Project Title: Personal Expense Tracker

Concept

A simple app where users can log their daily expenses, categorize them (e.g., Food, Travel, Shopping), and view a summary of expenses for the current month. The app will use Flutter for the UI, gRPC for communication between the client and server, and a database for storing data.

Components

Frontend (UI in Flutter)

- A **Flutter app** for Android/iOS with a clean and simple interface.
- **Pages/Features**:
    1. **Home Screen**:
        - Displays the total expenses for the current month and a list of daily expenses.
        - "Add Expense" button to navigate to the form.
    2. **Add Expense Screen**:
        - A form to enter expense details:
            - **Title** (e.g., "Lunch")
            - **Amount**
            - **Category** (Dropdown: Food, Travel, Shopping, etc.)
            - **Date**
    3. **Summary Screen**:
        - Shows a bar chart or pie chart of expenses by category.

Backend (gRPC Server)

- A gRPC server built with **Python** (using **gRPC and Protobuf**) or **Go** for simplicity.
- **Services/Endpoints**:
    1. `ListExpenses`: Returns a list of all expenses for a given date.
    2. `AddExpense`: Adds a new expense to the database.
    3. `DeleteExpense`: Deletes an expense by its ID.
    4. `GetSummary`: Returns a summary of expenses grouped by category.

Database

- Use **SQLite** or **PostgreSQL** for storing expense data.
- **Schema**:

    - `expenses` table:
        - `id`: Unique identifier (UUID).
        - `title`: String.
        - `amount`: Float.
        - `category`: String.
        - `date`: Date.

Development Steps

1. **Frontend (Flutter)**:
    - Design the UI with Flutter widgets like `ListView`, `DropdownButton`, and `TextFormField`.
    - Use the `grpc` package in Flutter to make gRPC calls to the server.
2. **Backend (gRPC Server)**:
    - Define the gRPC API using Protocol Buffers (Protobuf).
    - Implement the gRPC service in Python or Go.
    - Connect the server to the database using an ORM like **SQLAlchemy** (Python) or native SQL queries (Go).
3. **Database**:
    - Create the `expenses` table.
    - Write functions for CRUD operations (e.g., fetch, insert, delete).
4. **Connect Flutter and Backend**:
    - Generate Dart gRPC stubs from the Protobuf definition.
    - Use the stubs in Flutter to call backend services.
5. **Testing**:
    - Test gRPC endpoints using **BloomRPC** or CLI tools.
    - Ensure the app correctly displays data from the backend and handles user input.

Bonus Features (Optional for Future Expansion)

- Add user authentication to track expenses for individual users.
- Enable export of expenses to a CSV or PDF.
- Provide insights like highest spending category or daily spending trends.

This project introduces **Flutter** app development, **gRPC** protocol, and backend integration while keeping the functionality straightforward.