PROJECT SPEC

Description

My idea is to build a budget tracking app to help users have better control over spending.

The functionalities would include:

1) Create categories and add spendings/purchases to it.
2) Set a budget limiter and the app would remind the users if they're going above budget.

Backend Framework – I will be using Flask for the backend functionality as its simple and ideal for small projects.

Database – SQLite would be the apt choice for this project due to its easy accessibility and ease of use.

GUI – The project would work ideally on a Tkinter GUI.

Vignette

1) Create Categories and Add Spendings/Purchases

Emily, a college student, logs into her expense tracker app and clicks on the "Categories" tab. She sees existing categories and selects "Add Category" to create one for her coffee expenses. After entering "Coffee," she saves it, and it appears in her list. She then navigates to "Add Spending," selects "Coffee," inputs her spending amount, and adds a note. Upon saving, she receives confirmation, and her spending updates in her financial summary.

Technical Details:

- Backend: Flask handles API requests for categories and spendings.
- Database: SQLite stores category and spending data.
- Data Flow: User inputs are sent to Flask, which updates the SQLite database.
- Error Handling: Validate category names to prevent duplicates.

2) Set a Budget Limiter and Receive Reminders

Emily goes to the “Budget” section to set a monthly limit for her coffee expenses. She inputs $100 for “Coffee” and clicks “Set Budget.” As she adds purchases, the app tracks her spending. When she nears her limit, a notification pops up: “You are nearing your budget limit for Coffee. Current spending: $90. Remaining: $10.” This helps her control her spending.

Technical Details:

- Backend: Flask manages budget settings and spending tracking.
- Notifications: Sends alerts when nearing budget limits.
- Database: Budget settings stored in SQLite linked to user accounts.
- Data Flow: Budget data is stored and checked against new spending entries.

Technical Flow(Blocks)

1) **User Interface (UI)**

Function: Handles user input and displays output.
User Actions: Inputs for categories, spendings, and budgets; outputs for confirmations and notifications.
Data Types: Strings and numbers.

2) **Controller (Flask)**

Function: Processes requests and communicates with the database.

Data Flow:

Receives data from the UI.

Sends requests to the database and returns responses.

Data Types: Requests, responses (JSON).

3) Database (SQLite)

Function: Stores categories, spendings, and budgets.

Data Flow: Saves new records and retrieves existing data.

Data Types: Tables (categories, spendings), rows (records).

Example Data Flow

1) Creating a Category:
   Input: User enters category name.
   Flow: UI → Controller → Database → Controller → UI (confirmation).

2) Adding a Spending Entry:
   Input: User selects category and enters spending details.
   Flow: UI → Controller → Database → Controller → UI (update summary).

3) Setting a Budget:
   Input: User inputs budget limit.
   Flow: UI → Controller → Database → Controller → UI (budget status).

4) Notification System:
   Checks budget after each spending entry and notifies the user if nearing limits.

Link to Repository –

https://github.com/sandeep711/HCI584-Project