Personal Expense Tracker – Review Document

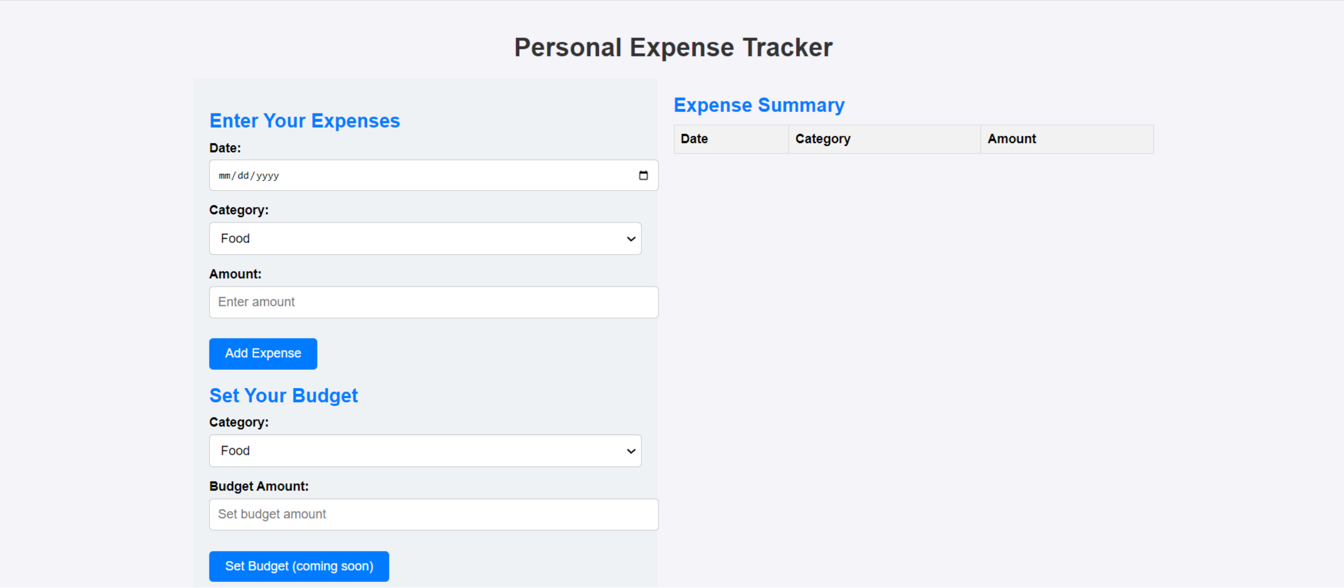
Project Overview

The **Personal Expense Tracker** project aims to provide users with a simple, web-based interface to track their daily expenses. Users can input expenses by specifying a date, category, and amount. The app dynamically displays a summary of these expenses in a table. Future features will include the ability to set budget limits for different categories and visualize spending patterns using graphs. The long-term goal is to give users a clearer picture of their spending habits, helping them to manage their finances more effectively.

Implemented Features So Far

The following core features of the project have been implemented:

- **Expense Entry Form**: Users can enter their expenses through a form that includes fields for date, category, and amount.
- **Dynamic Expense Table**: The app displays a summary of the entered expenses in a table format that updates dynamically when new expenses are added.
- **Modular Backend**: The app uses a modular structure, with core expense-handling logic stored in expense_tracker.py to keep the project organized and maintainable.
- **UI Layout**: The user interface has been adjusted based on feedback to improve user experience. The form for adding expenses is positioned on the left side, while the expense summary table is on the right side.
- **Budget Setting Placeholder**: A form for setting budgets has been added as a visual placeholder, but it is not functional yet. This sets the stage for future development of budget management features.



Personal Expense Tracker

Enter Your Expenses

Date: mm/dd/yyyy

Category: Food

Amount: Enter amount

Set Your Budget

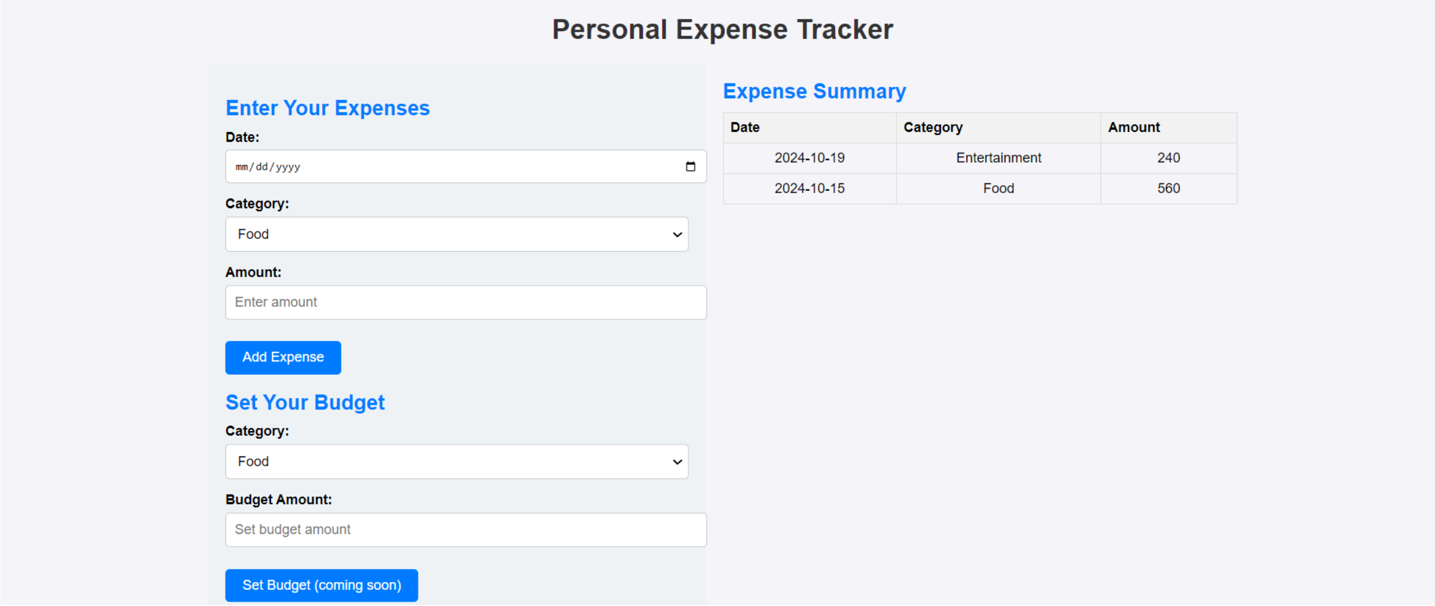
Category: Food

Budget Amount: Set budget amount

Expense Summary

Date	Category	Amount

Fig1. This is where you enter your expense detail and select category



Personal Expense Tracker

Enter Your Expenses

Date: mm/dd/yyyy

Category: Food

Amount: Enter amount

Set Your Budget

Category: Food

Budget Amount: Set budget amount

Expense Summary

Date	Category	Amount
2024-10-19	Entertainment	240
2024-10-15	Food	560

Fig2. This image shows how the user will view the expense summary in tabular format

Issues Encountered and Solutions

- **Issue 1: UI Layout and Clutter**
  - **Problem**: Initially, the form and table took up the full-page width, which made the layout feel cluttered.
  - **Solution**: Updated the layout to a left-right structure, where the form is on the left and the table on the right. This layout is more user-friendly and provides cleaner visual experience.
- **Issue 2: Module Naming**
  - **Problem**: Initially, the module managing expenses was named functions.py, which lacked clarity.
  - **Solution**: Renamed it to expense_tracker.py to better reflect its purpose. This naming convention makes the code more readable and easier to maintain.
- **Issue 3: Budget Setting**
  - **Problem**: I haven’t implemented the budget-setting functionality yet, though the placeholder is present.
  - **Solution**: This feature will be developed in the next version. For now, the form serves as a visual reminder of this future functionality, and the button remains disabled.

Unresolved Issues and Proposed Solutions

- **Budget Setting**:
  - **Current State**: The form for budget setting is a placeholder.
  - **Proposed Solution**: In the next version, I will implement the logic to allow users to set budget limits for each category and track whether they stay within those limits. This will likely involve creating a separate function in expense_tracker.py to handle budget storage and comparison against current expenses.
- **Data Persistence**:
  - **Current State**: All expenses are currently stored in memory (in a list), meaning they are lost when the app is restarted.
  - **Proposed Solution**: Implement data persistence using SQLite or file-based storage in the next phase. This will ensure that expenses and budget data are saved across sessions.

Where Help is Needed

- **Data Persistence**: I would appreciate guidance on how to implement data persistence for this project. Should I use a lightweight solution like SQLite, or would it be better to implement file-based storage for a simpler solution?
- **Graphs for Spending Visualization**: I will need assistance in selecting the best tools or libraries (e.g., Matplotlib, Plotly) for adding graphical visualizations of spending patterns. Any advice on how to integrate these into the Flask app would be helpful.

Milestones for the Next Weeks (Version 2)

1. **Milestone 1: Budget Setting Functionality**
   - **Description**: Implement the ability for users to set budget limits for each category and receive feedback when they are approaching or exceeding these limits.
   - **Dependence**: Needs to be done before spending visualizations, as they may include budget-related data.
2. **Milestone 2: Data Persistence**
   - **Description**: Add a mechanism (e.g., SQLite database or file storage) to persist expenses and budget data across user sessions. This will ensure that users' data is not lost when the app is restarted.

   - **Dependence**: Required for any future user-specific data handling (e.g., personalized budgets, visualizations).

3. **Milestone 3: Spending Visualizations**
   - **Description**: Integrate graphical representations of spending patterns (e.g., bar graphs, pie charts) using libraries like Matplotlib or Plotly. This will give users insights into their spending habits.
   - **Dependence**: Requires data persistence to pull long-term spending data.

4. **Milestone 4: User Authentication (Optional)**
   - **Description**: Implement user authentication so multiple users can track their expenses separately. This would allow users to log in and view their specific expense data and budgets.
   - **Dependence**: Requires data persistence to store user information securely.

<u>Self-Reflection</u>

- **Progress Satisfaction**: I am generally satisfied with the progress so far. The core features (expense entry and summary) are working as expected, and the project is well-structured with modular code that will make future development easier.
- **Feasibility**: I believe I can finish the project within the remaining weeks, as long as I stay on track with the milestones. The next steps are clear and achievable.
- **Challenges**: Implementing the left-right layout was a bit more difficult than expected, but after addressing the layout issue, the project is looking much cleaner and more professional.
- **Expectations vs. Reality**: Managing modular code (e.g., splitting logic into expense_tracker.py) was easier than expected, which makes future feature implementations more manageable. However, integrating data persistence and visualizations might take more time than originally anticipated.
- **Light Bulb Moment**: While working on the project, I realized the importance of planning the UI layout early. The initial cluttered design impacted usability, and after implementing the left-right layout, the user experience greatly improved. This has changed how I plan to approach UI/UX design in the future.