**Software Engineer Take-Home Assignment**

**Overview**

You have been tasked to create an MVP application to facilitate the utilization of LLMs. The MVP comprises:

1. **Task 1:** A backend component that interacts with the LLMs
2. **Task 2:** A frontend component that presents LLM data to the users

**Assignment Requirements**

1. You can choose to submit either **Task 1** or **Task** **2**; additional consideration will be given to candidates who complete both tasks.
2. You must use Git for version control. Submit your solution through a Git repository of your choice and share the repository link with us.
3. Do show the progress of your work via atomic Git commitments
4. You must provide a README.md containing instructions on how to build and run your solutions locally.
5. You must demonstrate good engineering practices in developing the app. Eg. TDD.

**Task 1**

Create a backend component with the following features:

1. CRUD conversations that contain a history of queries and responses from an LLM (OpenAI GPT 3-turbo).
2. Send prompt queries and receive responses from the LLM.
3. Prompts need to contain the existing conversation history as context.
4. All prompts sent to the LLMs and the responses returned need to be anonymized and stored in a database for auditing purposes.

OpenAPI documentation for this backend component containing detailed explanations is provided as a separate .yaml file (you can upload the file and view it at [editor.swagger.io](https://editor.swagger.io/)). If you decide to deviate from the design provided, please explain the rationale in the README.md. You will need to utilize the following as part of your implementation tech stack, relevant links are provided below.

1. Python >= 3.8
2. [FastAPI](https://fastapi.tiangolo.com/)
3. [Pydantic](https://docs.pydantic.dev/2.5/)
4. [Beanie](https://beanie-odm.dev/)
5. [OpenAI Python Client](https://platform.openai.com/docs/api-reference?lang=python)

Additional consideration will be given to candidates able to utilize these technologies in their tech stack

1. [Docker](https://docs.docker.com/)
2. [MongoDB](https://www.mongodb.com/docs/)
   1. [MongoDB docker image](https://hub.docker.com/_/mongo)

**Task 2**

Create a frontend component to chat with an LLM containing the following features:

1. CRUD a conversation with an LLM.
2. Send and display prompts in one conversation to the backend in **Task 1** (or using API mocking service).
3. Display LLM properties.
4. Update LLM properties.

Additional features to enhance the user journey/experience will be a bonus.

You will need to use the following as part of your implementation tech stack, relevant links are provided below.

1. NodeJS >=18.17
2. [NextJS 13](https://nextjs.org/docs/app/api-reference/components)
3. [React 18](https://react.dev/reference/react)
4. [Mantine 6](https://v6.mantine.dev/)
5. [React-Query 4](https://tanstack.com/query/v4/docs/react/overview)

Additional consideration will be given to candidates able to utilize these technologies in their tech stack

1. [Docker](https://docs.docker.com/)

Additional instructions below:

1. OpenAPI documentation for the backend component containing detailed explanations is provided as a separate .yaml file (you can upload the file and view it at [editor.swagger.io](https://editor.swagger.io/)).
2. You may use an API mocking service (e.g. <https://beeceptor.com/>) for your backend.
3. You can use ChatGPT to help generate mock data for testing!