# RAG-based AI Application that answers questions about yourself!

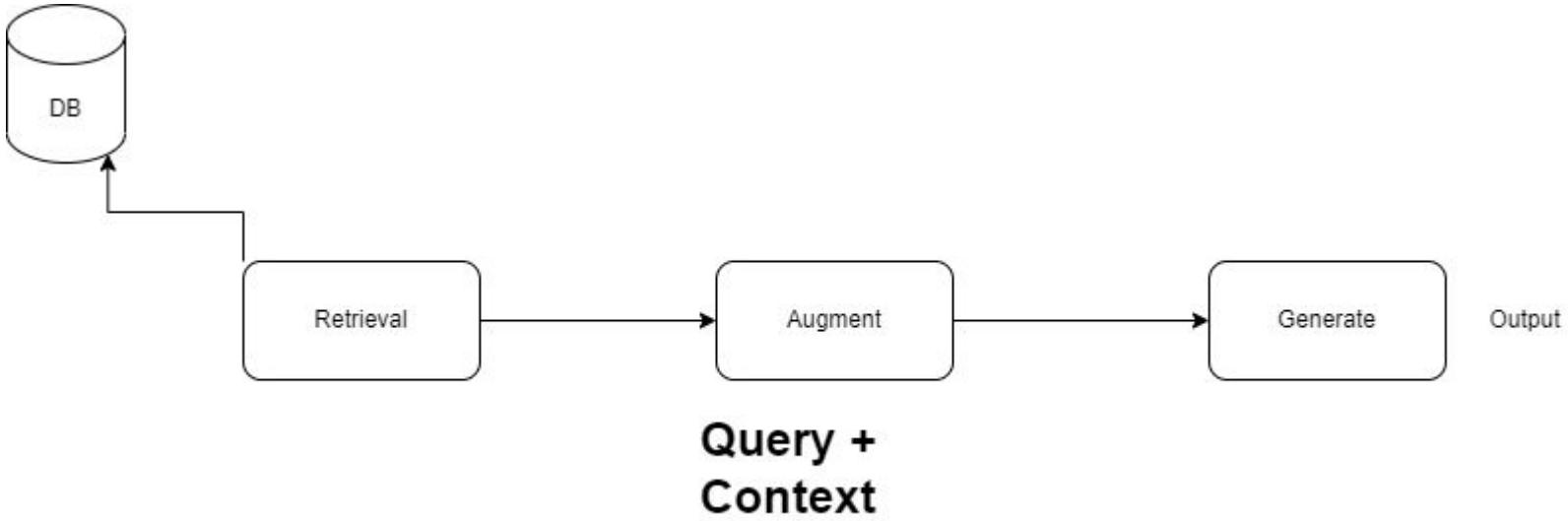
# Problem Objective

Develop a Retrieval-Augmented Generation (RAG) based AI system capable of answering questions about yourself. The system should handle inquiries in English  and manage follow-up questions effectively.

# Solution Pipeline

- Data ingestion
- RAG using Langchain/ LLamaIndex

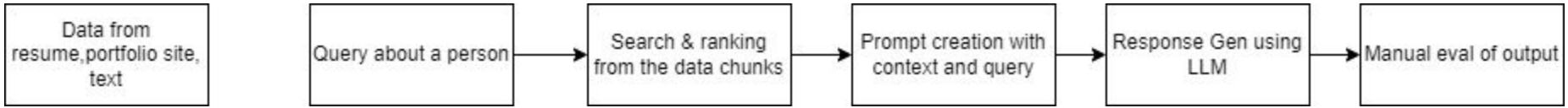
# High level system design

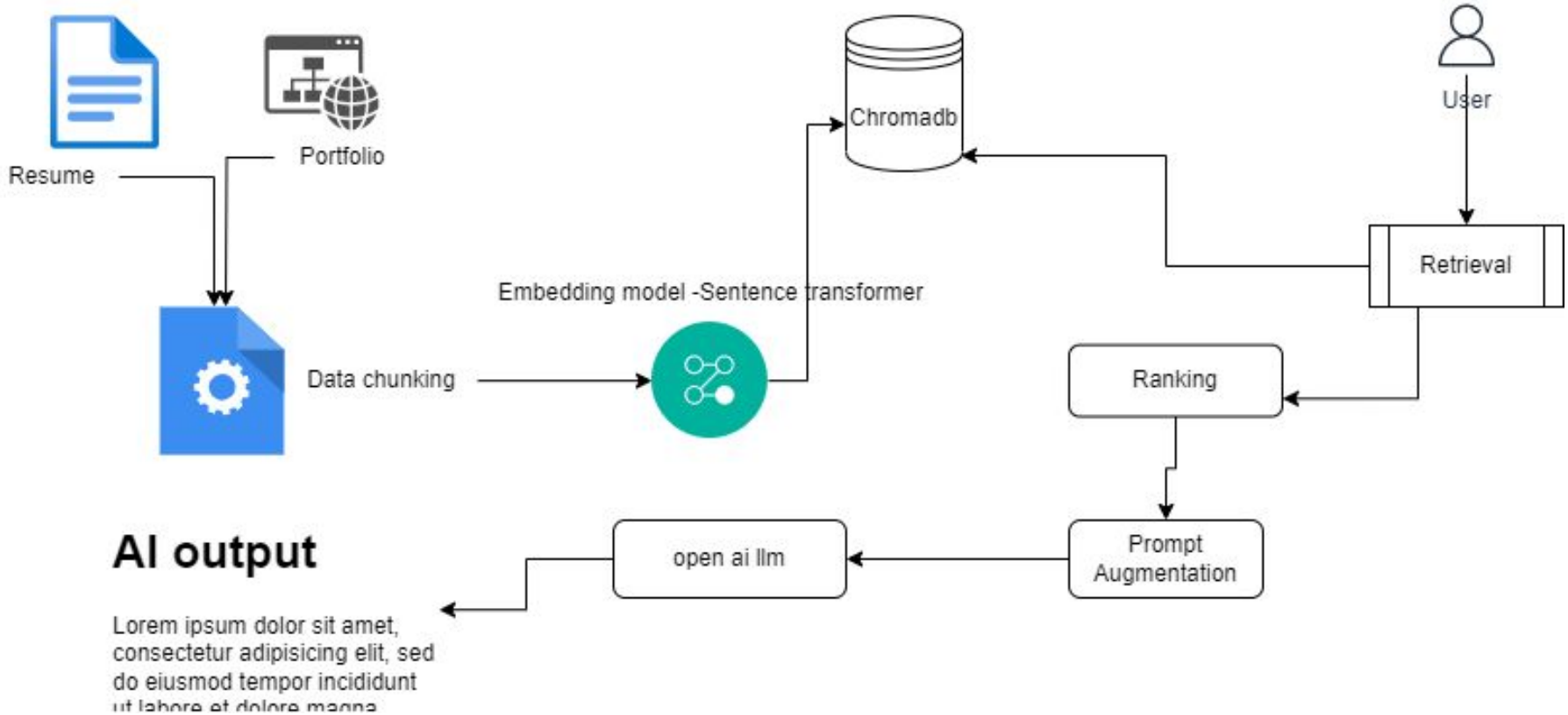


# High level system design

1. **Retrieval Engine**: Searches vast databases for relevant data using advanced algorithms, ensuring retrieval of the most pertinent information.
2. **Augmentation Engine**: Integrates retrieved data with input queries, enriching context for generating more informed responses.
3. **Generation Engine**: Formulates coherent and contextually relevant responses based on augmented input, utilizing sophisticated language models to enhance understanding.

# High level architecture




Resume
Portfolio
Chromadb
User
Retrieval
Embedding model -Sentence transformer
Data chunking
Ranking
Prompt Augmentation
open ai llm
AI output
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

# How the system works?

- We collect user data like - resume, their portfolio site and use langchain based document loader, web based loader to read the contents of the user data.
- Then we use langchain text splitter to split the text documents to multiple chunks with overlaps
- The chunks of data are converted to embeddings using fastembed/sentence transformer and uploaded to local chroma db.
- We initialize a retriever using the vector db and keyword based search(BM25) and combine as hybrid retriever.

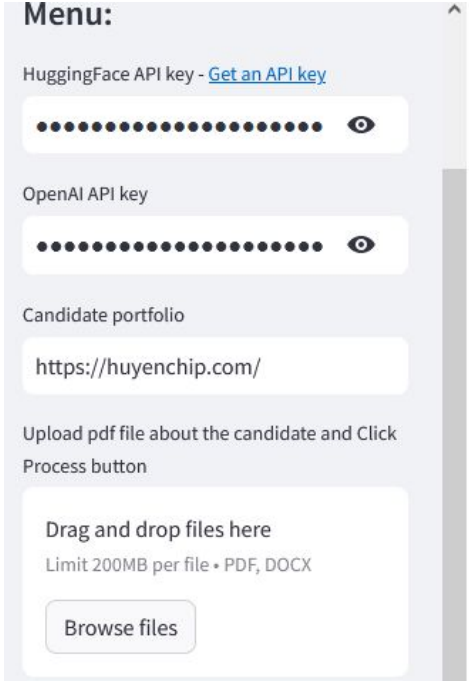
# How the system works?

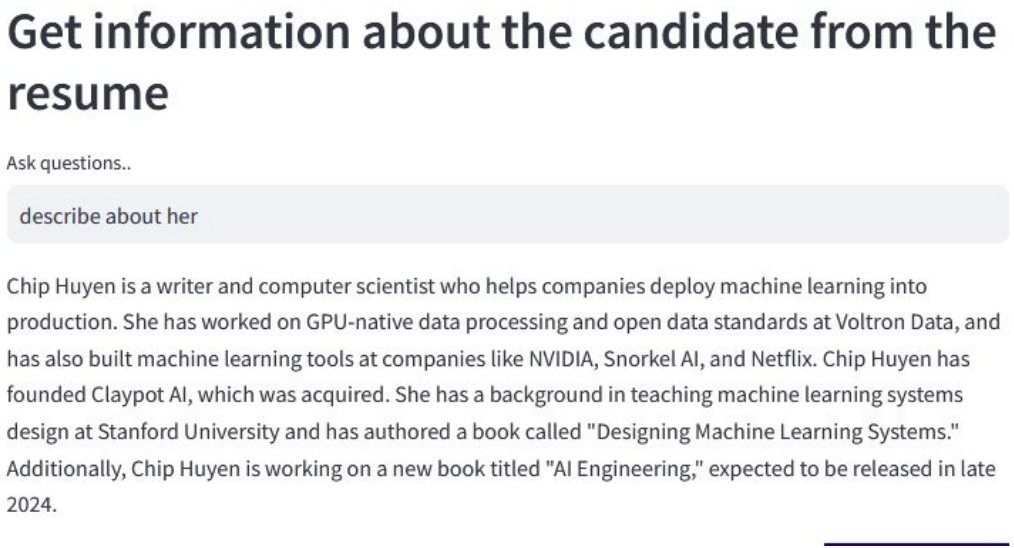
- We then use fastrank reranker to rerank the results from retriever.
- We then create a retrieval chain with memory to remember the pervious contexts so that the user can follow up with the questions
- When the user enters a query, the query is converted to embeddings and using retriever - we search based on vector similarity and keyword similarity and return the list of context which are relevant to the user query
- This context and query with memory history is augmented to prompt and passed to LLM for result generation.

# Future experiments

1) Try different embeddings model
2) Play around with the reranking model,num of documents to be retrieved
3) Use LLM to eval if the retrieval for the query is correct or not
4) After the output generation, store the results as dataframe and use another LLM to eval the results. Exploring phoenix by arize ai.

# Screenshots



Thank you