You are required to complete one of the following technical assessments that best reflects your skills and interests. The goal is to demonstrate your ability to design and implement a secure, scalable, and maintainable back-end system, particularly in the context of AI product development.

You may use boilerplate code and AI-generated UI to save time, but all key logic should be written and structured by you.

Tech Stack: Python (FastAPI or Flask)

Focus: Secure API Development, AI Integration, System Design

Submission: Public GitHub Repository

Assessment 01

✅ **Objective**

Design and implement a secure back-end system for an **AI-Powered Text Assistant** that allows users to:

- Authenticate securely
- Submit prompts for AI-based processing
- View their request history and summaries
- Get usage analytics (per user)

---

1. Secure User Authentication

- Implement user registration & login with **JWT or OAuth 2.0**
- Password hashing required
- Token expiration & refresh mechanism recommended
- Secure all endpoints to be **user-specific**

---

2. AI Prompt Processing API

- Implement /ask-ai endpoint:

    - Accepts a prompt and returns a **response** from OpenAI/Gemini (e.g., “Summarize this text”)
    - Record each prompt, response, and timestamp in the database
- Include a **"daily limit per user"** (e.g., max 10 requests/day)

---

3. AI Usage History & Analytics

- Create endpoints:
    - /history: Show a user’s previous prompts & AI responses
    - /analytics: Return basic usage stats (e.g., # of prompts used today, avg response time, etc.)

Assessment 02

✅ **Objective**

Build a secure backend for a **"Smart Document Assistant"** web app. Users can:

- Upload documents (text/PDF)
- Ask questions about the content
- Receive LLM-generated answers
- View and search Q&A history

🧱 **Tasks**

1. User Authentication (30–45 min)

- Implement secure login/signup using JWT or OAuth 2.0
- Users can only see their own data

---

2. Document Upload & Processing

- Create /upload endpoint (accept .txt or .pdf)
- On upload:

    - Extract text (use any library for parsing)
    - Store file metadata & extracted content in DB
- Securely store uploaded content (local or mocked cloud storage)

---

3. AI Question Answering

- Implement /ask endpoint:
    - Input: question + document ID
    - Output: AI answer
    - Save prompt, response, latency, and tokens used (if available)

---

4. User History & Search

- Endpoint: /history
    - Returns a list of all questions & responses by the user
- Optional: add keyword search/filtering by document title or question

---

5. Security & Best Practices

- Input validation/sanitization
- Auth-protected endpoints
- Abuse prevention (rate-limiting, quotas)
- Demonstrate secure file handling (e.g., prevent malicious uploads)