Personal Expense Tracker - Project Specification

1. Project Overview

Description:
The "Personal Expense Tracker" is a desktop or web-based application designed to help users efficiently manage and track their daily expenses across various categories, such as food, transportation, entertainment, and utilities. Users can input their expenditures, and the tool provides visual insights into their spending habits. Additionally, the app allows users to set monthly budgets for each category, alerting them when they approach or exceed these limits. The goal is to help users gain better control over their finances by providing them with a clear understanding of their spending patterns.

External Mechanisms:

- **Data Visualization:** Python's Matplotlib or Seaborn libraries for generating visual insights like pie charts and bar graphs.
- **Database Management:** SQLite or Pandas for storing and processing user input data.

Graphical User Interface (GUI):
The application would ideally feature a simple web interface built using Flask, where users can input their expenses and view summaries. It could also have a desktop interface built with Tkinter or run in a command-line interface (CLI) for minimal implementation.

Remote Control:
In the future, the application could offer API functionality to allow remote entry of expenses or integration with other financial tools.

2. Task Vignettes (User Activity "Flow")

Task 1: Inputting Daily Expenses

Vignette:
John, a user who is trying to better manage his finances, opens the Personal Expense Tracker on his desktop. The main screen displays input fields for the date, amount, and category (e.g., groceries, entertainment). John enters $50 for groceries and $20 for transportation. He clicks the "Add Expense" button, and the data is saved. As John continues to enter his expenses throughout the week, the app begins to visualize his spending habits.

Technical Details:

- User inputs are stored in a Pandas DataFrame or SQLite database.
- Each entry includes the amount, category, and date.
- Data is saved locally and displayed in a summary table on the main screen.

Task 2: Setting and Managing Budgets

Vignette:
To avoid overspending, John decides to set a monthly budget of $200 for entertainment. He navigates to the "Budgets" section of the app and sets his limit. As the month progresses, John enters more entertainment expenses. When he reaches $180, the app alerts him that he is nearing his budget limit, suggesting that he may want to reduce his entertainment spending for the rest of the month.

Technical Details:

- Budget limits are set by the user and stored in the database.
- The application continuously monitors total spending in each category.
- Alerts are triggered when spending approaches or exceeds the budget limit.

Task 3: Visualizing Spending Trends

Vignette:
At the end of the month, John wants to review his spending habits. He clicks on the "View Reports" button, which generates a pie chart showing how his money was distributed across different categories. The chart reveals that most of his spending was on groceries and entertainment. Additionally, the app provides a bar graph comparing his spending across the last three months, helping John understand his financial trends over time.

Technical Details:

- Data visualization is handled using Matplotlib or Seaborn.
- The app generates visual reports based on user data, including pie charts and bar graphs.
- Users can view visual summaries of their spending for the current month and compare them with previous months.

3. Technical Flow

Data Flow:

- **Input:** User inputs daily expenses, budget limits, and other financial data.
- **Processing:**
    - Data is processed using Pandas for organization and summary generation.
    - Budget limits are monitored against total spending in real-time.
- **Output:** The application generates visual summaries (charts/graphs) and alerts based on the processed data.

Blocks:

- **Expense Input Block:**
  - Manages the entry and storage of daily expenses.
  - Interfaces with the database for CRUD (Create, Read, Update, Delete) operations.
- **Budget Management Block:**
  - Handles budget setting and monitoring.
  - Triggers alerts when budget limits are reached.
- **Visualization Block:**
  - Creates graphical representations of spending data.
  - Uses Matplotlib/Seaborn for chart generation.
- **GUI Block:**
  - Displays the input fields, summaries, and visual reports to the user.
  - Manages user interactions and inputs.

Data Types:

- **Input Data:** Expense entries as dictionaries or DataFrame rows.
- **Processing Data:** Summary statistics, budget status as floats/integers.
- **Output Data:** Visualizations (graphs/charts), alerts as strings or objects.

User Interaction:

- **Input:** User enters data through forms in the GUI.
- **Output:** Visual summaries and alerts are displayed to guide user decisions.

Link to GitHub Repository:
https://github.com/niharikaux/Personal-Expense-Tracker/tree/main