This is a technical assessment designed to assess various skills needed for our projects. Since our stack mostly involves Python, FastAPI, React, docker, Pinecone (vector database), and OpenAI's API. The goal is to evaluate your skills in backend development, frontend development, integration of APIs, docker setup and database handling.

Assessment Outline

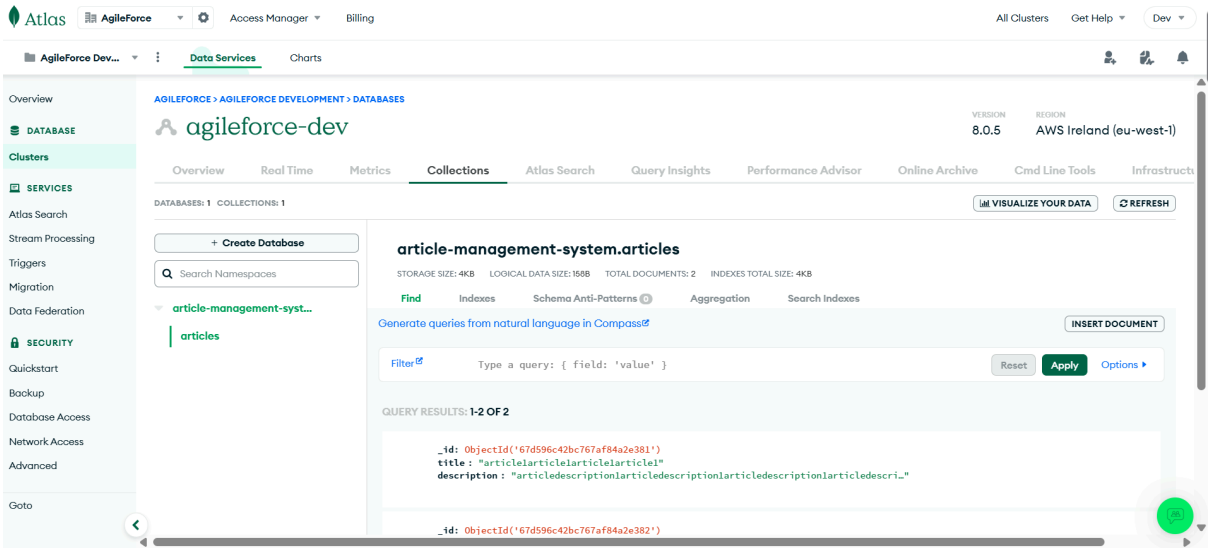
1. Backend Development (FastAPI Python & Integration with MongoDB)
2. Frontend Development (React, HTML, Tailwind CSS)
3. Docker setup and Build with docker
4. Integration with OpenAI API
5. Working with Vector Database (Pinecone)

1. Backend Development

- Build simple RESTful APIs using FastAPI to handle basic CRUD operations.
- Create an API for managing a list of "articles" stored in mongoDB.
- A snapshot of mongoDB collection is attached which clarifies how it looks and the connection string is provided as well.
- Each article has an _id, title and content.

db: article-management-system, **collection:** articles

```
mongo_uri = "mongodb+srv://agileforcedev:lUCbGVdKlWUU7nph@agileforce-dev.z3e1f.mongodb.net/?retryWrites=true&w=majority&appName=agileforce-dev"
```



Implement the following endpoints:

1. GET /articles/ - Retrieve a list of articles.
2. GET /articles/{id} - Retrieve a specific article by ID.
3. POST /articles/ - Create a new article.
4. PUT /articles/{id} - Update an article by ID.
5. DELETE /articles/{id} - Delete an article by ID.

2. Frontend Development

- Create a basic React frontend with HTML and Tailwind CSS to interact with the above APIs.
- Develop a simple user interface that allows users to view, add, edit, and delete articles.
- Display a list of articles with a title and a truncated version of the content that is clickable.
- Include a form for creating and updating articles with an edit option with each article and one create button on top.
- Add basic error handling and form validation (e.g., empty fields).
- There should be a simple query area where we can query to pinecone. [details in pinecone section]

3. Docker setup and Build with docker

- Include Docker configuration to containerize your FastAPI backend and MongoDB connection**.** Add dockerfile, requirements.txt and docker-compose.yml i.e. All container required files for it.

  Dockerfile:

  FROM python : x.y

  WORKDIR /app …

  Docker-compose.yml:

  version: "x.y"

  services:

  api:

  build:

4. Integration with GPT-4o API

- Implement a basic integration with the OpenAI GPT-4o API using Python.
- Extend the backend from Section 1 to add an endpoint [POST /articles/{id}/summarize] that generates a summary for an article using the GPT-4o API.

- It should be an option against a list of articles to generate a summary of article content and then it should show the summary in pop-up.
- Use a dummy API key for the OpenAI API assessment and we will replace it with a real one and it should work.

5. Working with Vector Database (Pinecone)

- Demonstrate basic usage of Pinecone for vector storage and search.
- Extend the backend to add functionality for storing article embeddings and searching for similar articles.
- Create an endpoint to generate embeddings for the article content (use a simple text embedding model from open ai).
- It should be an option against a list of articles to generate embeddings and then it should do so.
- Similar to Open AI, use a dummy API key for the Pinecone assessment and we will replace it with a real one and it should work.

- Implement these endpoints:

  POST /articles/{id}/embed: Generate and store article embeddings.

  GET /articles/search: Find articles similar to a given query.

Evaluation Criteria:

- Code organization, best practices, and structure
- Effective error and exception handling
- Use of Python data models when appropriate
- Database integration and communication
- Strong understanding of React concepts, including components, state management, and hooks
- Backend API integration and response handling
- UI/UX principles, including layout, form validation, and responsive design
- Managing asynchronous requests efficiently
- Familiarity with vector databases
- Basic knowledge of embeddings and similarity search