**Title** : Python programming

**Project**: Expense Tracker

**Name:** Sivarani

**Roll No:** 213H1A0503

**Code:**

**1. \*User Input and Data Management\***

**- Develop functions to capture user inputs for daily expenses.**

**- Example:**

**python**

**def add\_ expense (expenses, date, amount, description, category):**

**expenses. Append ({'date': date, 'amount': amount, 'description': description, 'category': category})**

**2. \*Data Storage\***

**- Use a list of dictionaries to store expense data or implement file handling for persistence.**

**- Example:**

**python**

**import json**

**def save\_ expenses\_ to\_file(expenses, filename='expenses.json'):**

**with open (filename, 'w') as file:**

**json. Dump (expenses, file)**

**def load\_ expenses\_ from\_ file (filename='expenses. json'):**

**try:**

**with open (filename, 'r') as file:**

**return json. Load (file)**

**except FileNotFoundError:**

**return []**

**3. \*Expense Categories\***

**- Allow categorization of expenses during input.**

**- Example:**

**python**

**categories = ['food', 'transportation', 'entertainment', 'utilities', 'others']**

**def add\_ expense (expenses, date, amount, description, category):**

**if category not in categories:**

**category = 'others'**

**expenses. Append ({'date': date, 'amount': amount, 'description': description, 'category': category})**

**4. \*Data Analysis\***

**- Implement functions to generate monthly summaries and category-wise expenditure.**

**- Example:**

**python**

**def get\_ monthly\_ summary (expenses, month):**

**monthly\_ expenses = [expense for expense in expenses if expense['date']. Starts with (month)]**

**total = sum(expense['amount'] for expense in monthly\_ expenses)**

**return total, monthly\_ expenses**

**5. \*User Interface\***

**- Create a simple text-based or graphical interface using libraries like Tk inter.**

**- Example using Tk inter:**

**python**

**import tk inter as tk**

**def main ():**

**root = tk. Tk ()**

**root. Title ("Expense Tracker")**

**tk. Label (root, text="Date (YYYY-MM-DD):"). Grid (row=0)**

**tk. Label (root, text="Amount:"). Grid (row=1)**

**tk. Label (root, text="Description:"). Grid (row=2)**

**tk. Label (root, text="Category:"). Grid (row=3)**

**date\_ entry = tk. Entry(root)**

**amount\_ entry = tk. Entry(root)**

**description\_ entry = tk. Entry(root)**

**category\_ entry = tk. Entry(root)**

**date\_ entry. Grid (row=0, column=1)**

**amount\_ entry. Grid (row=1, column=1)**

**description\_ entry. Grid (row=2, column=1)**

**category\_ entry. Grid (row=3, column=1)**

**def add\_ expense \_ callback ():**

**date = date\_ entry. Get ()**

**amount = float (amount\_ entry. Get ())**

**description = description\_ entry. Get ()**

**category = category\_ entry. Get ()**

**add\_ expense (expenses, date, amount, description, category)**

**save\_ expenses\_ to \_file(expenses)**

**tk. Button (root, text='Add Expense', command=add\_ expense\_ callback). Grid (row=4, column=1, sticky=tk. W, pad y=4)**

**tk. Button (root, text='Quit', command=root. Quit). Grid (row=4, column=2, sticky=tk. W, pad y =4)**

**root. Main loop ()**

**if \_\_name\_\_ == "\_\_main\_\_":**

**expenses = load\_ expenses\_ from\_ file ()**

**main ()**

**6. \*Error Handling\***

**- Ensure proper error handling for user inputs and file operations.**

**- Example:**

**python**

**def add\_ expense (expenses, date, amount, description, category):**

**try:**

**amount = float(amount)**

**if category not in categories:**

**category = 'others'**

**expenses. Append ({'date': date, 'amount': amount, 'description': description, 'category': category})**

**except Value Error:**

**print("Invalid amount. Please enter a number.")**

**7. \*Documentation\***

**- Provide detailed comments and a README file explaining the project, setup, and usage instructions.**

**- Example:**

**markdown**

**# Expense Tracker**

**## Overview**

**The Expense Tracker application allows users to manage and analyze their daily expenses.**

**## Features**

**- Add daily expenses with amount, description, and category.**

**- View monthly and category-wise summaries.**

**- Simple and intuitive user interface.**

**- Error handling for invalid inputs.**

**## Setup**

**1. Clone the repository.**

**2. Run `main.py` to start the application.**

**## Usage**

**- Input your daily expenses in the provided fields.**

**- Use the interface to add expenses and view summaries.**

**By following this outline, you can develop a comprehensive and functional Expense Tracker application that meets all the project objectives and**