**Name:** Ajay Jawade (2202425),

Sakshi Salunkhe (2202454),

Amrut Dhappadhule (2202417),

Krishna Limbare (2202435),

Vaishnavi Devare (2202413) **Class:** MCA-I

Expense Tracker App - Code Explanation

This document explains the Python-based Expense Tracker App developed using tkinter. The app helps users track their expenses by adding, filtering, viewing the total, and clearing expenses. It uses a text file to save and load expenses.

# Overview

The Expense Tracker app is a simple graphical user interface (GUI) program that allows users to manage their expenses. The main functionalities include: - Adding an expense with details (amount, description, category).- Displaying the total expenses.- Filtering expenses by category.- Clearing all expenses from the system.

# Main Modules

## ExpenseManager Class

The `ExpenseManager` class is responsible for managing the expenses. It handles adding, saving, loading, and clearing expenses. The expenses are stored as a list of tuples, where each tuple contains: amount (float), description (string), and category (string). The `save\_expenses` method stores the expenses in a text file, while the `load\_expenses` method loads them from the file.

## ExpenseTrackerApp Class

The `ExpenseTrackerApp` class is the main GUI for the Expense Tracker app. It initializes the GUI components using tkinter and provides functionality to the user. This includes: adding expenses, displaying total expenses, filtering by category, and clearing all expenses.

# Code Explanation

## ExpenseManager Methods

1. `**\_\_init\_\_(self, filename='expenses.txt')**`: Initializes the ExpenseManager, loads expenses from the file, and stores them in a list.

2. `**add\_expense(self, amount, description, category)**`: Adds a new expense to the list and saves the expenses to the file.

3. `**get\_total(self)**`: Returns the total amount of all expenses by summing the amounts.

4. `**get\_expenses\_by\_category(self, category)**`: Returns all expenses that match the selected category.

5. `**clear\_expenses(self)**`: Clears all stored expenses and updates the file.

6. `**save\_expenses(self)**`: Saves the expenses to the file in a CSV format.

7. `**load\_expenses(self)**`: Loads expenses from the file into the app.

## ExpenseTrackerApp Methods

1. `**\_\_init\_\_(self, root)**`: Initializes the main window and all GUI components. It sets up labels, text fields, buttons, and the display area.

2. `**add\_expense(self)**`: Handles adding an expense by retrieving values from user inputs, validating the inputs, and updating the displayed expenses.

3. `**show\_total(self)**`: Displays the total of all expenses in a pop-up message.

4. `**filter\_expenses(self)**`: Opens a filter window where the user can select a category to filter expenses and display the matching results.

5. `**clear\_all(self)**`: Clears all expenses from the system after confirming the action with the user.

# GUI Structure

The GUI is created using tkinter and follows a simple layout with various components:

1. **Labels:** Used to display text prompts for user input (Amount, Description, Category).

2. **Entry Fields:** Used to input values for Amount and Description.

3. **ComboBox:** Dropdown for selecting the expense category.

4. **Buttons:** Provide functionality for adding expenses, viewing total expenses, filtering by category, and clearing expenses.

5. **Text Area:** Displays the added expenses and the total amount.

6. **Pop-up Messageboxes:** Display error messages, success messages, and confirm actions like clearing expenses.

# How it Works

1. **Adding an Expense:** When the user enters an amount, description, and selects a category, the expense is added to the list. The expense is stored in the file and displayed in the text area.

2. **Showing Total:** The user can view the total of all expenses by clicking the 'Show Total Expense' button.

3. **Filtering by Category:** Users can filter expenses by category using a dropdown menu, and the filtered results are displayed.

4. **Clearing Expenses:** The user can clear all stored expenses after confirming the action through a pop-up message.

# Conclusion

This Expense Tracker app provides an efficient way to track personal expenses, filter them by category, and keep an overview of total spending. The code is simple yet effective, and it demonstrates basic file handling and GUI development using Python's tkinter library.