AI Intern Assessment Task

Part 1: Prompt Engineering + AI Microservice + Streamlit UI

Objective

Build an AI-powered web prototype that:
- Accepts user input
- Generates AI-based responses using prompt engineering
- Stores all interactions in a PostgreSQL database via a Python microservice
- Presents responses and history via a Streamlit UI

Core Functional Requirements

1. Prompt Engineering & AI Model Usage

Create a prompt-based function that, given a user query, produces two different outputs:
- One using a casual or creative prompt style
- One using a formal or analytical style

Examples:
- For a topic like 'climate change', generate:
  - A casual summary
  - A formal academic explanation

Use either:
- OpenAI API (GPT-3.5/4)
- HuggingFace models (e.g., T5, Falcon)
- Or any other public LLM (can mock if API access isn't available)

Optional bonus: Chain two prompts together for refinement (e.g., generate → polish → summarize).

2. Microservice Backend (FastAPI or Flask)

Design a backend with the following API endpoints:

POST /generate
- Accepts: { "user_id": "abc123", "query": "Explain blockchain" }
- Returns: { "casual_response": "...", "formal_response": "..." }
- Saves input and both outputs to Postgres (include timestamp, user ID)

GET /history?user_id=abc123
- Returns all past interactions for the given user in reverse chronological order

Database Schema Suggestion (PostgreSQL):
Table: prompts
- id (UUID)
- user_id (TEXT)
- query (TEXT)
- casual_response (TEXT)
- formal_response (TEXT)
- created_at (TIMESTAMP)

3. Streamlit UI

Build a simple interface with:
- A text box for user input
- A 'Generate' button
- Two sections displaying the AI responses
- A collapsible/history sidebar showing all past queries and results for the user

Design Tips:
- Make UI clean and readable
- Optional: Add dropdown to select tone/style

Bonus Features (Optional)

- Authentication mock (just use user_id field across frontend/backend)
- Loading spinner for async response
- Streamlit form validation
- Environment-based configuration (.env)

Testing

- Write unit tests for:
  - Prompt formatting logic
  - AI generation logic (mock responses)
  - API route validation
- Write at least one integration test that simulates a complete flow

Deliverables

- Github Codebase with clear structure and documentation
- README.md with setup steps, tech used, and explanation of prompt strategies
- requirements.txt or pyproject.toml
- .env.example file
- hosted demo link