ByteGenie FullStack Developer Test

The objective of this test is to create an NLP application that allows users to interact with documents using natural language to perform various document related tasks, such as

- question answering on documents;
- writing abstracts or short summaries for an entire document;
- identifying key topics in the documents;
- categorising document text passages into pre-defined categories;
- summarising documents by topics of interest;
- extracting quantitative information from documents;
- etc.

The text data for this project can be found in "text_segments.csv", which contains actual text segments extracted from a few company documents. The data contains ('doc_name', 'pagenum', 'text') columns, where doc_name is a unique document name (PDF document in this case).

This text is extracted from company annual reports from the last few years, so it will contain text about nature of business, revenue, approach to tax, business risks, liquidity risks, emission reduction plans, growth forecasts, impact of Covid19, etc.

The solution should consist of a python API and a react or NextJS UI. The UI should use the python API as the backend. The python API should preferably use Flask or FastAPI, but other frameworks are also acceptable. Ideally, the API should be containerised in a Docker container, so that it can be run on any machine by just running the container—however, the containerisation is preferred but optional.

Since there are several document-related tasks a user might want to perform on their documents (as described above), in the interests of time and effort, this test application does not need to support all these tasks. It is fine to pick one or two of the tasks that you find most interesting (e.g. question-answering and writing document summary, or text classification and summarising, etc.), and focus on performing them well.

The purpose of the test is not so much to do everything, but to see how you can combine various backend and front-end tools to build an application that can perform at least one task well.

Some examples of the tools you might find useful for this task would include:

- HuggingFace model repository for specialised models related to text classification, text summarisation, and question answering;
- sentence-transformers library to generate sentence embeddings;
- Embedding models to create text embeddings;
- Vector databases to store embeddings;
- Python Flask library to build an API;

- LLM APIs (e.g. OpenAI and DeepInfra) to perform live NLP tasks.

Please submit your application as one or more GitHub repos, which should contain all the code needed for running the application.

The API GitHub repo should include a README.md file that summarises:
- main steps and motivation for any data engineering/processing on the raw data before making it available to the API;
- the main functionalities of the API;
- the key challenges you faced in solving the problem;
- how would you improve the backend, if you had more time to work on it?

The UI GitHub repo should include a README.md file that summarises:
- key functionalities of the UI;
- the API endpoints the UI uses;
- the key challenges you faced in building the front-end;
- how would you improve the front-end, if you had more time to work on it?

If you have any questions, please feel free to send them to majid@byte-genie.com