Technologies (1)

- **Docker**
  - Both the backend (FastAPI) and the frontend (Streamlit) are containerized using Docker
  - Containerization speeds up prototyping, development, and deployment
  - Containerization also provides consistency across different environments
  - The system can also be easily deployed on the cloud by either (i) cloning the github repository to a cloud's VM and building and running the container, or (ii) pulling the Docker image from Docker Hub and running the container.
- **FastAPI**
  - Is used in the backend to provide a service-oriented architecture
  - Has good asynchronous capabilities and ease of use.
  - Can be used to create RESTful APIs quickly and efficiently
- **Streamlit**
  - Streamlit is used for the frontend for fast prototyping
  - When we have more time, we can rebuild a better frontend using, for example, JavaScript Frameworks and Libraries

Technologies (2)

- **Qdrant for the vector database**:
  - Qdrant is used as the vector database for its efficient handling of embeddings.
  - In this project, Qdrant Cloud, a managed service, is used for faster prototyping
- **Embedding**
  - OpenAI Embedding Models / APIs were chosen (versus, for example, Google Generative AI Embedding Models / APIs).
- **Question Answering**
  - For Question Answering, OpenAI ChatGPT Models / APIs were also chosen over Google Generative AI Models / APIs for the same reason.
- **Cost consideration**
  - OpenAI's APIs are paid services, while Google Generative AI Models / APIs are free of charge.

OpenAI

OpenAI

Backend

- **Docker:** We use containerization to facilitate development and deployment
- **OpenAI Embedding Models/APIs**: Generate embeddings for text chunks and queries (using OpenAI's embedding models such as "text-embedding-3-small" , can be changed in *src/config/settings.py*)
- **Vector Database (Qdrant Cloud)**: Stores and retrieves embeddings efficiently.
- **Backend (FastAPI)**: Manages query processing, embedding retrieval, and communication between components.
- **OpenAI ChatGPT Models/APIs**: Generate responses based on the query and the retrieved text chunks (using chatgpt models such as "gpt-3.5-turbo", can be changed in *src/config/settings.py*)

Vector database - Qdrant

- Qdrant is a high-performance vector search engine designed to manage and search through large amounts of high-dimensional data.
- It is particularly useful for applications in machine learning, artificial intelligence, and data analysis.
- Key features of the Qdrant Vector Database include:
  - **High Performance**: Optimized for handling large-scale datasets efficiently.
  - **Scalability**: Capable of horizontal scaling to manage increasing data volumes.
  - **Accuracy**: Uses advanced indexing techniques to ensure precise similarity searches.
  - **Integration**: Easily integrates with machine learning workflows and data pipelines.
- A vector size of 1536 and cosine similarity are used for this project.

Vector database – Qdrant Cloud

- Qdrant Cloud offers a fully managed service for the Qdrant Vector Database, providing users with a scalable, secure, and cost-effective solution for their vector search needs.
- Key benefits of Qdrant Cloud include:
  - **Managed Service**: Eliminates the need for infrastructure management, allowing users to focus on their applications.
  - **Ease of Use**: Simplifies the deployment and scaling of vector search solutions.
  - **Security**: Provides robust security features to ensure data integrity and privacy.
  - **Cost-Effective**: Utilizes a pay-as-you-go pricing model to optimize costs based on usage.
- Qdrant Cloud was chosen versus locally deployed Qdrant for this project.