Detailed RAG Chatbot Implementation Report

1. Document Structure & Chunking Logic

• Source Document Loading:
  - Uses `PyPDFLoader` to extract text page-by-page, preserving order.
  - Supports multi-file ingestion; can batch process directories.

• Cleaning & Text Normalization:
  - Regex patterns remove headers, footers, page numbers, and boilerplate.
  - Normalize whitespace, unify Unicode characters, strip control codes.
  - Lowercase or maintain casing based on downstream embedding sensitivity.

• Chunking Strategy:
  - `RecursiveCharacterTextSplitter` splits on sentence boundaries and newlines.
  - Configured with 300-token chunk size and 50-token overlap for seamless context.
  - Overlap ensures terms near boundaries are not lost between chunks.
  - Separators prioritized: paragraph breaks, punctuation, then whitespace.

• Storage and Metadata:
  - Each chunk stored with metadata: source file, page number, chunk index.
  - Metadata allows traceability back to original document for citation.

2. Embedding Model & Vector Database

• Embedding Model:
  - OllamaEmbeddings (`nomic-embed-text`): delivers 384-dimensional vectors.
  - Pros: on-premises inference, no external API latency or costs.
  - Alternatives:
    * `all-MiniLM-v2`: faster, smaller footprint, slightly lower accuracy.
    * `bge-small-en`: higher semantic fidelity, requires more compute.

• Vector Store (FAISS):
  - Flat index (Exact k-NN): constant-time lookup for up to ~100k vectors.
  - Serialization: `index.to_bytes()` for disk persistence and reload.
  - Scaling Options:
    * HNSW index for approximate but sub-second search at millions of vectors.
    * Cloud solutions (Pinecone, Weaviate) for multi-region availability.

• Embedding Workflow:
  1. Generate embeddings for each chunk via `embed_model.embed_documents()`.
  2. Store vectors and metadata in FAISS index.
  3. On query, compute `embed_model.embed_query()` and perform k-NN search.

3. Prompt Format & Generation Logic

• Prompt Template:
  - Structure:
    "Answer based only on the context below. Context: {context} Question: {input}"
  - Context concatenation includes top-k retrieved chunks separated by markers.
  - Token Budgeting: ensure combined context + question stays within model window.

• LLM & Streaming:
  - Model: Llama3-8b-8192 via `ChatGroq` API with `streaming=True`.
  - Benefits: token-by-token delivery yields responsive UX in Streamlit.
  - Configuration: temperature=0.1 for factual answers, max_tokens=512 limit.

• RAG Chain Construction:
- `RetrievalQA.from_chain_type(chain_type='stuff')` stitches all chunks into one prompt.
- Returns both the generated answer and source documents for citation display.

• Error & Truncation Handling:
- If context is too large, pre-truncate to nearest sentence boundary.
- Catch exceptions in retrieval or generation, with retry or fallback responses.

4. Notes on Hallucinations & System Limitations

• Hallucination Risks:
- Occurs when queries exceed indexed content or context is sparse.
- Mitigation: lower temperature, increase overlap, return source citations.

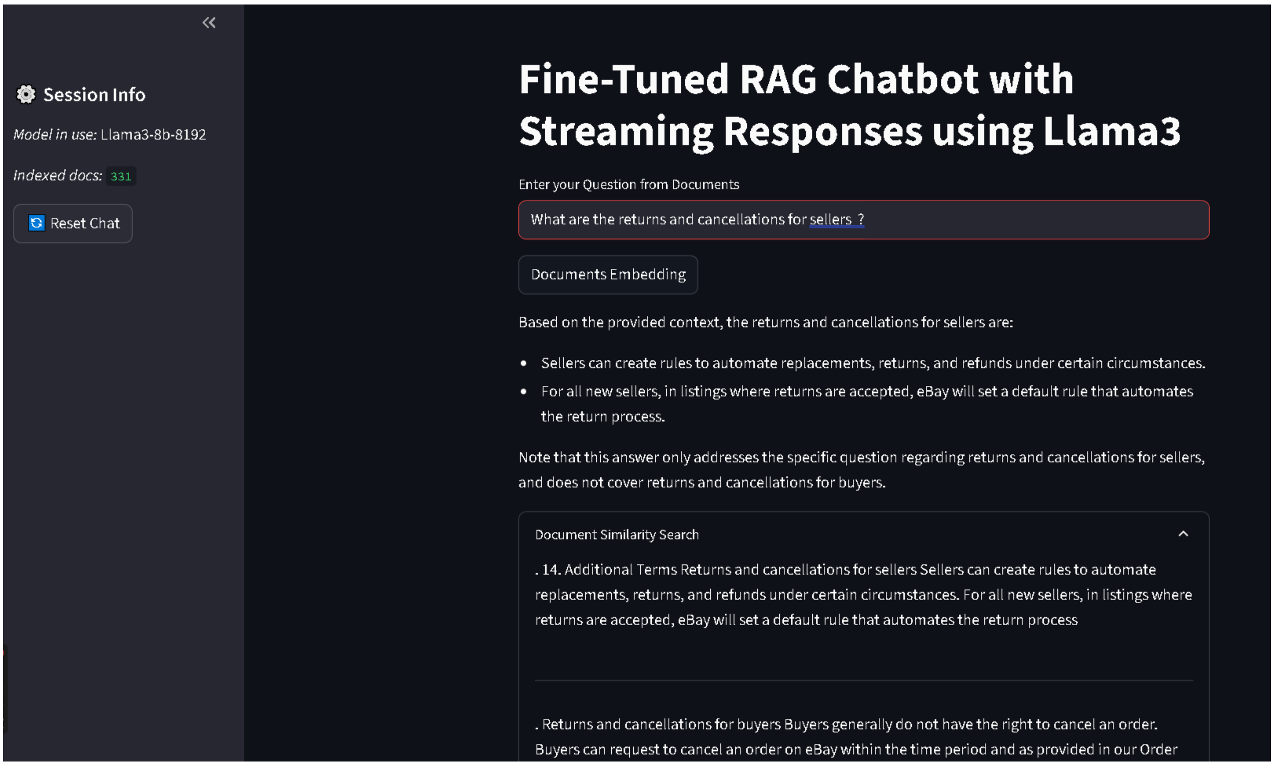
• Context Window Constraints:
- Llama3-8b supports up to 8192 tokens; larger docs require chunk filtering.
- Plan: use dynamic chunk scoring (TF-IDF) to select highest relevance.

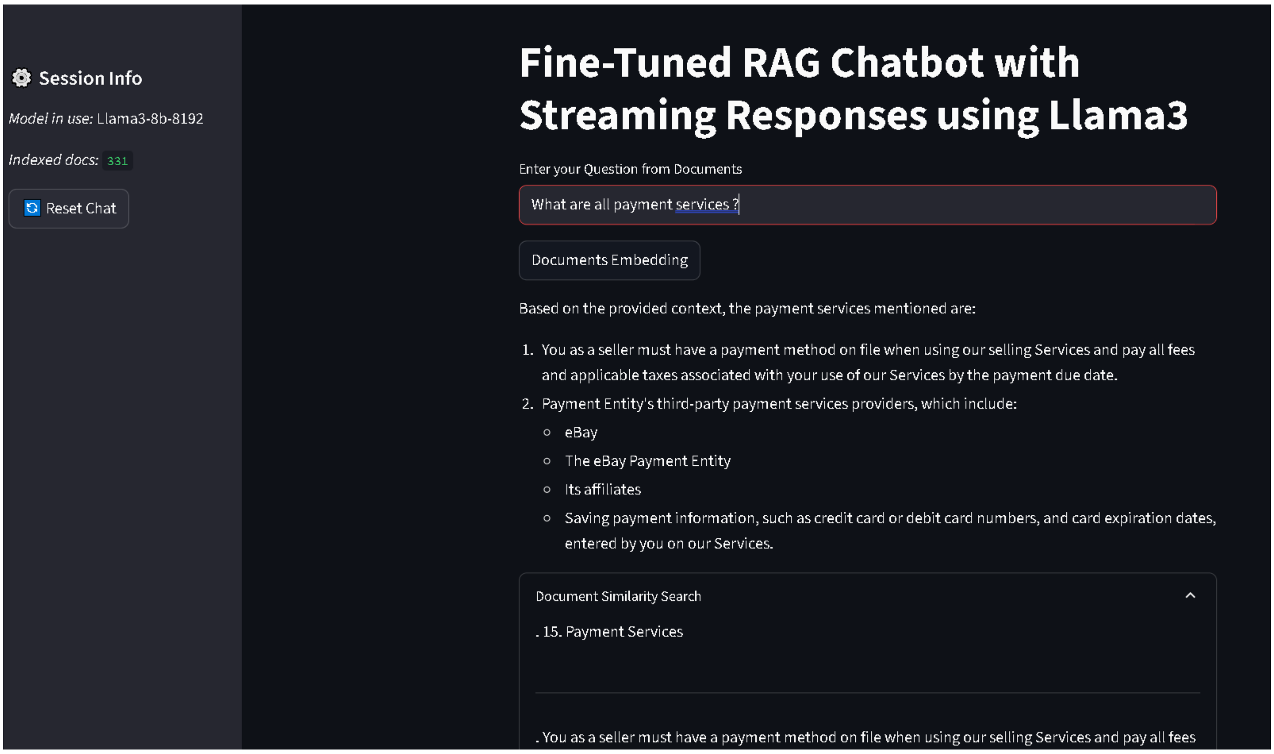
• Performance Considerations:
- FAISS flat index search <10ms for 100k vectors; embedding/query ~50ms.
- For larger data, use approximate indexes or batched embedding pipelines.

• Latency & UX:
- Streaming reduces perceived latency; network instability can interrupt streams.
- Implement heartbeat pings or fallback buffering for smoother UI.

• Future Enhancements:
- Hybrid search: combine dense embeddings with keyword filters for precision.
- Multi-language support: integrate language detection & language-specific models.

5. example queries with responses




Session Info
Model in use: Llama3-8b-8192
Indexed docs: 331
Reset Chat
Fine-Tuned RAG Chatbot with Streaming Responses using Llama3
Enter your Question from Documents
What are all payment services ?
Documents Embedding
Based on the provided context, the payment services mentioned are:
1. You as a seller must have a payment method on file when using our selling Services and pay all fees and applicable taxes associated with your use of our Services by the payment due date.
2. Payment Entity's third-party payment services providers, which include:
eBay
The eBay Payment Entity
Its affiliates
Saving payment information, such as credit card or debit card numbers, and card expiration dates, entered by you on our Services.
Document Similarity Search
. 15. Payment Services
. You as a seller must have a payment method on file when using our selling Services and pay all fees