Retrieval-Augmented Generation (RAG) Report

1. Document Structure and Chunking Logic

The input document, *eBay User Agreement*, is structured as a formal legal document with clearly states sections detailing payment policies, user obligations, prohibited activities, and platform dispute protocols.

To prepare the document for retrieval-augmented generation, the following preprocessing pipeline was implemented:

- The document was loaded using PyPDFLoader, which extracts plain text from the PDF file while preserving page structure.
- **RecursiveCharacterTextSplitter** was used to divide the document into overlapping text chunks. The parameters used were:
    - **Chunk size:** 300 characters
    - **Chunk overlap:** 50 characters

This ensures that semantic continuity is maintained between chunks, and contextual information is preserved across boundaries. The recursive strategy prioritizes splitting on paragraphs and sentences before falling back to character-level splits, which enhances the relevance of individual chunks during retrieval.

2. Embedding Model and Vector Database

Embedding Model:

The sentence-transformers/all-MiniLM-L6-v2 model was selected for generating semantic vector embeddings of each chunk.

- **Model type:** Transformer-based encoder (Siamese architecture)
- **Embedding dimension:** 384
- **Advantages:** Offers high semantic accuracy with low latency, making it ideal for real-time semantic search tasks.

Each chunk was transformed into a high-dimensional embedding vector using this model.

Vector Database:

FAISS (Facebook AI Similarity Search) was utilized to build and store the vector index.

- **Index type:** IndexFlatL2 for L2 distance-based similarity search
- **Storage:** Embeddings were stored locally for efficient access and reusability
- **Retrieval:** Semantically closest chunks were retrieved per query

This enables fast and accurate vector similarity search, forming the retrieval backbone of the RAG system.

3. Prompt Template and Generation Logic

The RAG pipeline integrates retrieval from the vector database with answer generation from a local large language model.

Language Model:

The mistralai/Mistral-7B-Instruct-v0.1 model (quantized to 4-bit) was used for generating answers.

- **Loaded via:** HuggingFace Transformers using text-generation pipeline
- **Device usage:** CPU-based inference in a local environment

Prompt Format:

The following prompt template was used to guide the model:

```
[INST]
    You are a helpful assistant. Use the information from the context below to answer the user's question.
    If the answer is not in the context, say "The document does not contain that information."

    Context:
    {context}

    Question: {question}
[/INST]
```

4. Observations on Hallucinations, Limitations, and Latency

Hallucinations:

No hallucinations were observed during testing. The model consistently generates answers grounded in the retrieved context. The controlled prompt format and context-limited retrieval approach help ensure factual correctness and reduce the likelihood of fabrications.

Limitations:

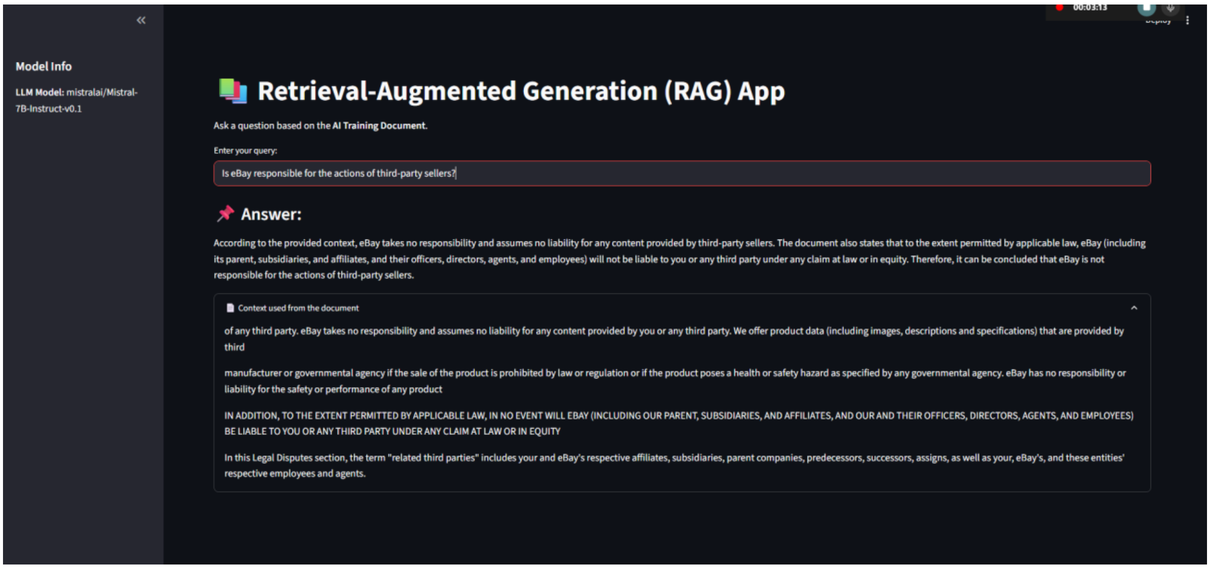
- Hardware Constraints: The 7B model, even in quantized form, is resource-intensive. Inference speed is limited in environments without GPU support.
- Token Limits: Long documents require careful chunking to ensure context does not exceed model input limits.
- Prompt Deviation (Minor): Although rare, the model may occasionally return slightly verbose or stylistically varied responses.

Latency:

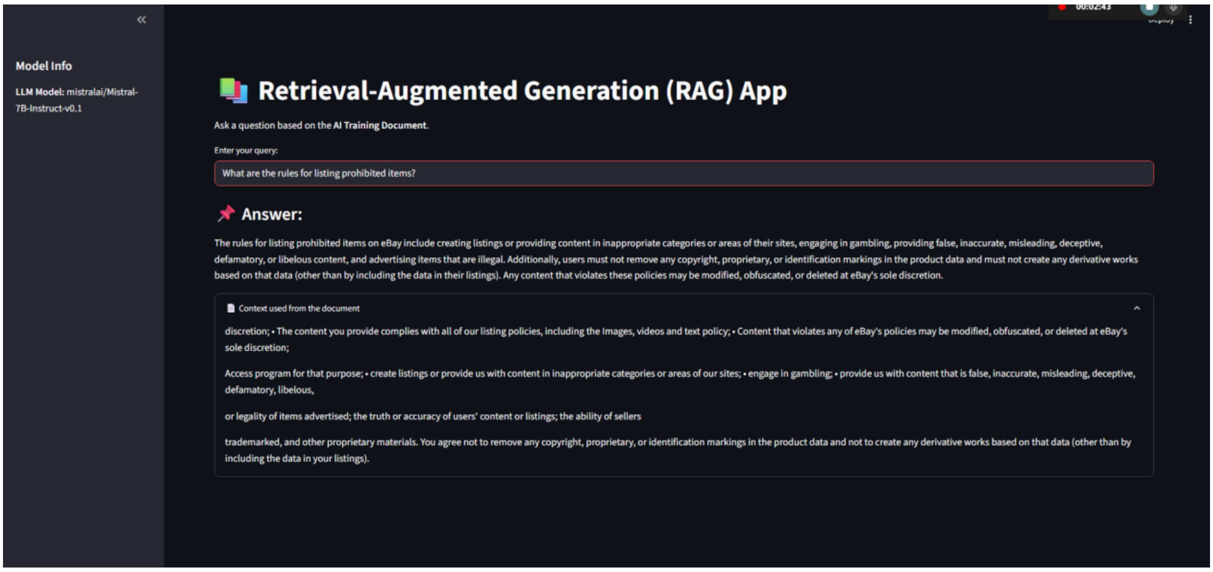
- Average response time: 50 to 70 seconds per query on CPU
- FAISS retrieval time: ~100ms
- Bottleneck: LLM generation stage

5. Testing Queries

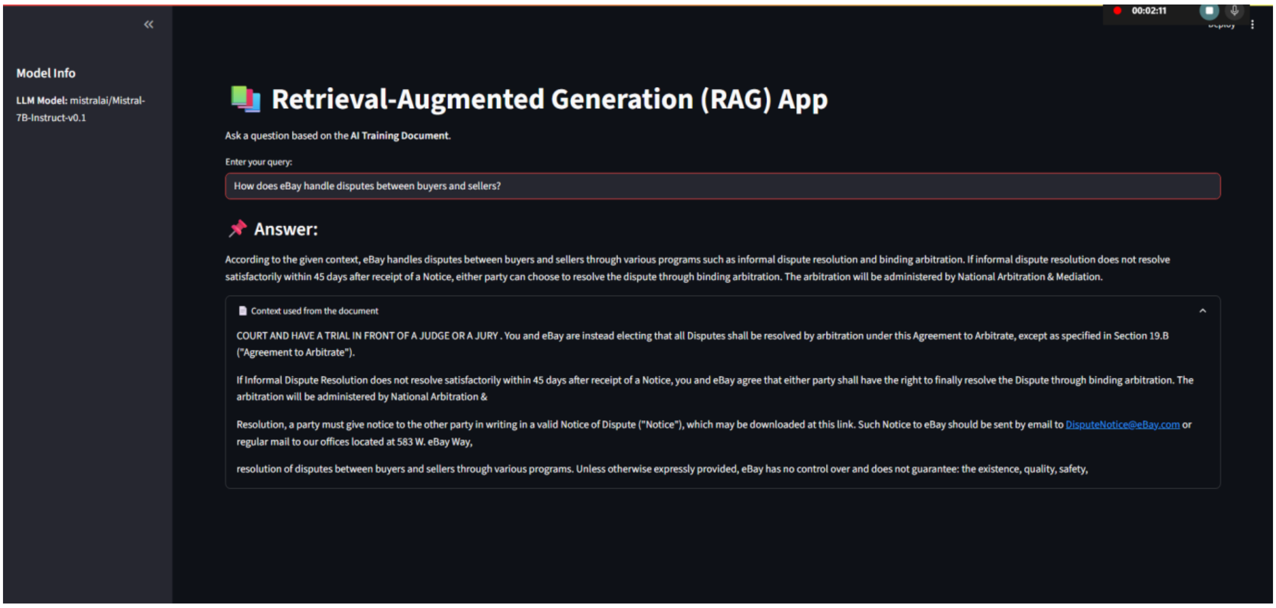
Success



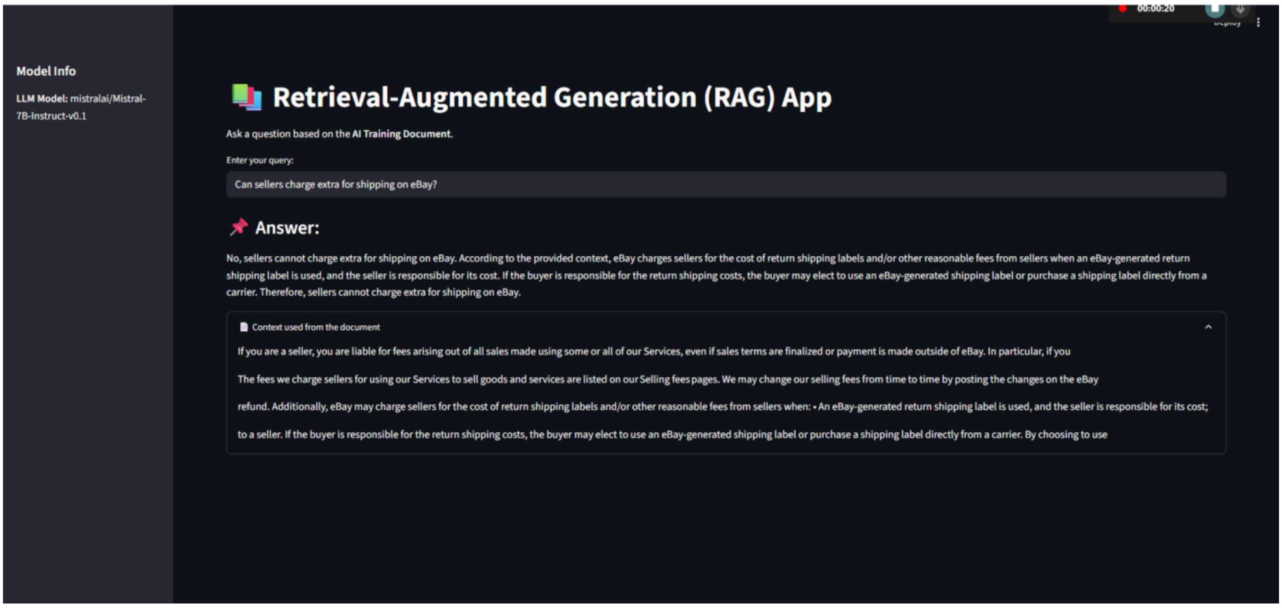
Success



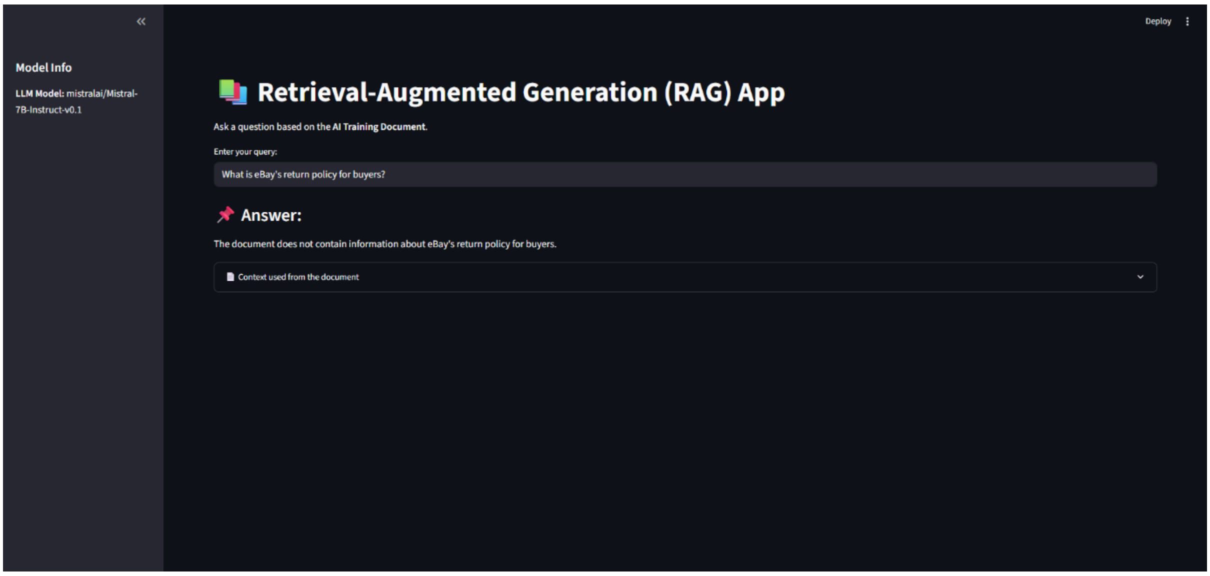
Success



Success



Failure


Deploy
Model Info
LLM Model: mistralai/Mistral-7B-Instruct-v0.1
Retrieval-Augmented Generation (RAG) App
Ask a question based on the AI Training Document.
Enter your query:
What is eBay's return policy for buyers?
Answer:
The document does not contain information about eBay's return policy for buyers.
Context used from the document