Retrieval-Augmented Generation with Qdrant and LLM Integration

Overview

This application implements a **Retrieval-Augmented Generation (RAG)** system, combining vector similarity search using **Qdrant** with reasoning-based text generation via **Groq's LLaMA model**. The system enables intelligent question answering over uploaded documents using advanced prompting strategies, providing interpretable outputs grounded in retrieved context.

System Architecture

Core Components

- **Vector Store**: Qdrant is used to store and search semantic vector representations of text chunks.

- **Embedding Model**: The all-MiniLM-L6-v2 model from sentence-transformers is used to generate 384-dimensional sentence embeddings.

- **LLM Backend**: Groq's hosted version of the meta-llama/llama-4-scout-17b-16e-instruct model is used for answer generation.

- **Frontend Interface**: Built with Streamlit, allowing interactive document uploads, query input, technique selection, and output display.

Functional Flow

A. Document Processing

1. **Supported Formats**: .txt, .pdf, .docx
2. **Conversion Logic**:
    - PDFs processed using pdfplumber
    - DOCX processed using python-docx
    - Files are saved as .txt in a local directory (texts/)
3. **Chunking Strategy**:
    - Token-based chunking with a fixed CHUNK_SIZE of 100 tokens and an OVERLAP of 30 to preserve context between segments.
4. **Embedding and Storage**:
    - Each chunk is embedded using MiniLM and upserted into a Qdrant collection with associated metadata (chunk text and source filename).
    - Collection is recreated upon DB rebuild to ensure consistency.

B. Querying and Reasoning

1. **User Input**:
    - Users input natural language questions.
    - Users select one or more prompting techniques to guide the LLM's reasoning.
2. **Retrieval**:
    - Top-k (default 5) relevant chunks are retrieved via cosine similarity from Qdrant.
3. **Prompt Engineering**:
    - Context is assembled from retrieved chunks.
    - Prompting techniques are translated into specific instructions appended to the query.
    - If “MCQ prompting” is selected, an additional structured format for multiple-choice questions is enforced.
4. **LLM Interaction**:
    - A structured system/user prompt is sent to Groq’s LLaMA model through their API.
    - The model responds with either a direct answer or formatted MCQs depending on the prompt.

Prompting Techniques Supported

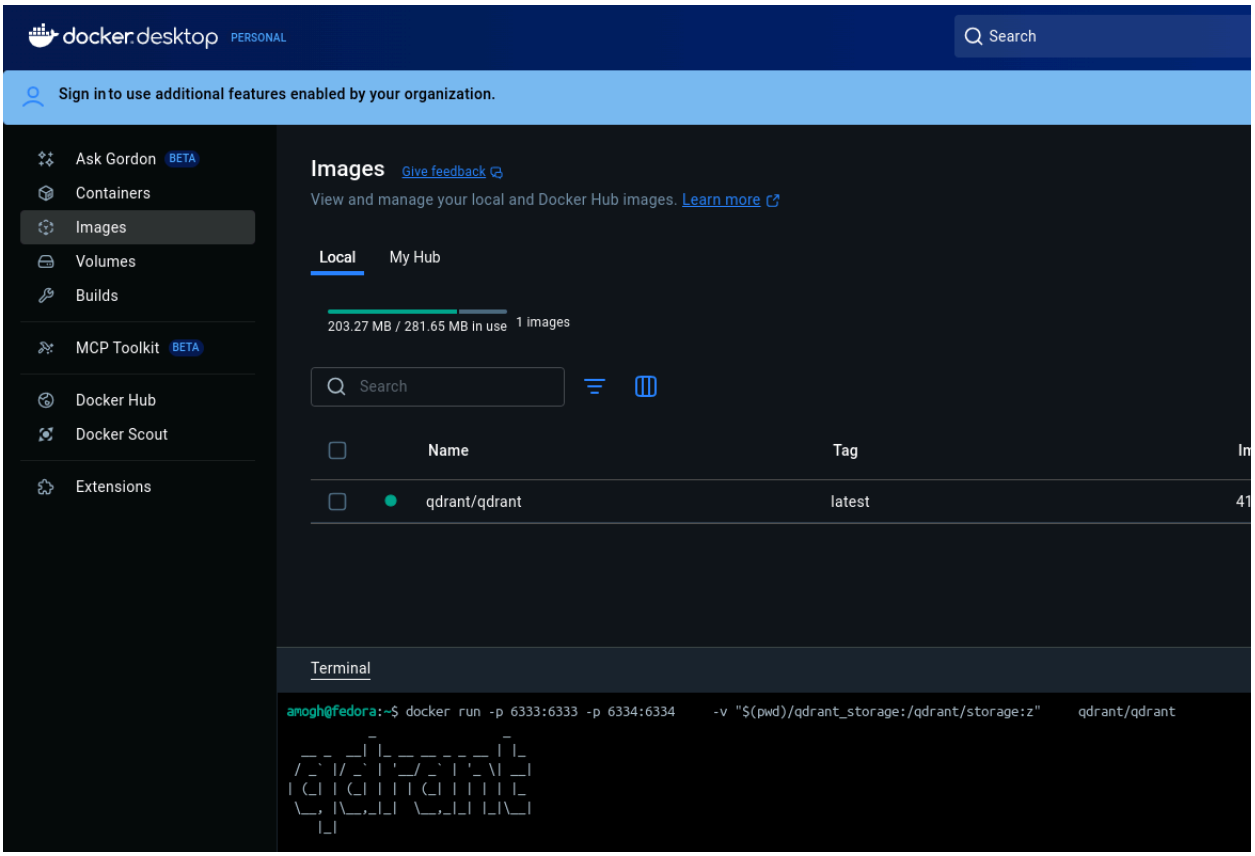
Technique	Description
Chain-of-Thoughts	Step-by-step reasoning.
Tree-of-Thoughts	Branching exploration paths before conclusion.
Role-based prompting	Assumes a relevant persona (e.g., expert, teacher).
ReAct prompting	Combines reasoning and action (e.g., verify, deduce).
Directional Stimulus	Injects guiding hints into context.
Step-Back prompting	Promotes re-evaluation of assumptions.
MCQ prompting	Converts context into MCQs with answer keys.

How to Use it

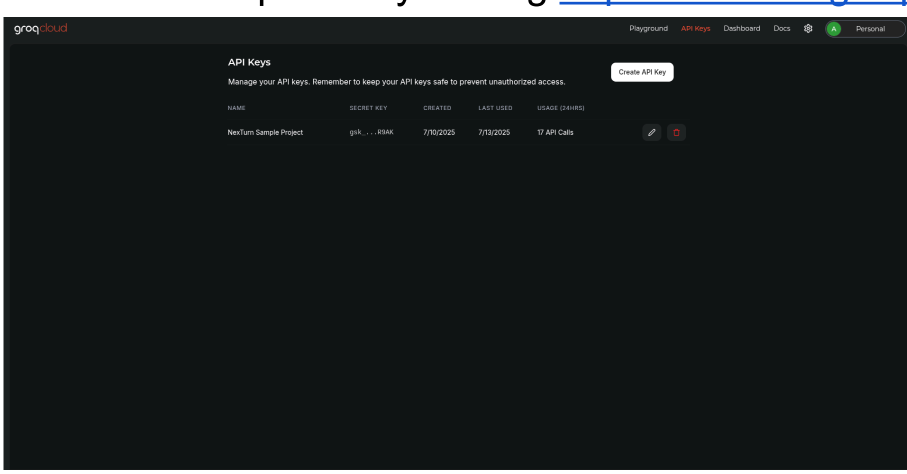
1. Pull docker image for vector database by using

   docker pull qdrant/qdrant

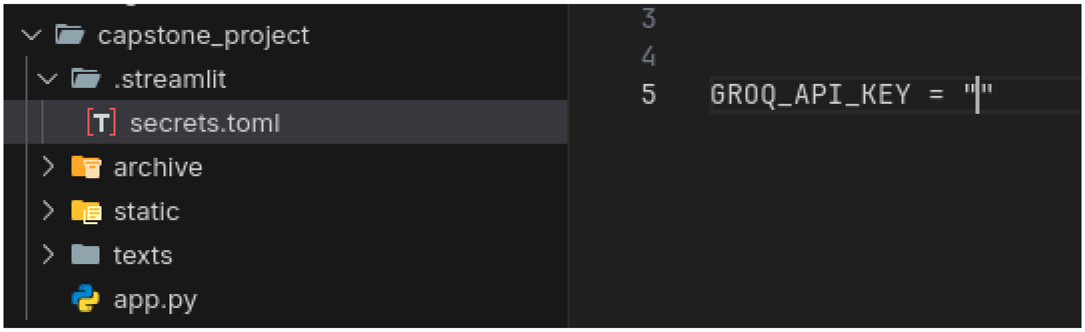
2. Run the image



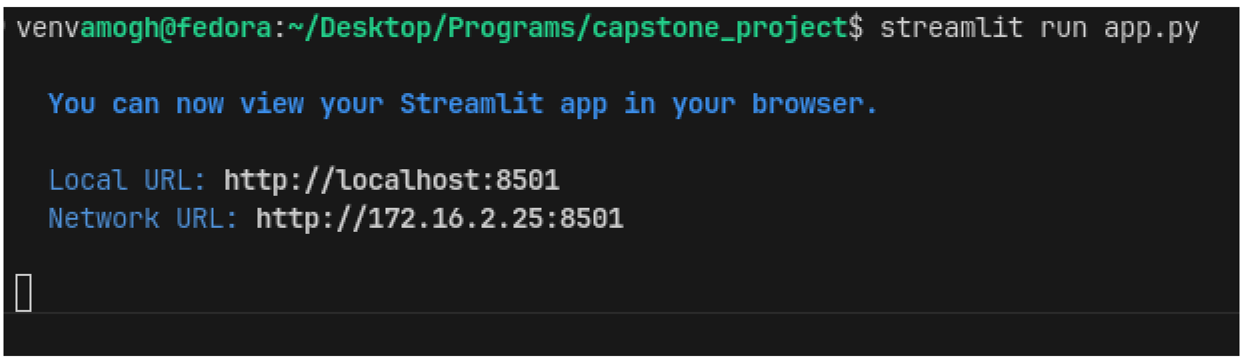
3. Generate Groq API key visiting https://console.groq.com/keys



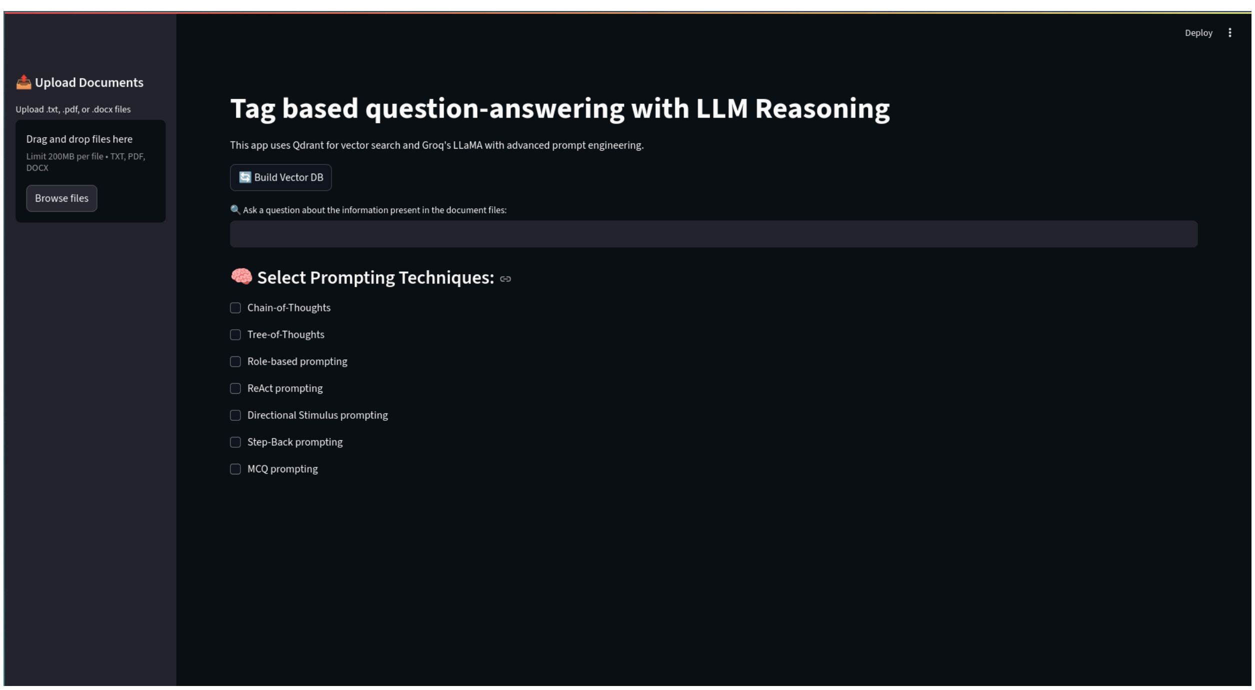
4. Paste it into .streamlit/secrets.toml



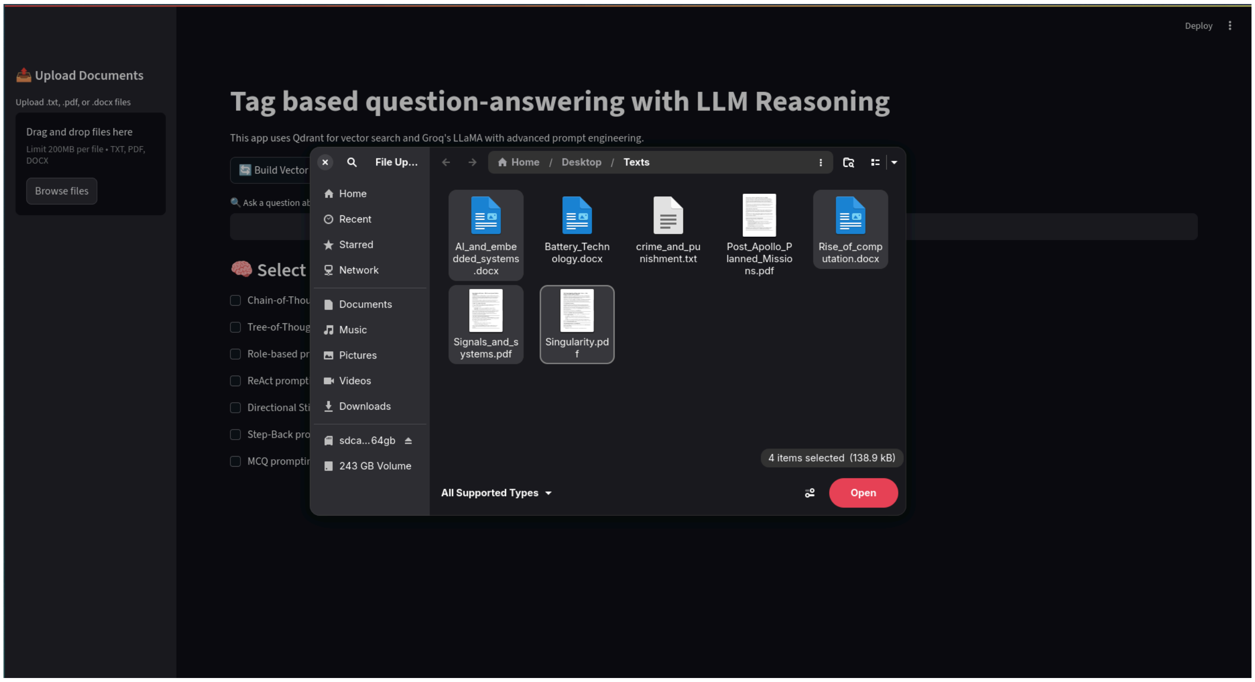
5. Run pip install -r requirements.txt to install necessary packages.
6. Run the app using streamlit run app.py



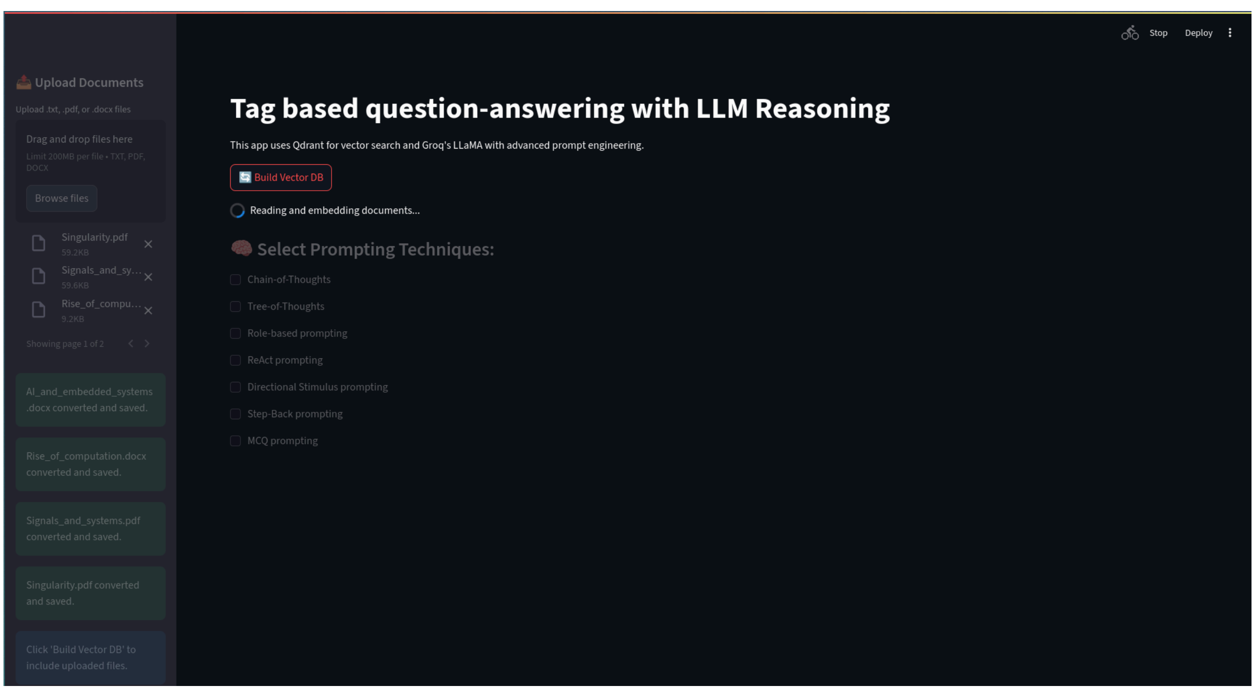
7. Open the browser where the app is running



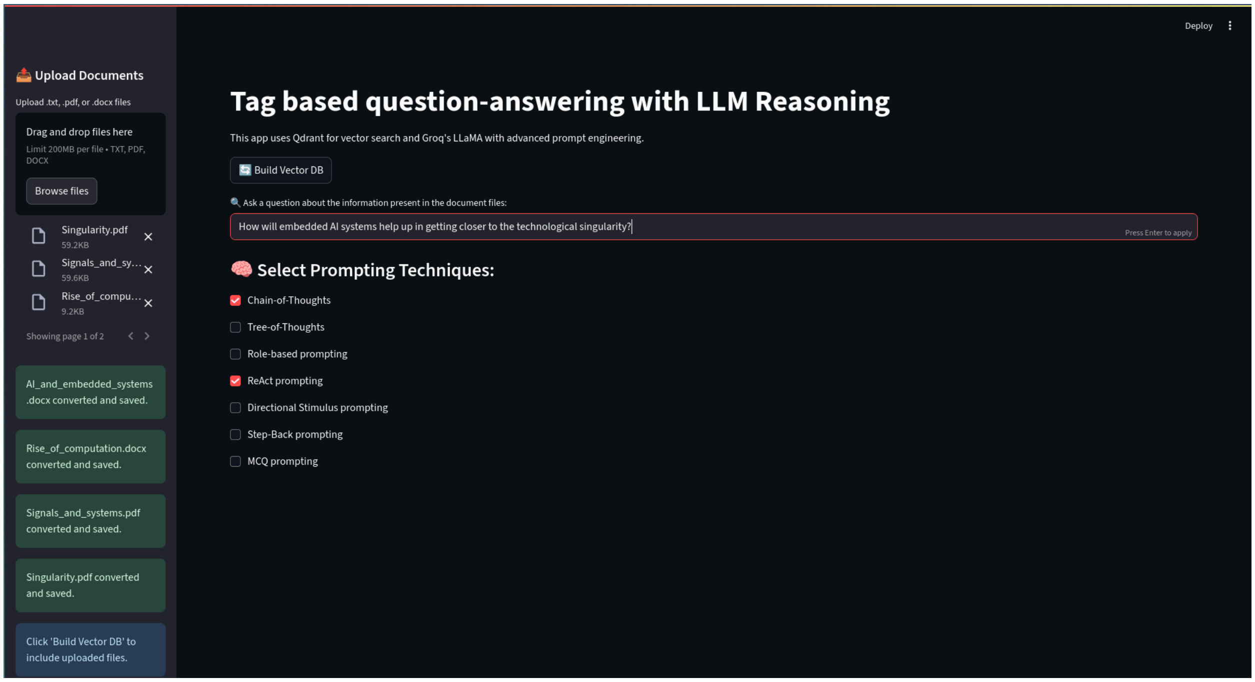
8. Select the files you want to query, the app can take .txt, .pdf and .docx files as input.



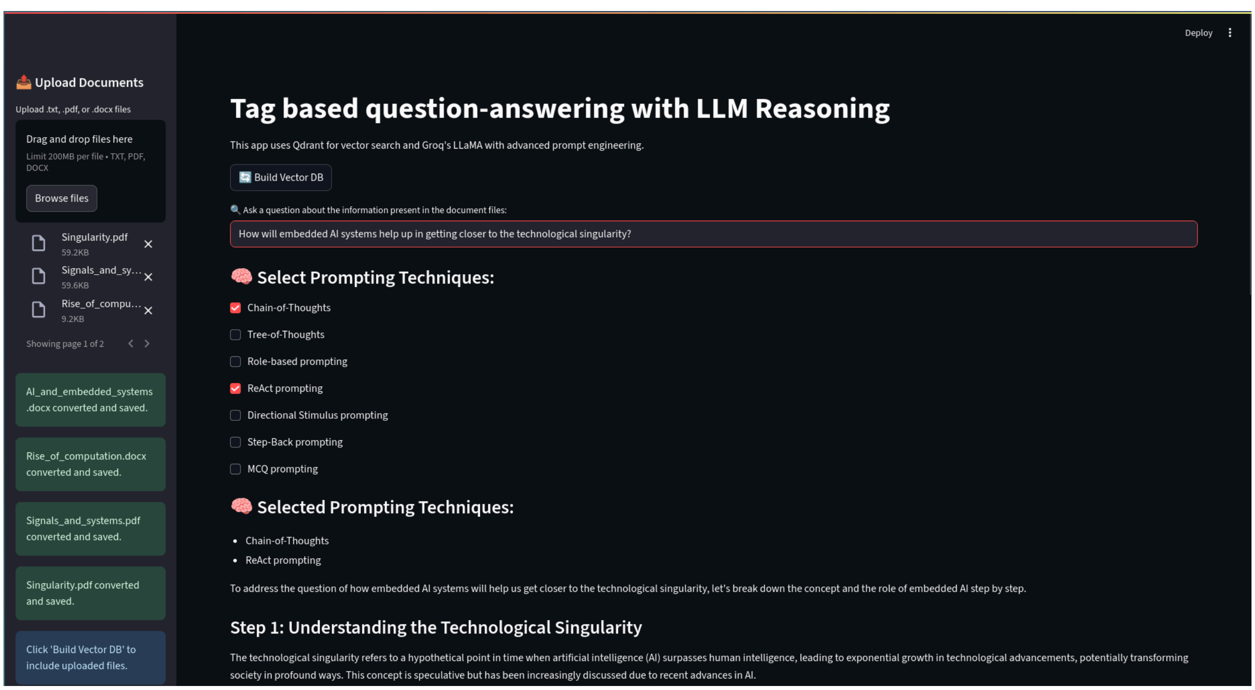
9. Once the files are uploaded, click Build Vector DB to build the vector database

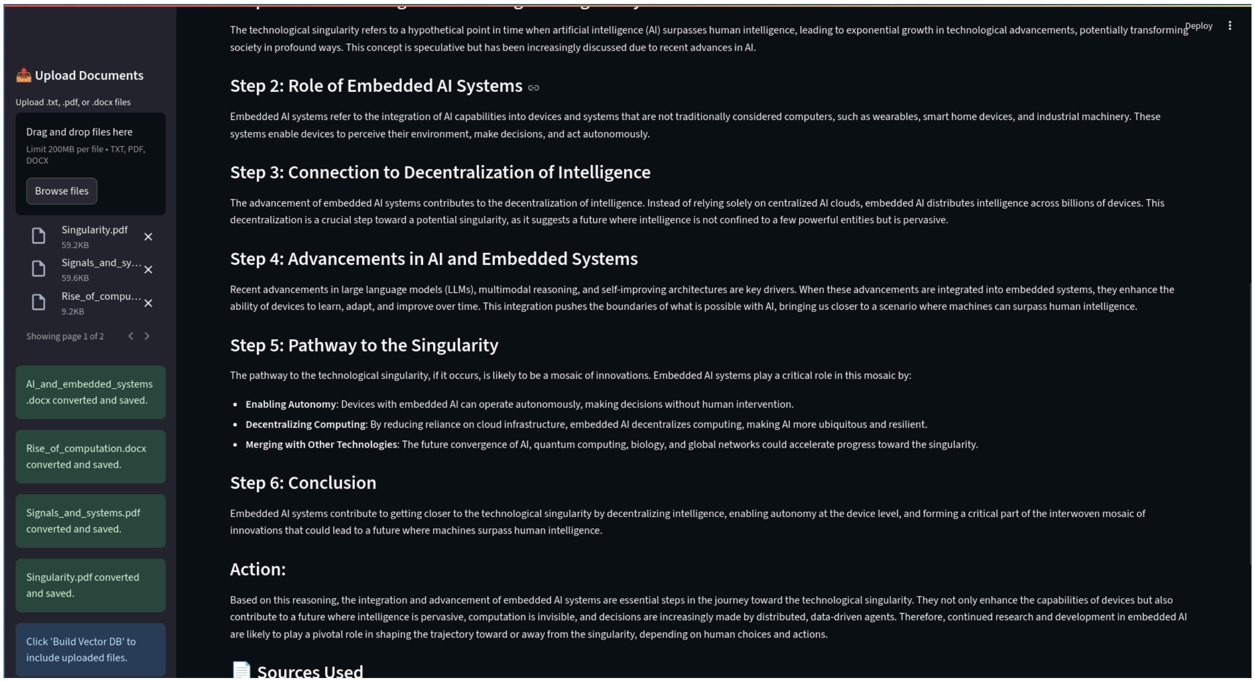


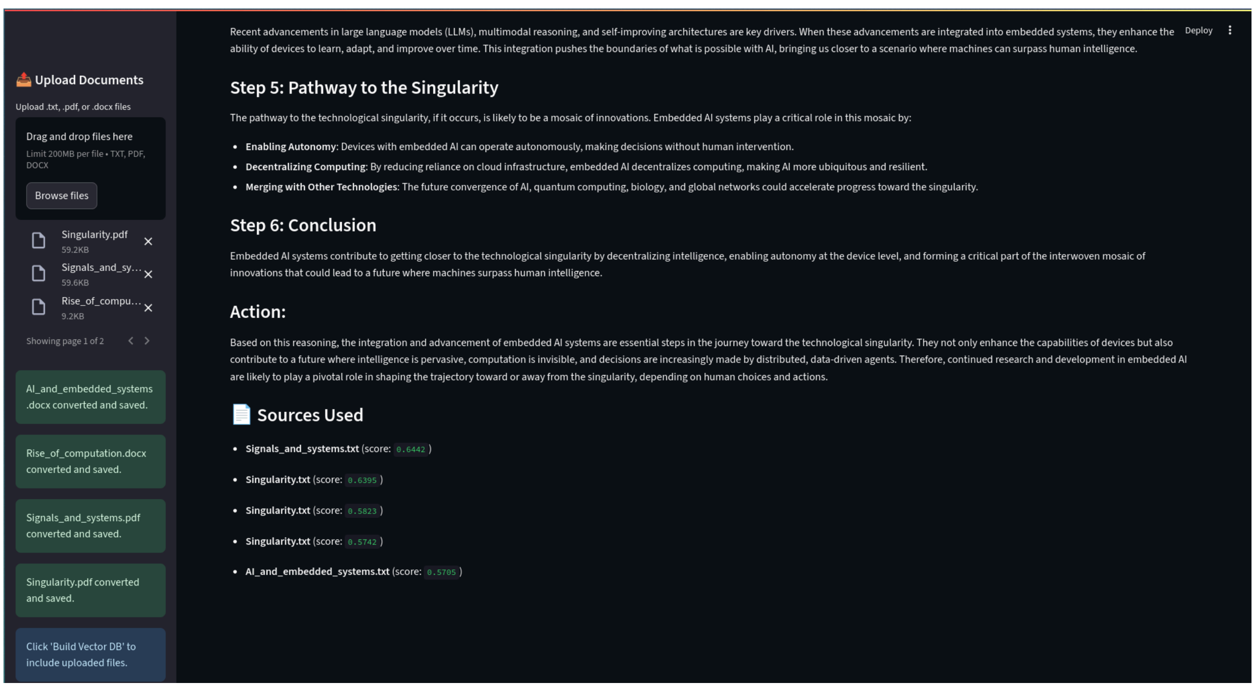
10. Once the vector database is built, select the tags from the list to select the kind of meta-prompt that is to be used to query the texts added and then the query.



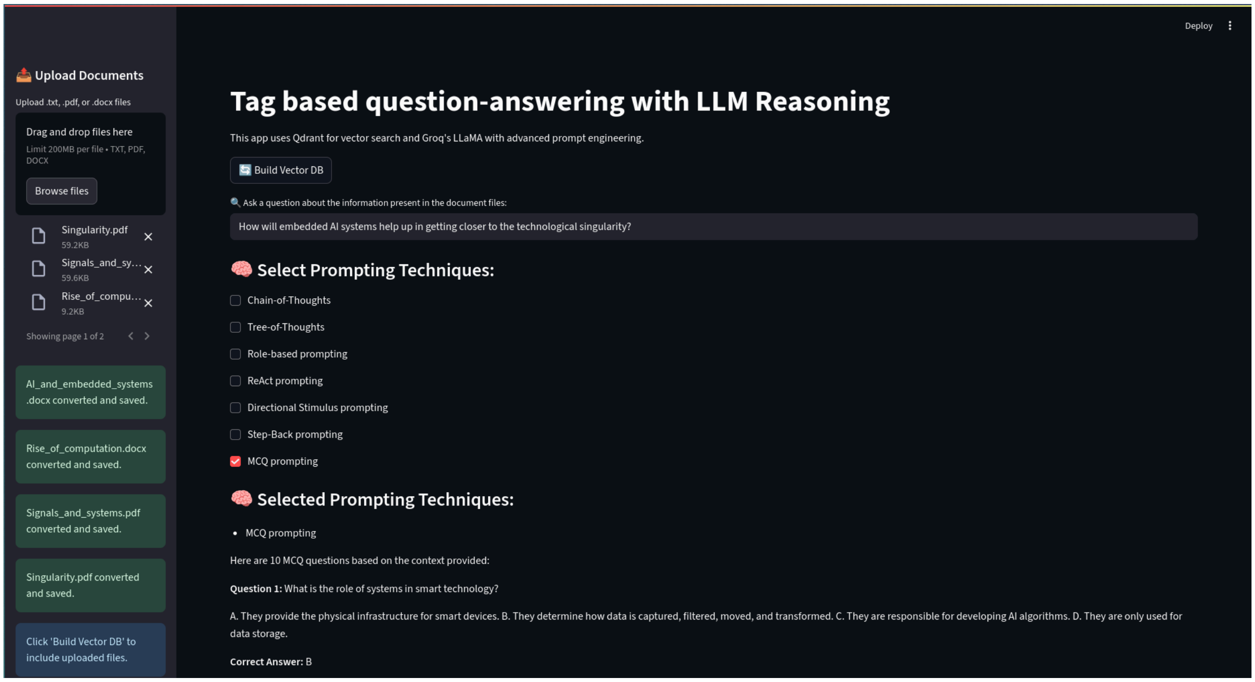
11. Output is generated.

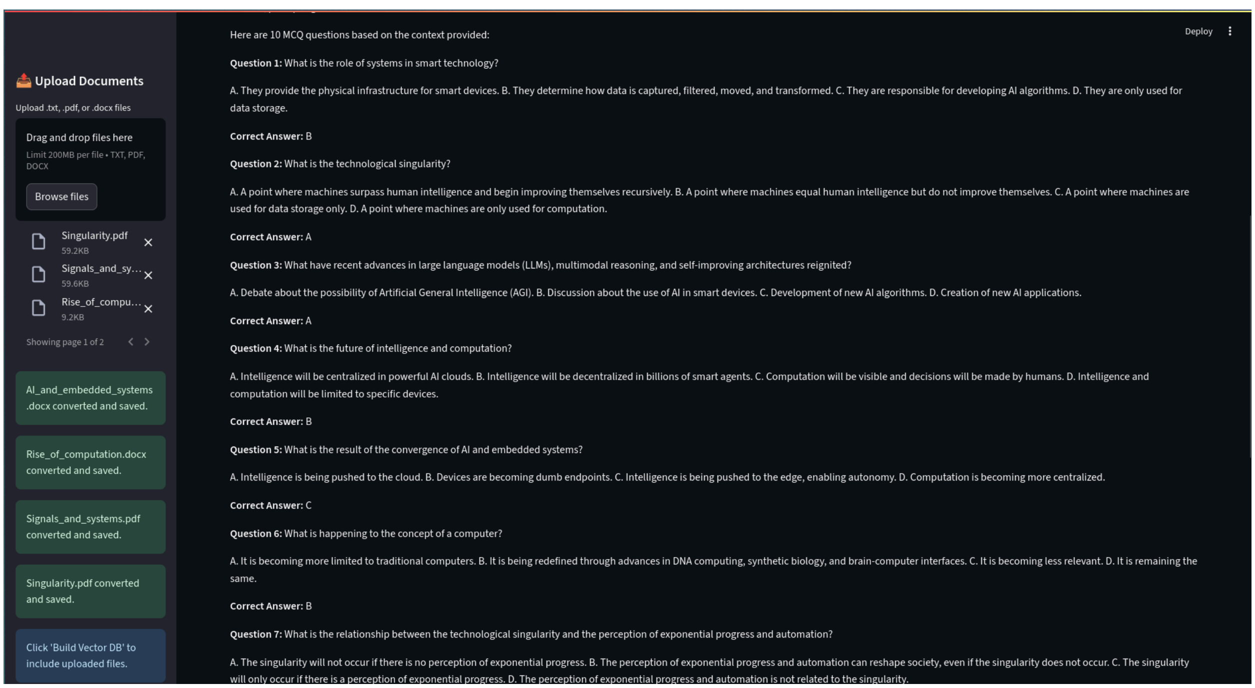


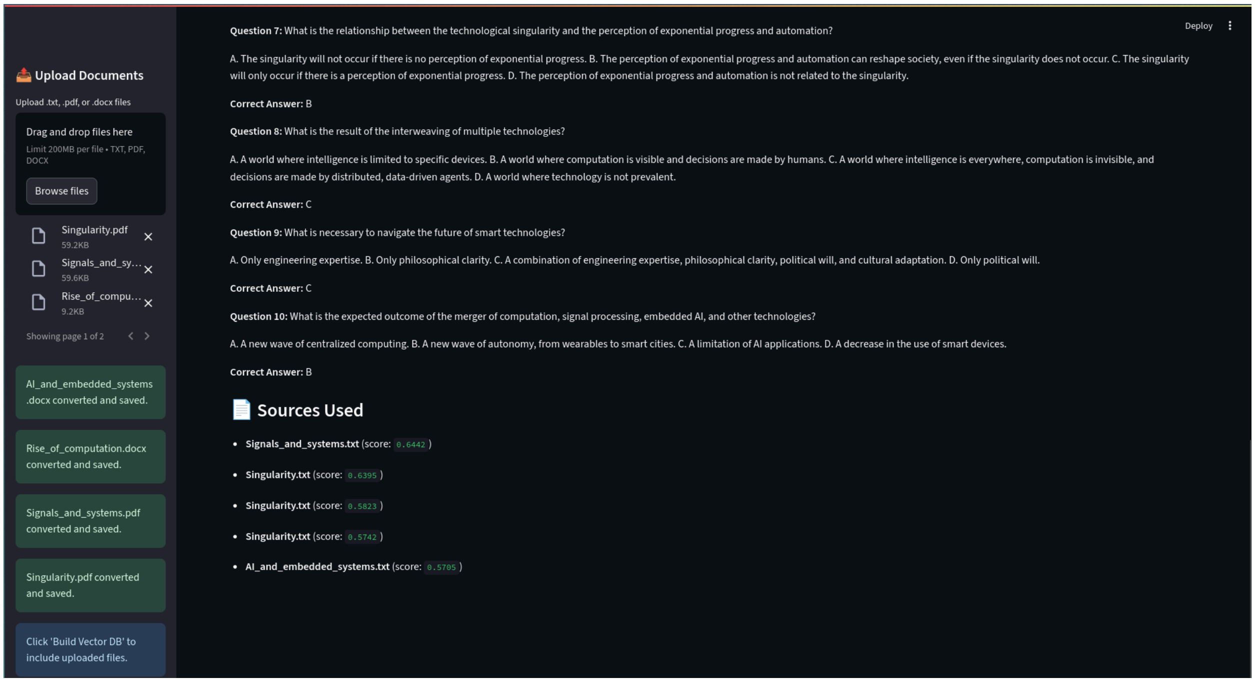

Upload Documents
Upload .txt, .pdf, or .docx files
Drag and drop files here
Limit 200MB per file • TXT, PDF, DOCX
Browse files
Singularity.pdf
59.2KB
Signals_and_sy...
59.6KB
Rise_of_compu...
9.2KB
Showing page 1 of 2
AI_and_embedded_systems.docx converted and saved.
Rise_of_computation.docx converted and saved.
Signals_and_systems.pdf converted and saved.
Singularity.pdf converted and saved.
Click 'Build Vector DB' to include uploaded files.
Deploy
The technological singularity refers to a hypothetical point in time when artificial intelligence (AI) surpasses human intelligence, leading to exponential growth in technological advancements, potentially transforming society in profound ways. This concept is speculative but has been increasingly discussed due to recent advances in AI.
Step 2: Role of Embedded AI Systems
Embedded AI systems refer to the integration of AI capabilities into devices and systems that are not traditionally considered computers, such as wearables, smart home devices, and industrial machinery. These systems enable devices to perceive their environment, make decisions, and act autonomously.
Step 3: Connection to Decentralization of Intelligence
The advancement of embedded AI systems contributes to the decentralization of intelligence. Instead of relying solely on centralized AI clouds, embedded AI distributes intelligence across billions of devices. This decentralization is a crucial step toward a potential singularity, as it suggests a future where intelligence is not confined to a few powerful entities but is pervasive.
Step 4: Advancements in AI and Embedded Systems
Recent advancements in large language models (LLMs), multimodal reasoning, and self-improving architectures are key drivers. When these advancements are integrated into embedded systems, they enhance the ability of devices to learn, adapt, and improve over time. This integration pushes the boundaries of what is possible with AI, bringing us closer to a scenario where machines can surpass human intelligence.
Step 5: Pathway to the Singularity
The pathway to the technological singularity, if it occurs, is likely to be a mosaic of innovations. Embedded AI systems play a critical role in this mosaic by:
Enabling Autonomy: Devices with embedded AI can operate autonomously, making decisions without human intervention.
Decentralizing Computing: By reducing reliance on cloud infrastructure, embedded AI decentralizes computing, making AI more ubiquitous and resilient.
Merging with Other Technologies: The future convergence of AI, quantum computing, biology, and global networks could accelerate progress toward the singularity.
Step 6: Conclusion
Embedded AI systems contribute to getting closer to the technological singularity by decentralizing intelligence, enabling autonomy at the device level, and forming a critical part of the interwoven mosaic of innovations that could lead to a future where machines surpass human intelligence.
Action:
Based on this reasoning, the integration and advancement of embedded AI systems are essential steps in the journey toward the technological singularity. They not only enhance the capabilities of devices but also contribute to a future where intelligence is pervasive, computation is invisible, and decisions are increasingly made by distributed, data-driven agents. Therefore, continued research and development in embedded AI are likely to play a pivotal role in shaping the trajectory toward or away from the singularity, depending on human choices and actions.
Sources Used


Upload Documents
Upload .txt, .pdf, or .docx files
Drag and drop files here
Limit 200MB per file • TXT, PDF, DOCX
Browse files
Singularity.pdf
59.2KB
Signals_and_sy...
59.6KB
Rise_of_compu...
9.2KB
Showing page 1 of 2
AI_and_embedded_systems.docx converted and saved.
Rise_of_computation.docx converted and saved.
Signals_and_systems.pdf converted and saved.
Singularity.pdf converted and saved.
Click 'Build Vector DB' to include uploaded files.
Deploy
Recent advancements in large language models (LLMs), multimodal reasoning, and self-improving architectures are key drivers. When these advancements are integrated into embedded systems, they enhance the ability of devices to learn, adapt, and improve over time. This integration pushes the boundaries of what is possible with AI, bringing us closer to a scenario where machines can surpass human intelligence.
Step 5: Pathway to the Singularity
The pathway to the technological singularity, if it occurs, is likely to be a mosaic of innovations. Embedded AI systems play a critical role in this mosaic by:
Enabling Autonomy: Devices with embedded AI can operate autonomously, making decisions without human intervention.
Decentralizing Computing: By reducing reliance on cloud infrastructure, embedded AI decentralizes computing, making AI more ubiquitous and resilient.
Merging with Other Technologies: The future convergence of AI, quantum computing, biology, and global networks could accelerate progress toward the singularity.
Step 6: Conclusion
Embedded AI systems contribute to getting closer to the technological singularity by decentralizing intelligence, enabling autonomy at the device level, and forming a critical part of the interwoven mosaic of innovations that could lead to a future where machines surpass human intelligence.
Action:
Based on this reasoning, the integration and advancement of embedded AI systems are essential steps in the journey toward the technological singularity. They not only enhance the capabilities of devices but also contribute to a future where intelligence is pervasive, computation is invisible, and decisions are increasingly made by distributed, data-driven agents. Therefore, continued research and development in embedded AI are likely to play a pivotal role in shaping the trajectory toward or away from the singularity, depending on human choices and actions.
Sources Used
Signals_and_systems.txt (score: 0.6442)
Singularity.txt (score: 0.6395)
Singularity.txt (score: 0.5823)
Singularity.txt (score: 0.5742)
AI_and_embedded_systems.txt (score: 0.5705)

12. There is also a dedicated MCQ mode to generate 10 questions.







Design Strengths

- **Modular and Extensible**: Clean separation of concerns between chunking, embedding, storage, retrieval, and reasoning.
- **Interactive UI**: Simple Streamlit interface for experimentation and input customization.
- **Advanced Prompting**: Supports hybrid reasoning methods to guide LLM output.
- **Multi-format Document Support**: Accepts standard file formats with robust text extraction.

Limitations & Recommendations

Issue	Recommendation
Static Embedding Model	Consider upgrading to a more powerful embedding model like text-embedding-3-small if latency and cost permit.
No persistent state across sessions	Integrate persistent storage or session state to avoid repeated rebuilds.
Basic error handling	Improve API failure transparency (e.g., include status codes, retry logic).
Chunking is token-based, not semantic	Implement sentence-aware or semantic chunking using tools like nltk, spaCy, or langchain.text_splitter.
Prompt is purely template-based	Allow manual editing or preview of final prompt before submission.

Security Considerations

- **API Key Management**: Relies on st.secrets; ensure .streamlit/secrets.toml is properly secured.

- **File Upload Handling**: No malicious content scanning—consider sandboxing or filtering file types further.

- **LLM Output Filtering**: No post-processing of LLM output. For production, consider moderation or filtering.

Conclusion

This system demonstrates a practical and flexible implementation of retrieval-augmented generation using a local vector store and hosted LLM. It is well-suited for domain-specific document comprehension, educational tooling, or internal knowledge base querying. Future improvements could include deeper semantic processing, session-based tracking, and richer feedback integration from users.

– Amogh K Umesh
Data Engineering Batch
NexTurn Intern 2025