📘 Assignment – Build a Splitwise Clone

🧩 What is Splitwise?

Splitwise helps people **track shared expenses** and figure out **who owes whom**. For this assignment, you'll create a **simplified version** that includes groups, expense splitting (equal or percentage), and balance tracking.

✅ Core Requirements

🔧 Backend (Python + FastAPI + PostgreSQL)

You need to implement REST APIs with the following features:

1. Group Management

- `POST /groups`: Create a new group with a name and list of user IDs.
- `GET /groups/{group_id}`: Get group details (name, users, total expenses).

2. Expense Management

- `POST /groups/{group_id}/expenses`: Add a new expense to a group.
  - Input: description, amount, paid_by (user_id), split_type (equal or percentage), splits.
- Support two split types:
  - **Equal**: Split equally among group members.
  - **Percentage**: Split based on percentages provided for each member.

3. Balance Tracking

- `GET /groups/{group_id}/balances`: See who owes whom in the group.

- `GET /users/{user_id}/balances`: See all balances for a particular user across groups.

Notes:

- No need for authentication/authorization.
- No payment/settlement functionality.
- Use PostgreSQL for persistence.
- Use SQLAlchemy or any ORM of your choice.

🖥️ Frontend (React + TailwindCSS)

Build a simple UI to:

- Create a group with users.
- Add an expense to a group.
- View group balances (who owes whom).
- View personal balance summary.

The UI should call your REST APIs and present the data cleanly with TailwindCSS.

🌟 Bonus (Optional): LLM-Powered Chatbot

Add an AI chatbot to your frontend that can answer natural language queries like:

- “How much does Alice owe in group Goa Trip?”
- “Show me my latest 3 expenses.”
- “Who paid the most in Weekend Trip?”

Implementation Tips:

- Use OpenAI or HuggingFace API.
- Send user query + structured context from your DB to the LLM.
- Return clean, formatted answers to the user in the chat UI.

📦 Deliverables

- ✅ GitHub Repository with:
  - `backend/` (FastAPI app)
  - `frontend/` (React app)
  - `docker-compose.yml` for full stack setup
  - 15 min loom video link (explain the system design, trade offs and working demo)
- ✅ README.md with:
  - Setup and run instructions with simple command
  - API documentation (OpenAPI or manually listed endpoints)
  - Any assumptions made
- ✅ (Optional) Deployed version or screen recording link