DATA+AI SUMMIT
BY databricks

Enhancing Audit Efficiency at Hapag-Lloyd with Generative AI

Ulrich Daniel
Michael Shtelma
Tania Sennikova

Presenters

Ulrich Daniel
Director of Corporate Audit Analytics at Hapag-Lloyd AG

Michael Shtelma
Lead Specialist Solutions Architect at Databricks

Tania Sennikova
Sr. Solutions Architect at Databricks

Agenda

1. Hapag-Lloyd company overview, challenges, and scale
2. Optimizing corporate audit at Hapag-Lloyd
3. Generating findings & executive summary
4. Chatbot for process documentation
5. Whats next

Equipping our fleet of 1.6mn containers with real-time tracking devices

Optimizing Corporate Audit

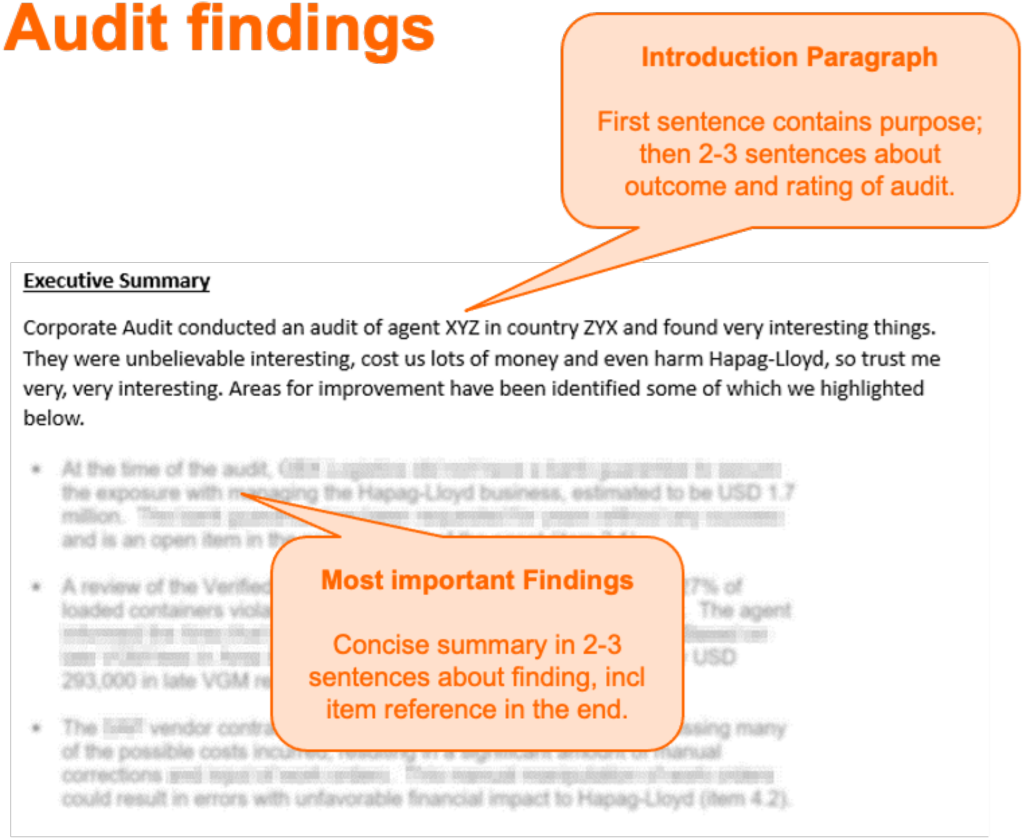
Executive Summaries and Audit findings

What?

Executive Summaries are a comprehensive text of a longer text/report mainly targeted towards company leads.

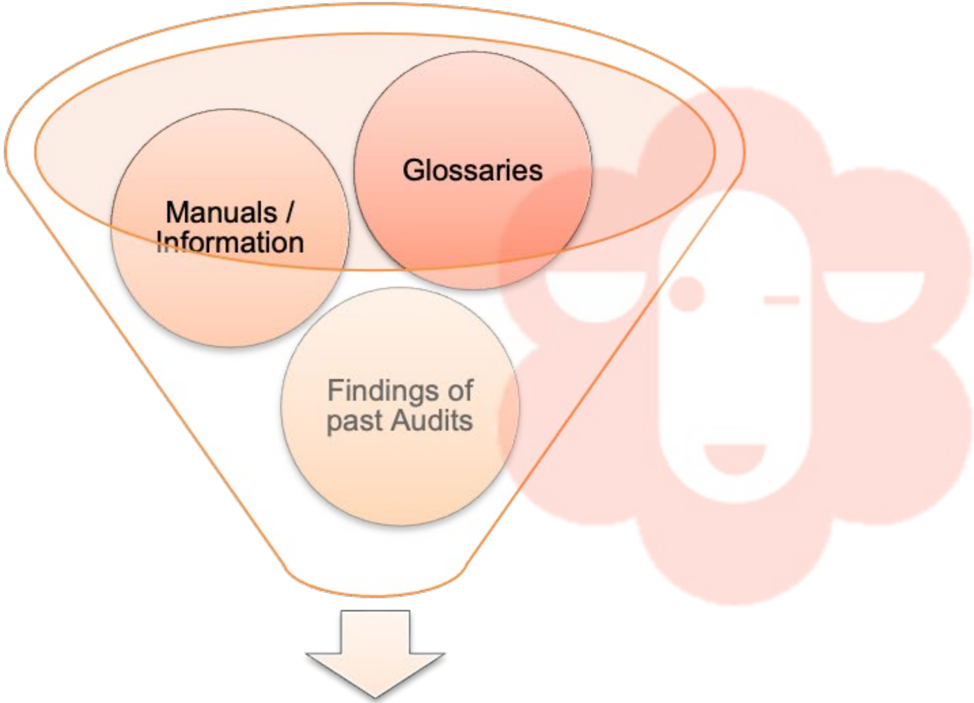
Why?

They have the purpose to give a concise overview to a (complex) topic to enable readers to make informed decisions.



The idea was born – using an LLM

- Write findings in the style of existing ones in audit reports
- Create an abstractive summary of given findings - the executive summary
- Q & A on process related questions and recommendations



Findings + Executive Summary

Generating findings & exec summary

Project stages

Define Problem

- Define the business problem
- Create evaluation dataset
- Define metrics

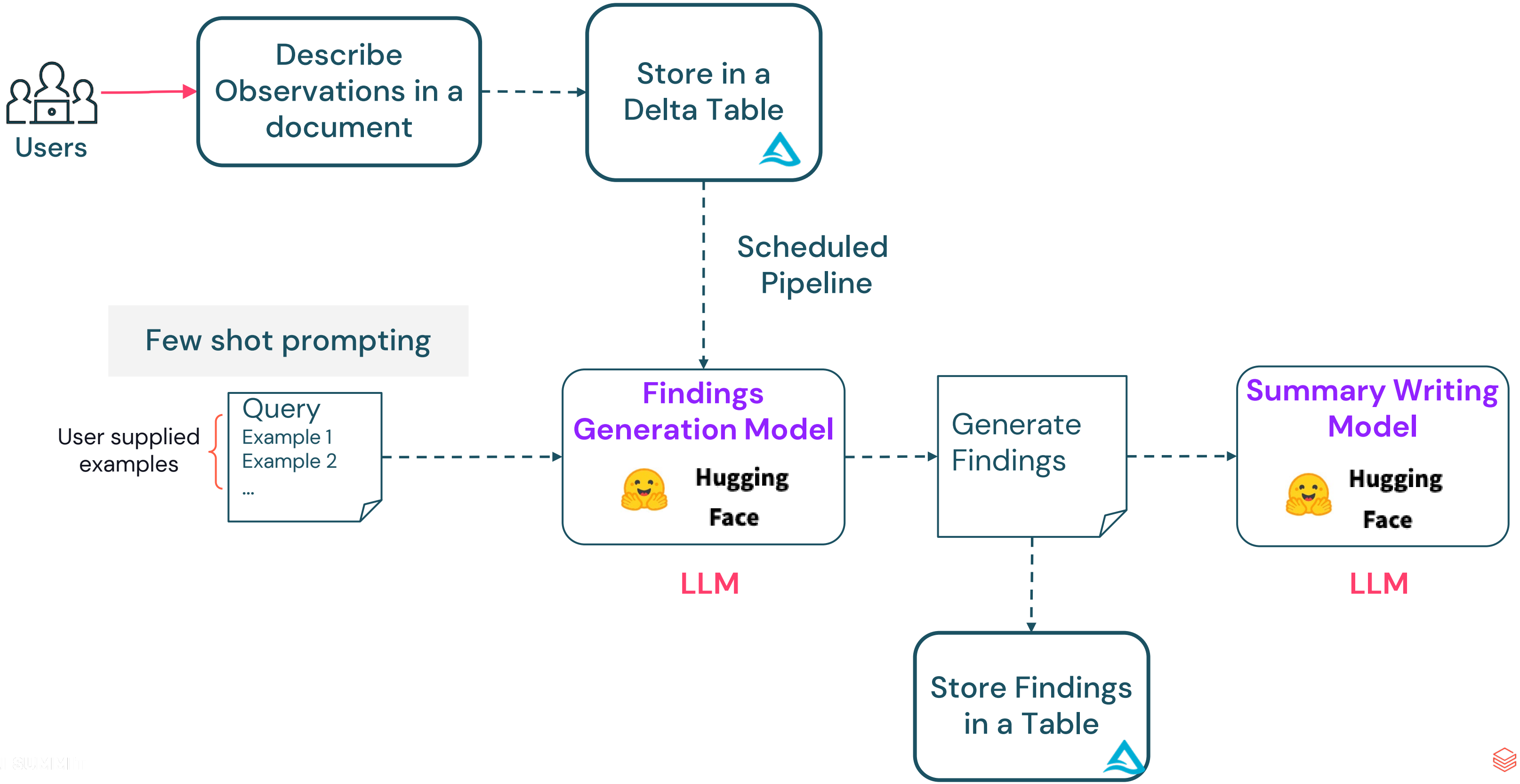
Model Adaptation & Evaluation

- Data preparation
- Choose a base model
- Prompt engineering
- Evaluating results

Deployment & application integration

- Model deployment
- Chat interface development

High Level Architecture



Modeling

Models we have tried

- Started MPT 30b
- First version used Llama 2 70b
- After that we have switched to Mixtral
- Now using DBRX

Modeling

Introducing DBRX:

- DBRX is Databricks' very own open source LLM
- DBRX is a transformer-based decoder-only LLM that was trained using next-token prediction
  - DBRX was pretrained on publicly available online data sources
  - It was trained on 12T tokens of carefully curated data and a maximum context length of 32k tokens
- DBRX Architecture:
  - Fine-grained sparse mixture-of-experts (MoE) model architecture
  - 132B parameters and supports context up to 32K tokens
  - While the model has 132B total parameters, only 36B of them are used for any given input when training, fine-tuning, or performing inference on the model

Evaluation

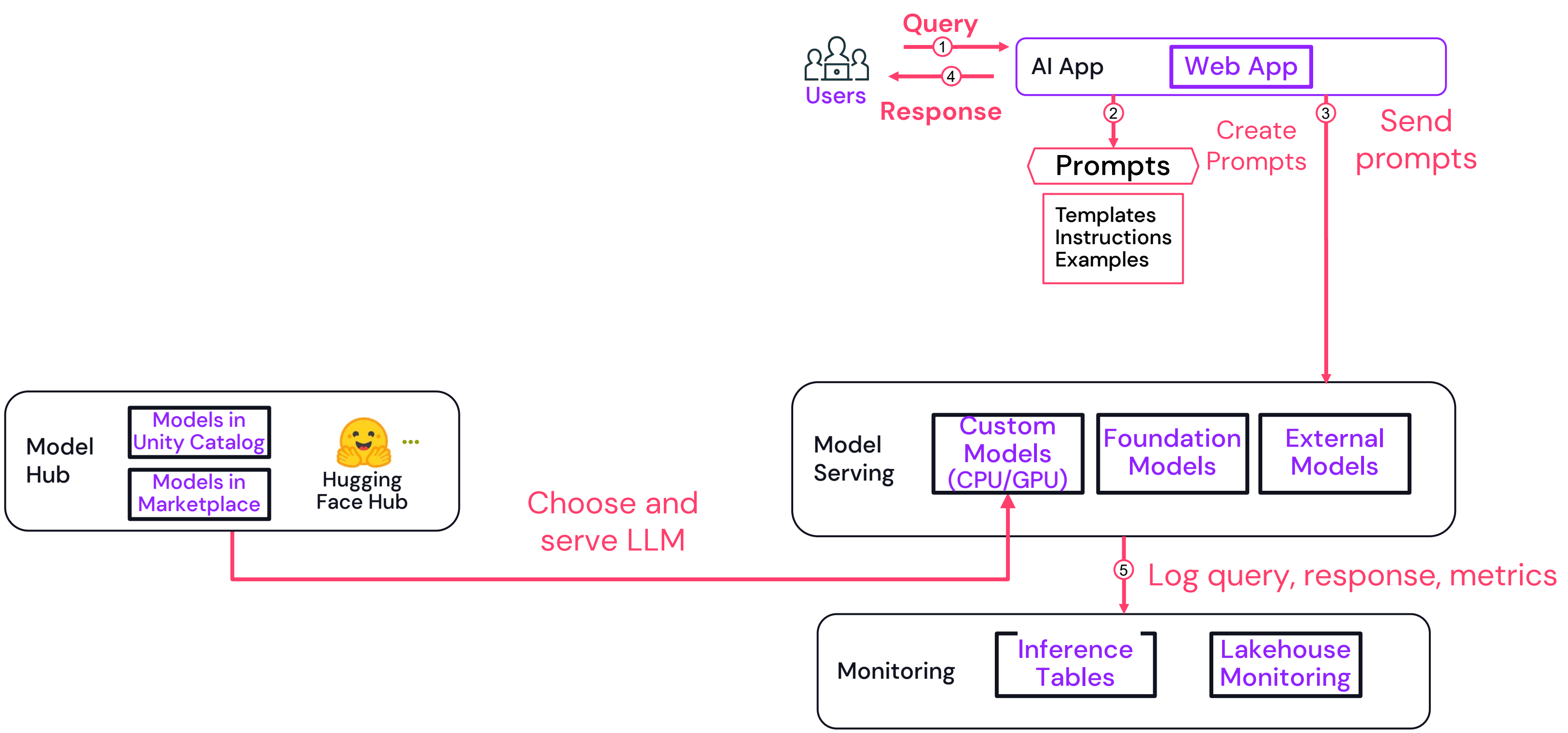
Our evaluation journey

- We have started with batch generation of the findings
  - Domain experts receive a big CSV with input and output.
  - In our case with bullet points and recreated findings
  - This approach is very time-consuming
- At first we had no bullet points: We have generated them using an LLM
- Now moving to an automated approach
  - LLM as a Judge uses another big LLM to evaluate the results
  - Supported in MLflow: `mlflow.evaluate`
  - We can define custom metrics: we need to provide several examples and prompts

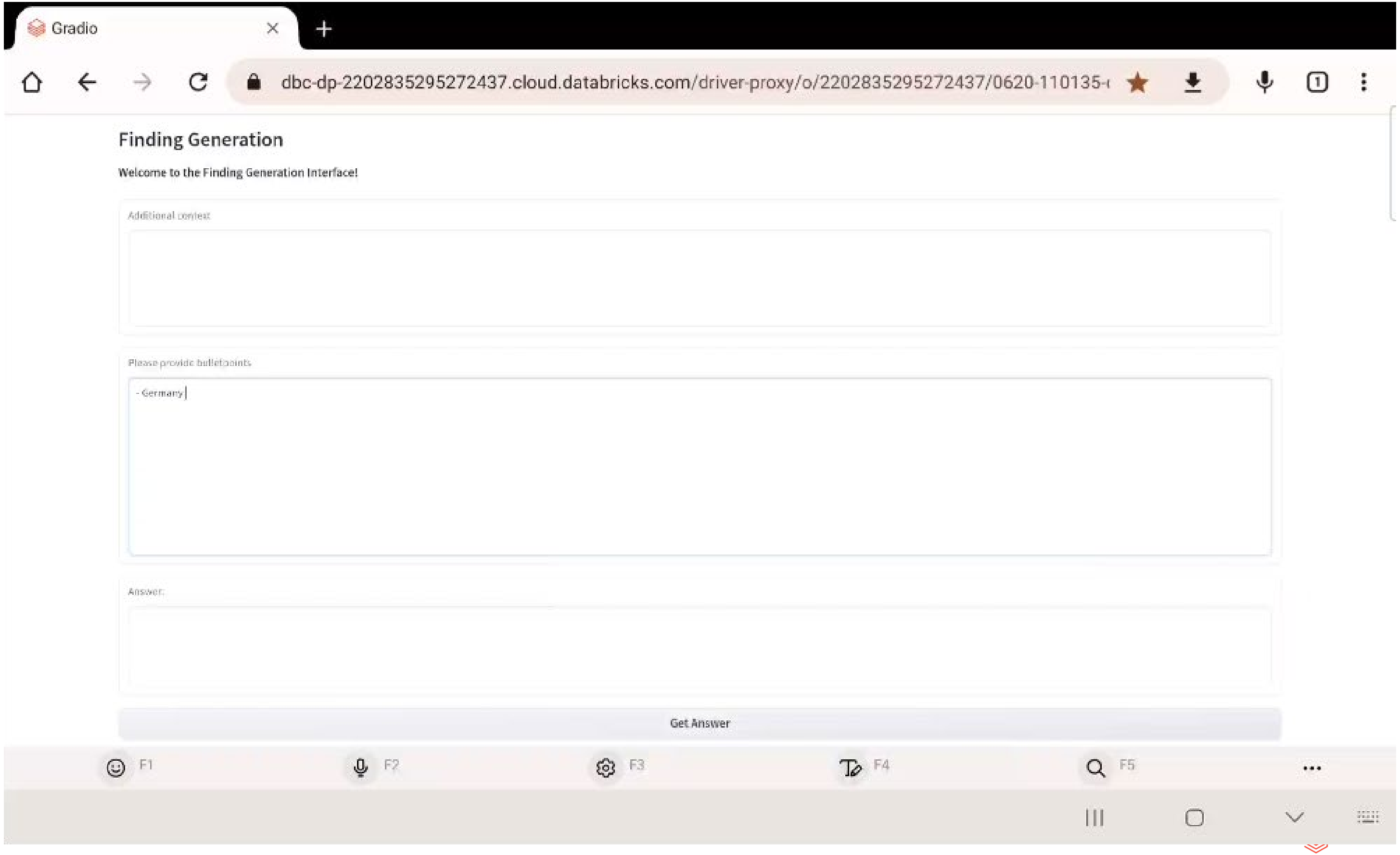
Deployment & Application integration

- Using Databricks Model Serving
- Deployed LLM using Databricks Foundational Model API Provisioned Throughput endpoints (GPU)
- The chain is deployed using classical Databricks Model Serving on CPU endpoint
- We are still using Gradio as a chat interface

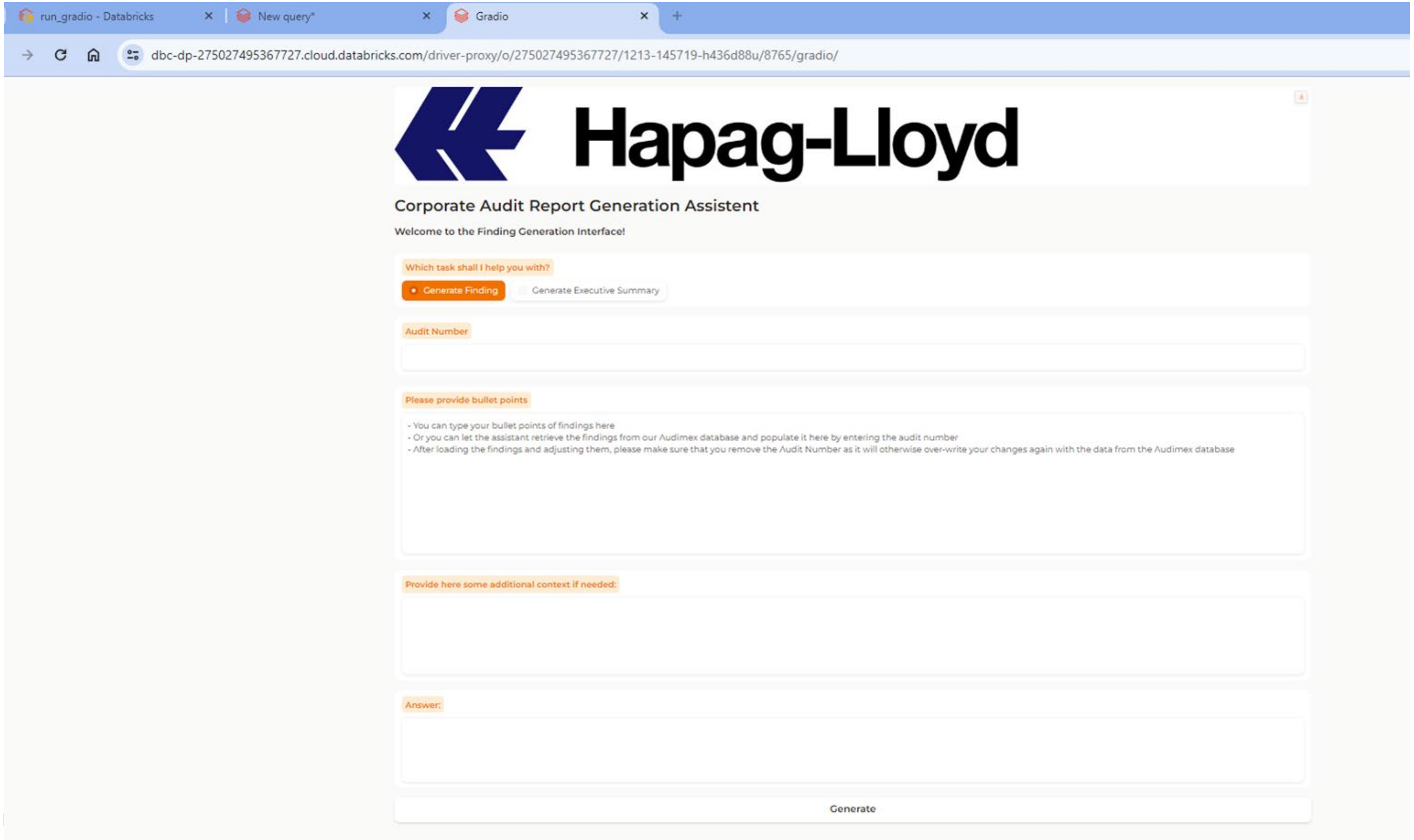
Application architecture: prompt engineering



Here we are – Running our prototype



Here we are – Running our prototype 2.0



Chatbot for process documentation

Project definition

- Auditors spend long time looking for a very specific pieces of information in different files:
  - This can be manuals
  - Different presentations, documents, etc
- They need a simple querying interface supporting natural language queries that allow them to ask for the specific facts defined in the documentation
- It should be possible to add new documents in runtime

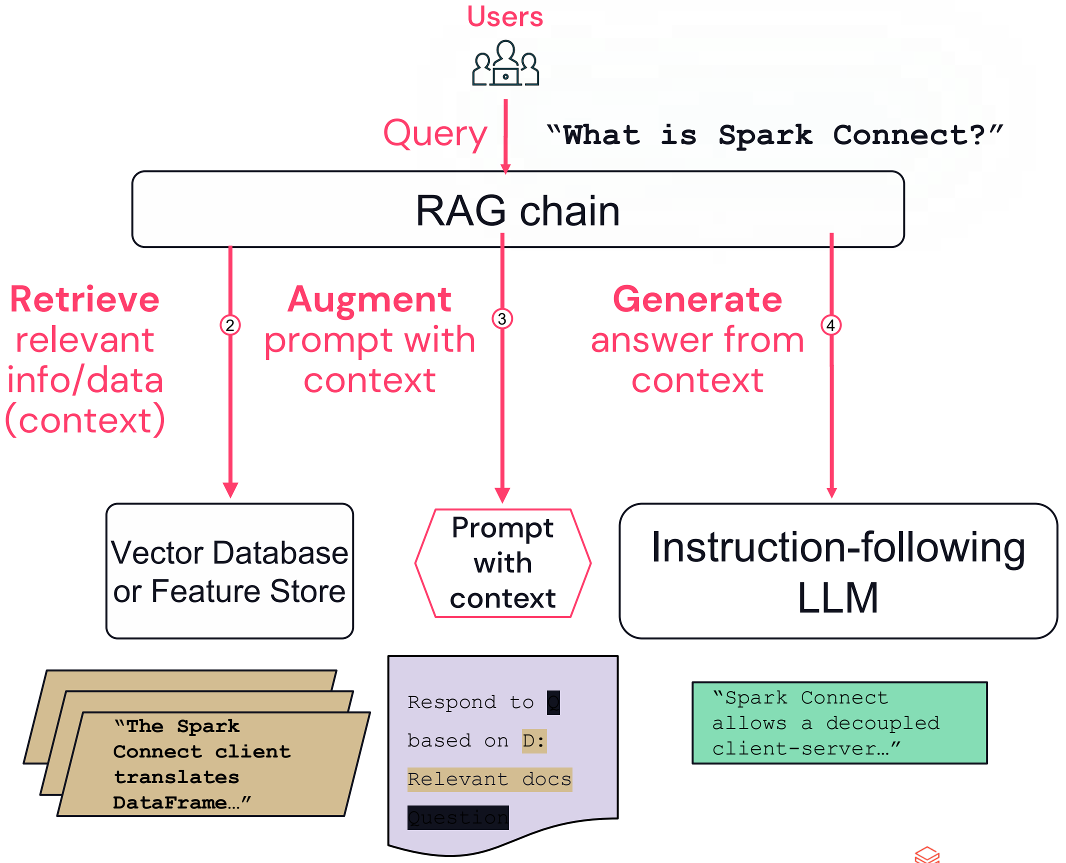
Retrieval Augmented Generation (RAG)

RAG uses LLMs as *reasoning engines*, rather than as static models.

Your data

+

an LLM "brain"



Project stages

Define Problem → Data preparation & Modeling → Deployment & application integration

Define Problem

- Define the business problem
- Create evaluation dataset
- Define metrics

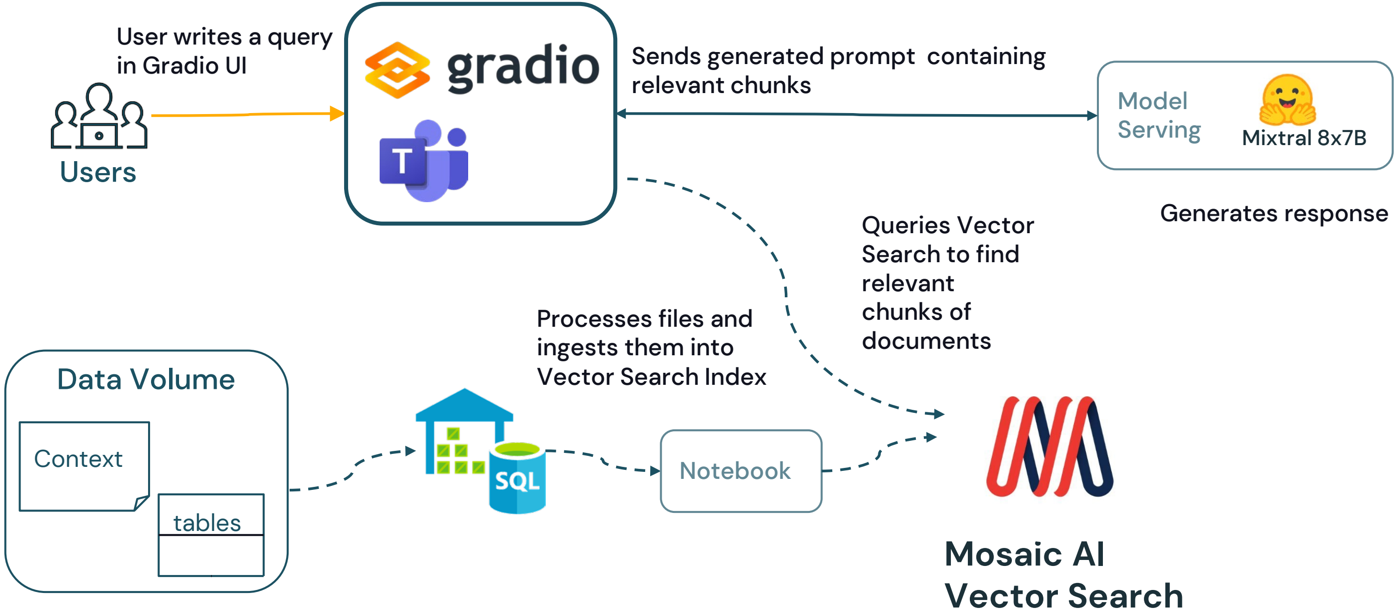
Data preparation & Modeling

- Data preparation
  - Parsing files
  - Chunking data
  - Calculating embeddings
  - Ingesting into Vector DB
- Choose a base model
- Prompt engineering
- Evaluating results
  - Retrieval evaluation
  - Overall evaluation

Deployment & application integration

- Data pipelines deployment
- Model deployment
- Chat interface development

High-level chatbot architecture



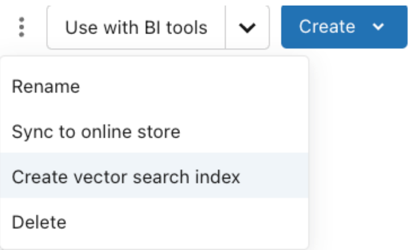
Vector Search

Create auto-updating vector indexes, managed by Unity Catalog

Choose your source table

id	text	col1	col2
1	The quick brown fox jumps ...		
2	How quickly daft...		
3	The five boxing wizards...		

Create semantic search index via Unity Catalog UI or via API



Choose any embedding model

Model Serving

- Foundation Model API
- Custom model
- External model

Call endpoint for real-time retrieval

```
result = index.similarity_search(
  query_text="What is Spark Connect?",
  columns=["id", "text", "link"],
  filters={"doctype": "wiki"})
```

- Ingestion pipelines managed for you
- Indexes managed by Unity Catalog
- Also, APIs for
  - Self-managed embeddings
  - CRUD API upsert/delete

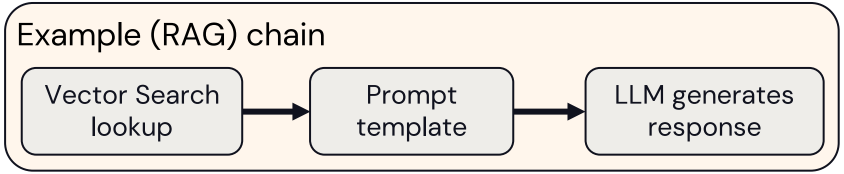
- Integrate with LangChain, LlamaIndex, etc.
- Scale out endpoints as needed

Documentation: AWS, Azure

Chains (and agents)

Building pipelines to include context and complex reasoning

Development



Chains and agents can string together modular LLM features in a structured way, such as for RAG chains.

Common frameworks include:

- LangChain
- LlamaIndex
- Hugging Face

Deployment and Tracking

mlflow

```
mlflow.langchain.log_model(lc_model=llm_chain, …)
```

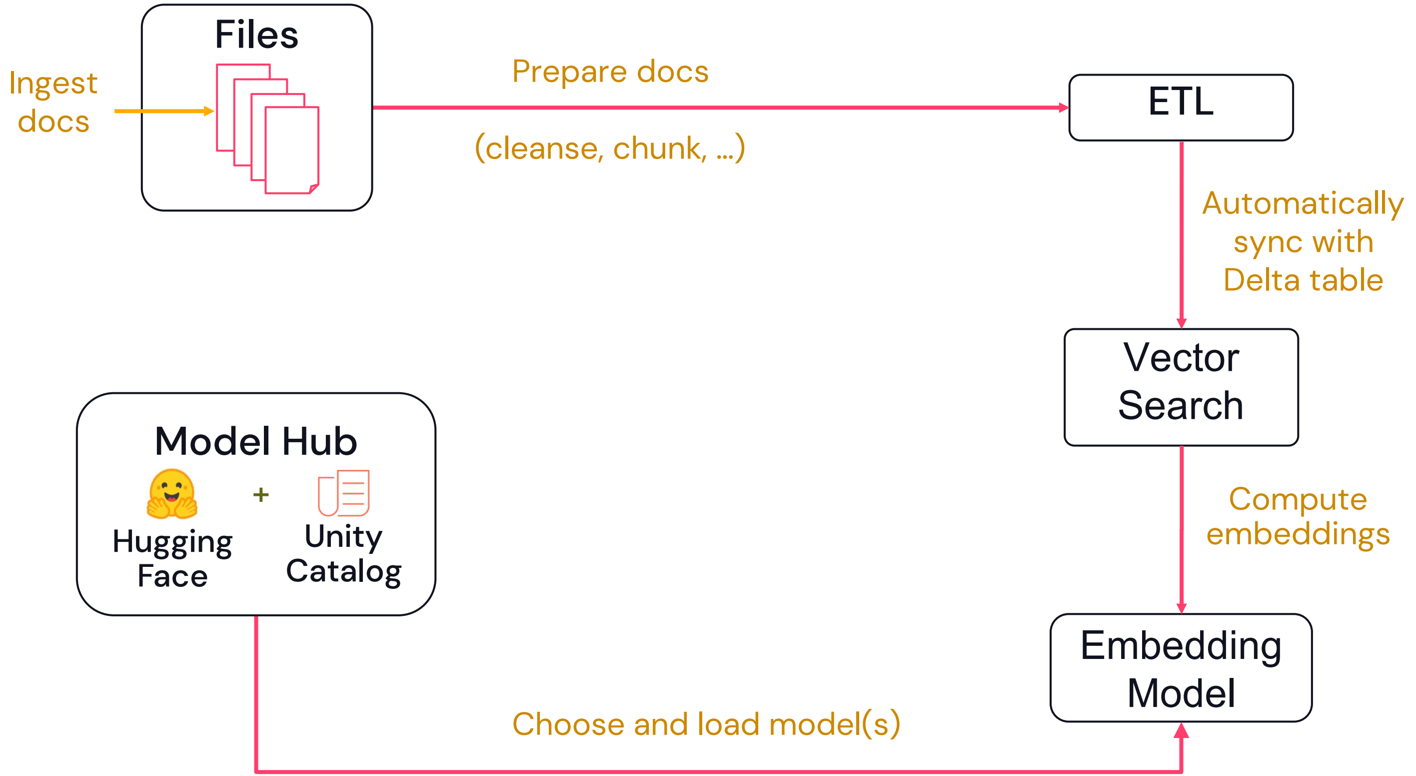
MLflow supports tracking and logging chains, agents, and models. Models can be registered in the Unity Catalog for governance and lineage tracking.

Built-in MLflow flavors include:

- LangChain
- OpenAI
- Transformers
- Sentence Transformers
- PyFunc (for any custom framework)

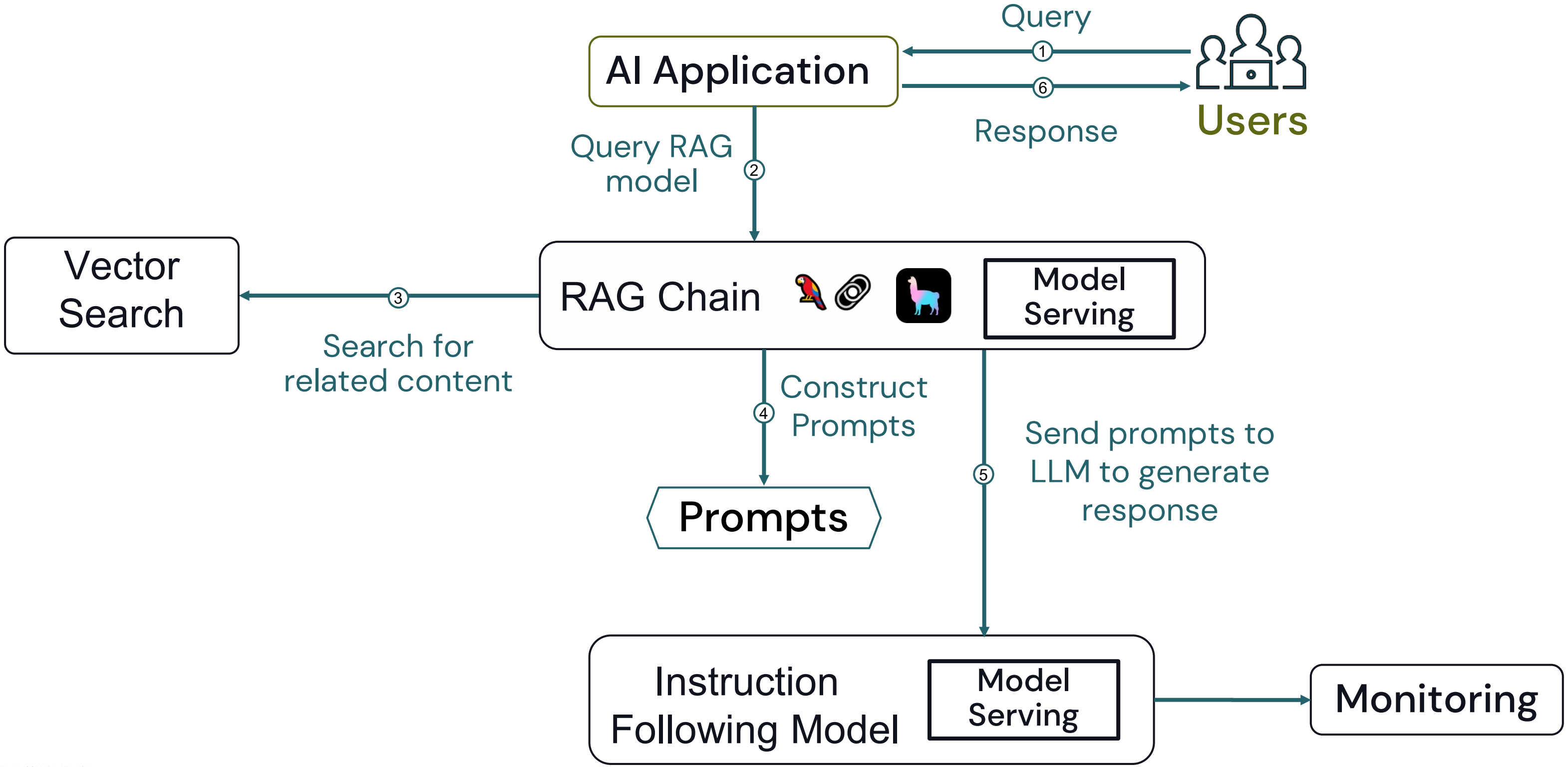
Application architecture: RAG

Preparation

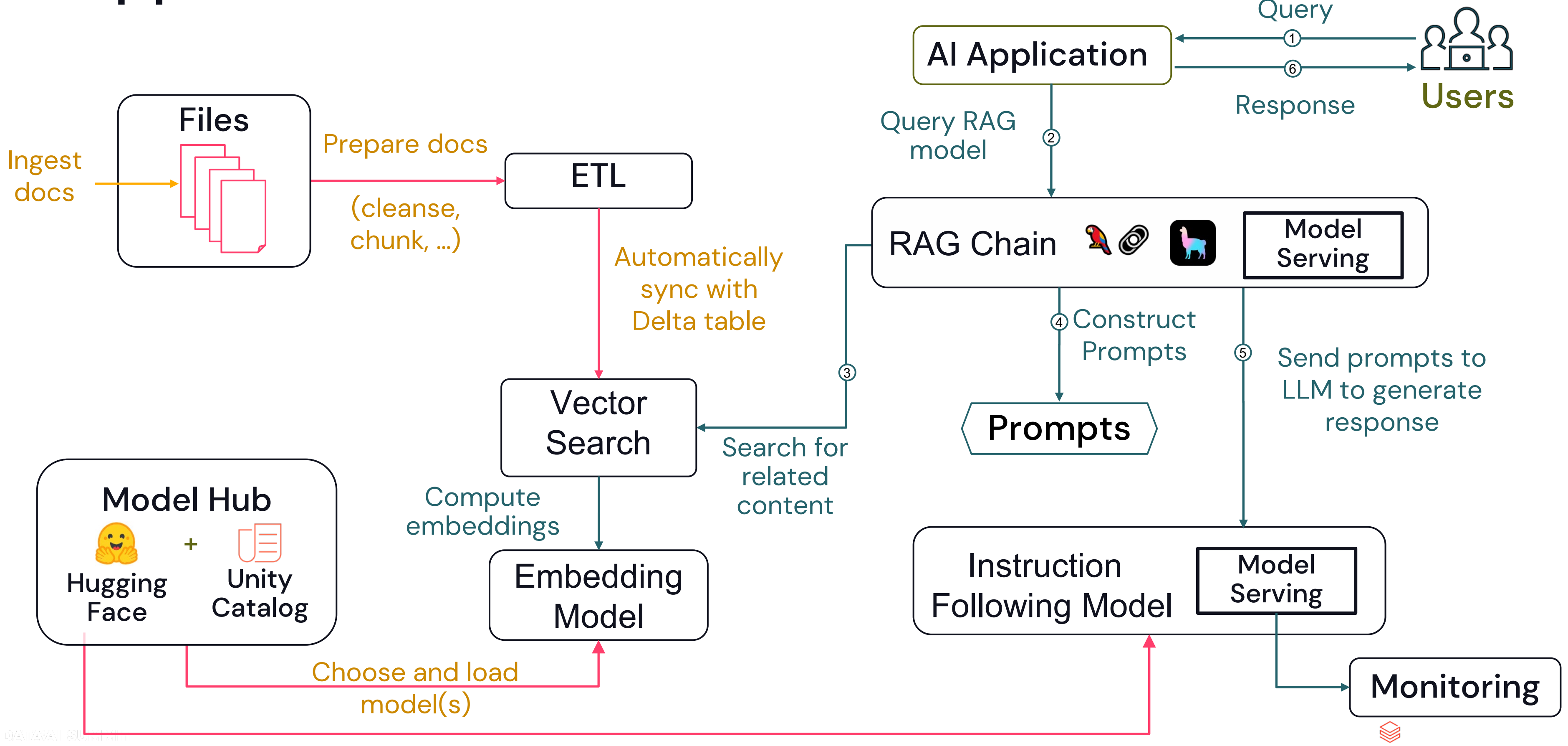


Application architecture: RAG

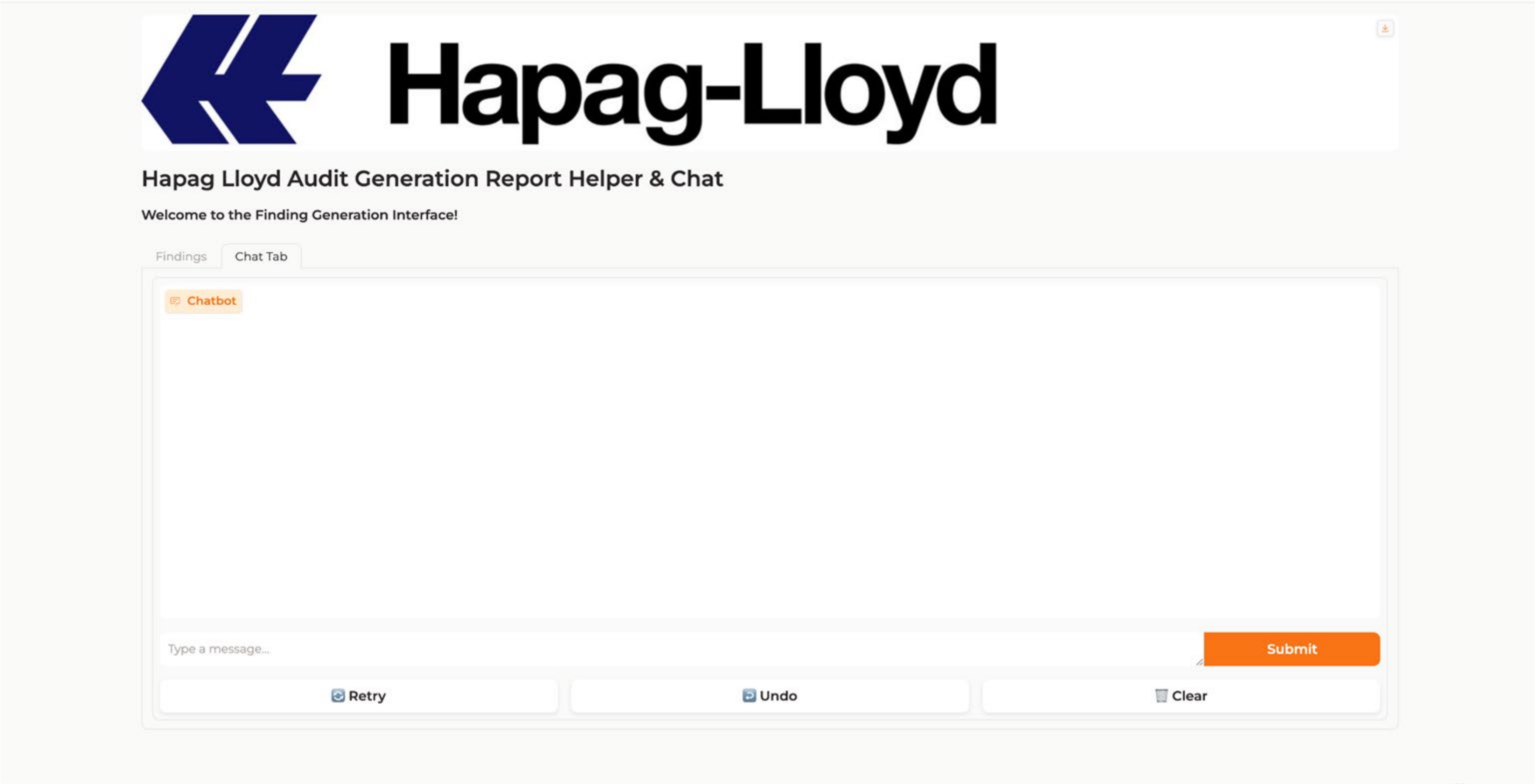
Prompt construction and execution



Application architecture: RAG



Here we are – Running Chat prototype



Whats next

Next Steps

1. The current solution is being tested by auditors of Hapag-Lloyd. There are plans to extend this solution and fine-tune a language model to help the Audit department better structure their reports.

2. Improve and automate evaluation using Mosaic AI Agent Evaluation framework

3. Various departments are increasingly recognizing the value of Generative AI for business. They are exploring proper implementations for multiple use cases, including but not limited to chatbots, summarizing large documents, and providing code assistance.

Thank you!
Questions?

Author Name
Date