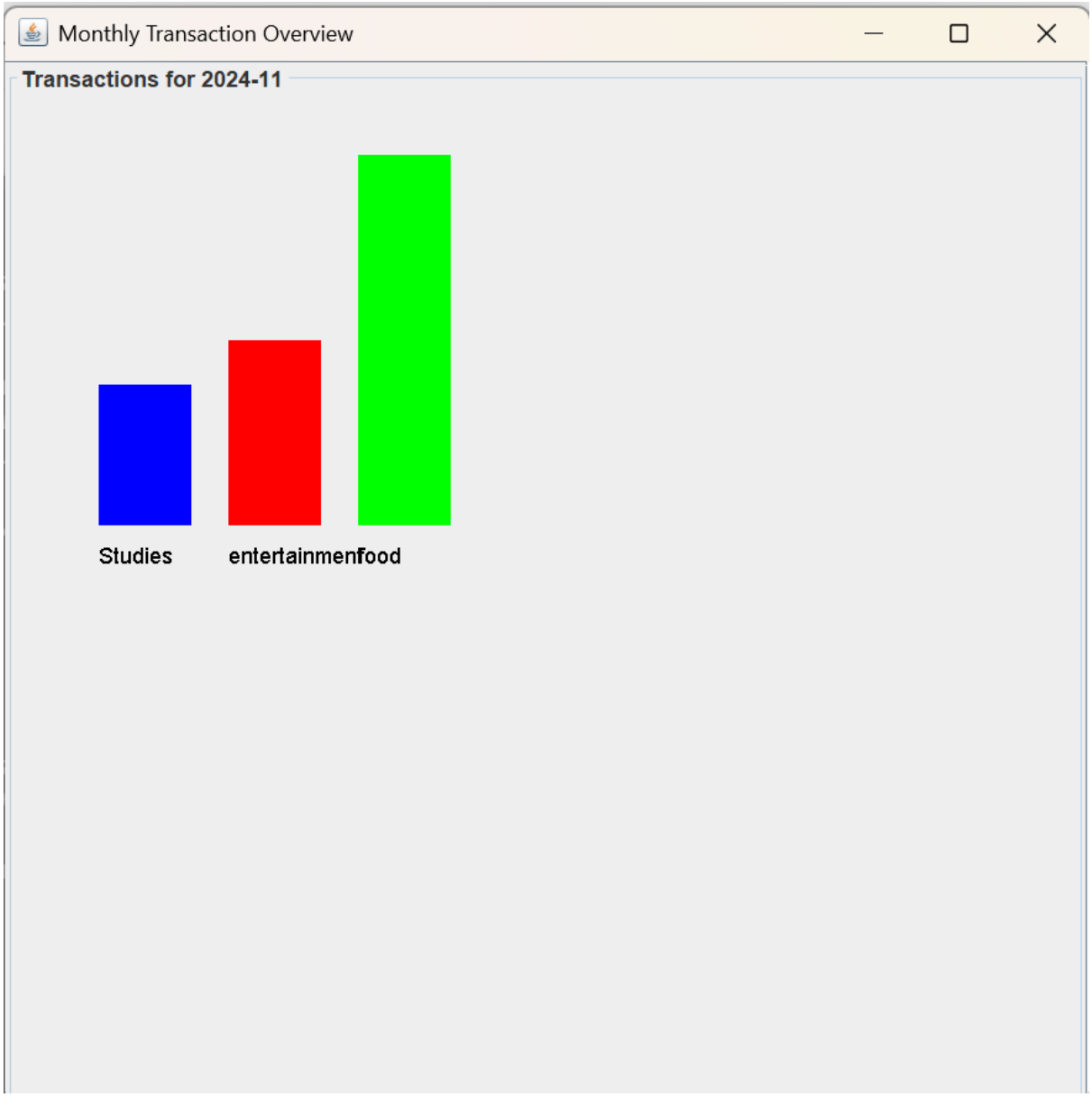

PocketPlan Expense Manager
Add Category & Threshold
Add Transaction
View Transactions
Overview
Input
Enter Amount:
6000
OK
Cancel
PocketPlan
MONEY MATTERS!!


PocketPlan Expense Manager
Add Category & Threshold
Add Transaction
View Transactions
Overview
Category
Select Category:
food
food
entertainment
Studies
PocketPlan
MONEY MATTERS!!


PocketPlan Expense Manager
Add Category & Threshold
Add Tr
View T
Overvi
Message
Alert: Spending limit exceeded for food. Current spending: 6000.0
OK
PocketPlan
MONEY MATTERS!!


PocketPlan Expense Manager
Add Category & Threshold
Add Transaction
View Transactions
Overview
Message
Transaction added successfully.
OK
PocketPlan
MONEY MATTERS!!


PocketPlan Expense Manager
Add Category & Threshold
Add Trans
View Trans
Overview
Message
Date: 2024-11, Amount: 6000.0, Category: food
Date: 2024-11, Amount: 3000.0, Category: entertainment
Date: 2024-11, Amount: 2300.0, Category: Studies
OK
PocketPlan
MONEY MATTERS!!


Monthly Transaction Overview
Transactions for 2024-11
Studies
entertainmen
food