Assignment: Comprehensive Backend and ML Application Development

**Date posted:** 23 November, 2024

# SUBMISSION LINK: <https://forms.gle/JE8PQjZnYXXZwRmu8>

# Objective

The assignments are designed to evaluate the candidate's proficiency in backend API development, database handling, prompt engineering, large language models (LLMs), and SQL-based data analysis.

## Part 1: CRUD Operations API Development

### Database Schema

You are provided with the following database schema:

| CREATE TABLE users ( id SERIAL PRIMARY KEY, name VARCHAR(255) NOT NULL, email VARCHAR(255) UNIQUE NOT NULL, created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ); |
| --- |

| CREATE TABLE orders ( id SERIAL PRIMARY KEY, user\_id INT REFERENCES users(id), product\_name VARCHAR(255) NOT NULL, quantity INT CHECK (quantity > 0), order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ); |
| --- |

### Requirements

* **API Development:** Build an API using **FastAPI** that performs CRUD operations for both users and orders tables.
* **Endpoints:**
  + POST /users - Create a new user.
  + GET /users/{id} - Fetch user details by ID.
  + PUT /users/{id} - Update user information.
  + DELETE /users/{id} - Delete a user by ID.
  + Similar endpoints for orders.
* **Concurrency:** Use **Uvicorn/Gunicorn** with multiple workers to handle concurrent requests efficiently.
* **Error Handling:** Implement robust error handling for:
  + Invalid input.
  + Database constraints (e.g., unique email violation, foreign key errors).
  + Resource not found.
* **Security:** Ensure the API uses **SSL** for secure communication.
* **Concurrency and Asynchronous Operations:**
* Use **async/await** for database and I/O operations to maximize performance.
* **Testing:** Write test cases for each API endpoint to ensure reliability.

**Deliverable:** A fully functional API with documentation and test results.

## Part 2: Data Processing Pipeline with APIs

### Objective

Integrate OpenAI or Gemini API and build a custom data processing pipeline.

### Requirements

* **Setup the Pipeline:**
  + Use **OpenAI** or **Gemini API** to process text data.
  + The input is raw, unstructured text, and the output should be structured JSON with specified fields.
* **Prompt Engineering:**
  + Design a well-defined prompt for the API to ensure accurate outputs.
  + Use **Pydantic** models to validate the API response structure and ensure data integrity.
* **Local Model Integration:**
  + Set up a locally hosted LLM (e.g., LLaMA or similar) and process the same data pipeline using it instead of the external API.
  + Compare the outputs of the external API and the locally hosted model.
* **Error Handling:**
  + Handle API failures, rate limits, and invalid responses gracefully without relying on built-in batch-processing commands.

### Deliverable

* Pipeline code with prompt engineering and validation scripts.
* Comparison report of outputs from external and local models.

## Part 3: Metrics Extraction and Automation from SQL

### Objective

Use PostgreSQL to analyze and extract insights from search and clicks data, then automate the process.

### Database Schema

| CREATE TABLE search\_clicks ( search\_id SERIAL PRIMARY KEY, search\_query VARCHAR(255), clicks INT DEFAULT 0, impressions INT DEFAULT 0, click\_through\_rate FLOAT, search\_date DATE DEFAULT CURRENT\_DATE ); |
| --- |

### Requirements

* **Metrics Analysis:** Write SQL queries to:
  + Calculate the **average click-through rate (CTR)** for each day.
  + Identify the **top 5 search queries** with the highest CTR over a given period.
  + Detect **queries with high impressions but low clicks** (possible optimization candidates).
* **Pipeline Automation:**

Develop a Python script to execute these SQL queries and store the results in a summary table:

| CREATE TABLE search\_insights ( id SERIAL PRIMARY KEY, insight\_date DATE, average\_ctr FLOAT, top\_queries JSONB, low\_performance\_queries JSONB ); |
| --- |

* Automate the script to run daily using a scheduler like **cron** or **Celery**

### Deliverable

* Use Supabase free tier to do above assignments.
* Make dummy dataset for all the above tasks.
* SQL queries and the summary table structure.
* Python script for pipeline automation.
* Sample output of the pipeline with mock data.

Asynchronous and Synchronous [Rohan Sidankar](mailto:rohansidankar@hushh.ai)