Norm Ai Full Stack Engineer Exercise

Technical Portion

Context: Norm Ai is developing innovative, AI-powered solutions to support compliance teams working in highly regulated industries. The engineering team works closely with internal legal experts to design and build the product. Most day-to-day fullstack engineering tasks involve working with an existing part of the codebase and this take home exercise mimics that workflow. Working with legal data requires highly curated RAG pipelines–we cannot blindly load documents into a vector store and expect acceptable results upon retrieval.

Objective: You will be provided access to a github repository that contains several existing services. The repository contains a list of laws taken from the fictional series "Game of Thrones" (no prior knowledge of the series is required for successfully completing this exercise!) saved as a PDF file. It also contains a minimal NextJS app in the `frontend` folder which you can ignore during part 1. Your task is in two parts. The first part is to implement a new service and provide access to that service via a FastAPI endpoint running in a docker container. The second part is to subsequently use that service to build a light frontend that makes calls to your endpoint.

Part 1: Backend Requirements (2 hours)

The Westeros Capital Group is struggling to understand the wide variety of laws and regulations that govern the kingdom. Norm Ai let them know that with modern technology, we could build software that allows them to interface with the laws in natural language.

- Begin by forking this GitHub repository and setting up your environment.
- Familiarize yourself with the structure of the starter code. Pay attention to the provided classes and methods in **`utils.py`** and the basic FastAPI setup in **`main.py`**.
    - Notice the dependencies and libraries used, such as **`pydantic`**, **`qdrant_client`**, and **`llama_index`**, and understand their roles.

1. Implementing **`DocumentService`** in **`utils.py`**:

- Your task is to process the **`docs/laws.pdf`** and create a list of **`Document`** objects from it.
    - Implement the **`create_documents()`** method. This involves reading the PDF, parsing the text into meaningful sections, and creating **`Document`** objects that encapsulate these sections and their content.
    - The parsing logic should accurately identify and separate different laws or sections within the PDF, ensuring the data structure aligns with the **`Document`** class requirements.

2. Enhancing **`QdrantService`** in **`utils.py`**:

- This step requires completing the query method.
- Implement logic to initialize the query engine, run the query with the provided string, and format the results into an **`Output`** object containing the query, response, and relevant citations.
- Ensure that **`self.k`**, which determines the number of similar vectors to return, is effectively used in the query processing.
- Feel free to complete this section however you would like, but one option is the **`CitationQueryEngine`** from **`llama_index`**.

3. Setting up the FastAPI endpoint and containerization using Docker:

- Create an API endpoint that accepts a query string and returns a JSON response.
- This endpoint should interact with the **`QdrantService`** to process the query and return the results.
- Ensure the output is correctly serialized using the **`Output`** class from **`pydantic`**.
- Use Docker to containerize the application. Feel free to modify the existing Dockerfile to suit any changes made during development.

4. Running the service:

- Update the **`README.md`** to include instructions on how to set up and run the application.
- It is sufficient to tell the user to query the application using the provided swagger documentation, just be sure to clearly describe the required steps.
- Document any assumptions, design choices, and important details that would help in understanding and evaluating the completed exercise.

Part 2: Frontend Requirements (1.5 hours)

After demonstrating the effectiveness of what you build in Part 1, the Westeros Capital Group is very excited and interested in potentially becoming a customer. They have asked us to build out a proof of concept for how this experience could live on the existing Norm Ai product.

1. Familiarize yourself with the existing code.

- In the `frontend` folder you'll see the NextJS app folder where routing is handled. Briefly read the **page.tsx** file.
- Run the client side locally. You can start in the **`README.md`** to run it and can make edits starting in **page.tsx** to see the changes reflected on the client.

2. Building a frontend experience:

- Think about how the user might want to have the output of the server visualized. Consider any metadata the API could include.
- Document any assumptions, design choices, and important details that would help in understanding and evaluating the completed exercise.
- Note that you are not expected to build out a full product. We're looking to see how you approach writing React – how you think about things like information visualization, componentization, etc… Please do not go beyond the expected time.

Remember, the focus is on mimicking the workflow of working with an existing codebase and integrating various components to build a full-stack solution. *You are encouraged to use GenAI tools (e.g., perplexity, GH co-pilot, ChatGPT, etc) when completing the above sections, but we expect you to understand your code.*

Note: this exercise was developed in early 2024, so some of the original LlamaIndex imports in the repository aren't really valid due to updates to the library. We expect the candidate to make the necessary updates to the python files to either handle the old version or migrate to a new version.

Reflective Response Section (0.5 hours)

Please write a short response to the following question:

What unique challenges do you foresee in developing and integrating AI regulatory agents for legal compliance from a full-stack perspective? How would you address these challenges to make the system robust and user-friendly?

Your response should reflect your current ideas and we request that you refrain from using GenAI tools for this section (but feel free to use any other resource). We have included a summary of Norm's mission below, which should be helpful as you think through your response.

Our mission:

Norm Ai is automating compliance processes, making them more efficient, cost-effective, and accurate than ever before, while also ensuring democratic guardrails for AI in autonomous roles. By converting complex regulations into intelligent AI programs, we enable compliance teams to operate with unprecedented speed and precision. Our vision extends beyond just assisting compliance teams; we aim to enable the integration of AI agents into daily life, ensuring that AI-driven business processes adhere to legal and societal norms through adoption of our Regulatory AI agents as oversight. At Norm Ai, we're committed to aligning AI with public policy, reflecting our society's collective will, and ushering in a new era of regulatory intelligence and societal-AI alignment.