Hiring - Chatbot

You are designing a chatbot that returns the best engineers for a given set of instructions provided to the chatbot. The chatbot should interface with the user and return results dynamically based on the information provided.

Information you should support for the search:

- Whether the worker is full-time or part-time
- Whether the worker fits the user's budget
- Whether the worker has the skills that the user wants (i.e. Python, React, AWS)
- Other queries in natural language (i.e. "worked at a big tech company")

Example queries:

- "I want to hire someone with experience in Python and Node. My budget is $10000 a month."
    - The chatbot follows up asking whether the user wants a full-time or part-time worker after showing some results
- "I want to hire someone who worked at a big tech company. I have an unlimited budget. They should be proficient in Python."
    - The chatbot follows up asking whether the user wants a full-time or part-time worker after showing some results
- "I want to hire a developer"
    - The chatbot follows up asking for the skills, budget, and whether the worker is part-time or full-time. The chatbot shows results after skills are provided

This search has two components: a scalar part (part-time / full-time, budget, and skills, which are all stored in the database) and a semantic part (supporting all other queries in natural language, which will require embeddings and a vector database).

Your search should be as low latency as possible. You will likely need to utilise various DB tricks to make this happen.

You should also build out a front-end to make testing this chatbot easier. Additionally, your search should still work properly if users from the database are deleted or if their resume information is mutated (which requires giving some thought to the connection between Pinecone and the SQL database).

Lastly, you should design a ranking algorithm to prioritize how the candidates are returned in the search. This should take into account the following:

- Their background (work experience, education, etc.)
- Github data (pre-processed and scored from their Github username). Note: these are not the actual Github usernames of the individuals, they are randomly selected public Github usernames.

It is entirely up to you how you implement this ranking and how granular you want to be with scoring each component of a candidate's background. It is also up to you how you want to prioritize each score when returning candidates.