bounce Insights

Welcome to our **AI / Machine Learning Engineer** coding challenge!

In this challenge, you will create a web application that implements a Retrieval-Augmented Generation (RAG) backend system to analyse and compare two provided market research reports. Your task is to build a frontend that communicates with a Python backend running FastAPI. The application should allow users to explore, analyse, and cross-compare the reports using AI-powered interactions via natural language. ***All results must be double clickable / provide source data.***

Challenge Overview:

Your task is divided into three main parts:

1. **Python Backend**: Develop a server using Python and FastAPI that acts as an intermediary between your frontend and the RAG system. Your server should handle requests from the frontend, process the reports, interact with the language model, and return the results to your frontend.
2. **RAG System**: Implement a Retrieval-Augmented Generation system that processes the reports, retrieves relevant information based on user queries, and generates insightful responses. Consider using techniques like vector embeddings, semantic search, and prompt engineering to enhance the quality of the generated insights.
3. **Frontend**: Create a simple user interface that allows users to interact with the reports and AI-generated insights. The design and functionality are up to you – be creative! Your frontend should make requests to your backend server to fetch data and AI-generated responses.

Datasets:

You will be working with two provided reports. Your task is to allow users to explore and compare these reports using AI-powered analysis powered by a RAG pipeline.

Feel free to manipulate / abstract the reports into other formats that may be more suited for an optimised / improved RAG workflow + user experience. You have the freedom to choose how to process and utilise these reports however it must employ a RAG based approach. Be creative and build something that showcases your skills in AI/ML and data analysis!

We're looking for code that is well-structured, readable, and efficient. You should use best practices for both frontend and backend development, and make use of any relevant libraries and tools that you feel will make your work easier. Good luck!

Mandatory Technologies:

- Python
- FastAPI
- A language model of your choice (e.g., OpenAI's GPT models, Hugging Face's models)
- Any frontend technology

Evaluation Criteria:

- Creativity and uniqueness of the RAG implementation
- Quality of AI-generated insights and comparisons
- Backend architecture and RAG system integration
- Rendering / displaying the source to backup any results provided by system
- Frontend design & UI/UX
- Error handling and edge cases
- Loading state management
- Code structure, quality, and best practices
- File/repository organisation
- README.md clarity and completeness
- Deployment of the application (e.g., Vercel, Render, Heroku)

Bonus Points:

- Implementing advanced query understanding and processing
- Visualisation of insights and comparisons
- Responsive design for various screen sizes
- Performance optimization of the RAG system
- Integration of additional AI features (e.g., sentiment analysis, topic modelling)
- Testing (e.g., Jest, React Testing Library for frontend; pytest for backend)
- Additional features that enhance the user experience and data analysis capabilities

Submission Guidelines:

You have **2 weeks from receipt of the email** containing the coding challenge to complete your submission. Reply to the email in which you received this coding challenge with the following:

1. A link to the deployed application where it's running live.
2. A link to the GitHub repository containing your source code, which must include:
   - A comprehensive README.md describing how to set up, start, and use the application
   - Example file structure:
     - ├── frontend/
     - ├── backend/
     - └── README.md

Good luck, and we look forward to seeing your creative RAG-powered survey analysis tool!