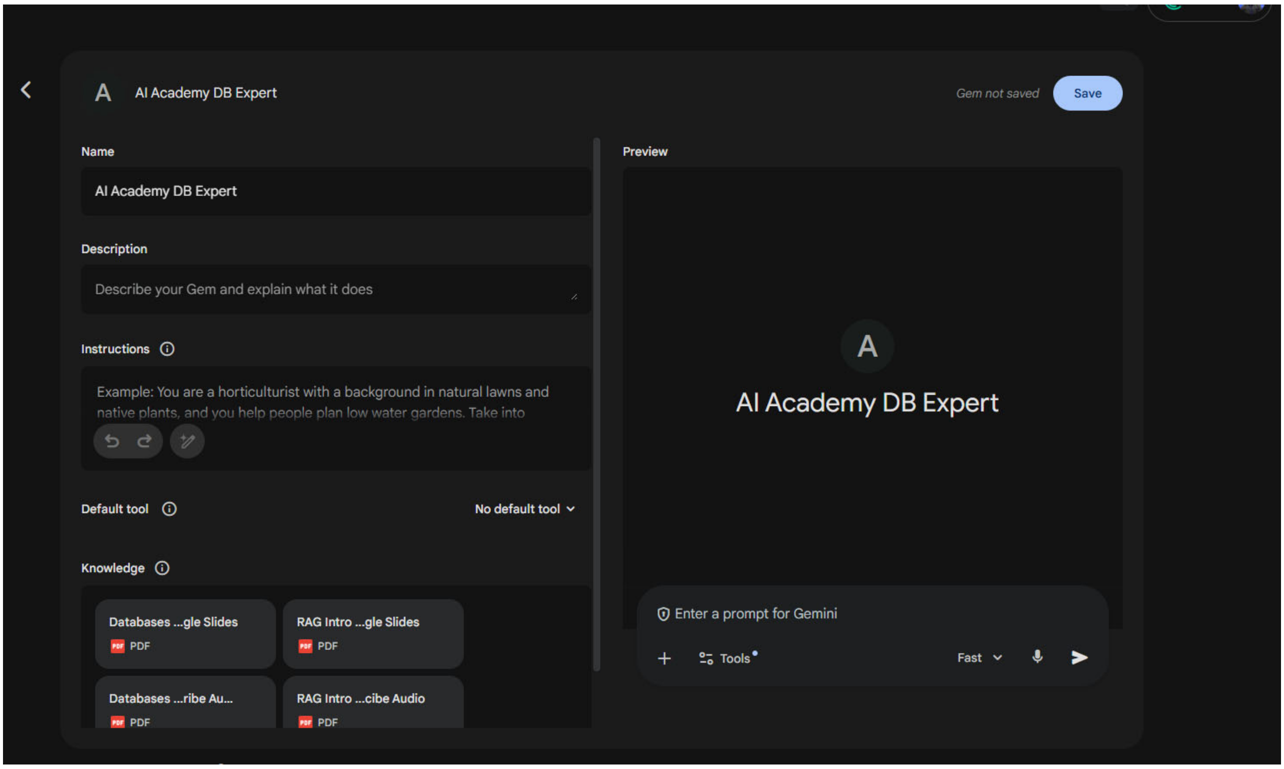
📑Module 4 Homework [AI Academy]

The Gem is designed to assist developers and data architects in understanding how to design, implement, and optimize Retrieval-Augmented Generation (RAG) systems and vector databases for generative AI applications.

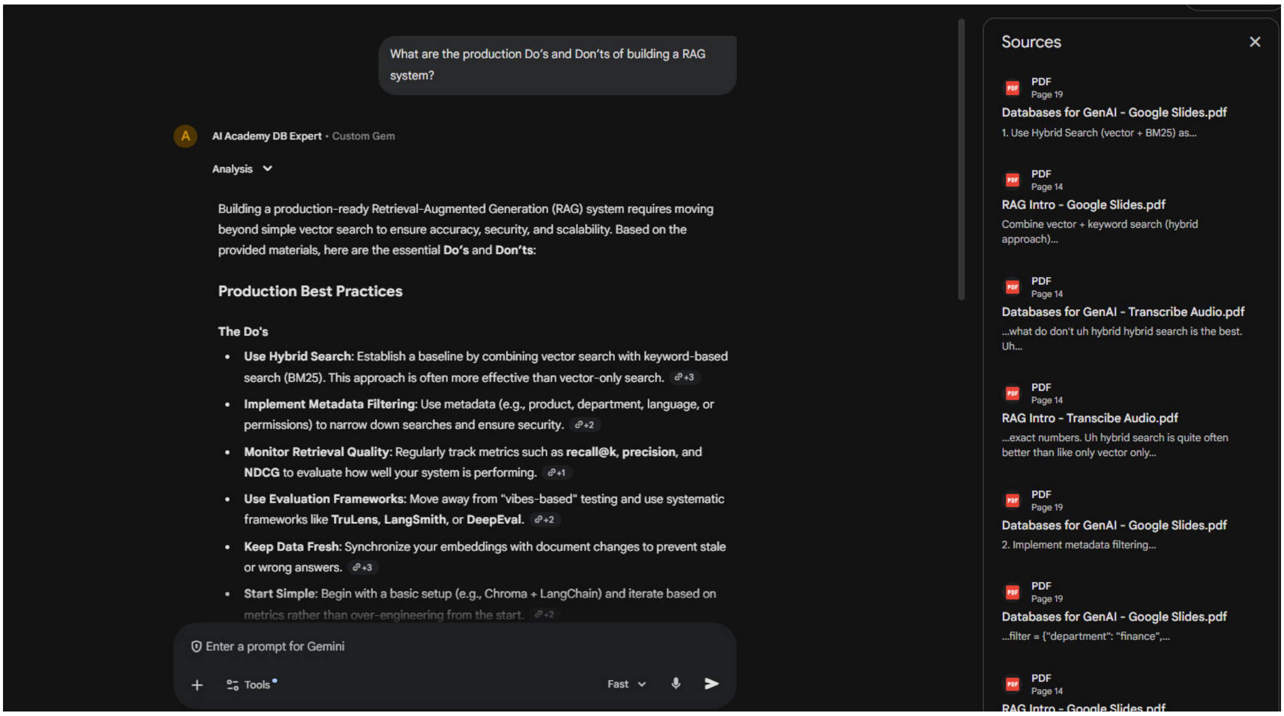
It specializes in explaining RAG architecture, chunking strategies, and hybrid retrieval methods. Additionally, it provides guidance on embedding model selection, vector database performance optimization, and agentic RAG patterns. The Gem is instructed to reduce hallucinations by grounding every response strictly in the Databases for GenAI and RAG Intro materials.
After configuring the instructions and uploading the relevant documents, I began testing it by asking simple introductory questions to evaluate its behavior and accuracy.

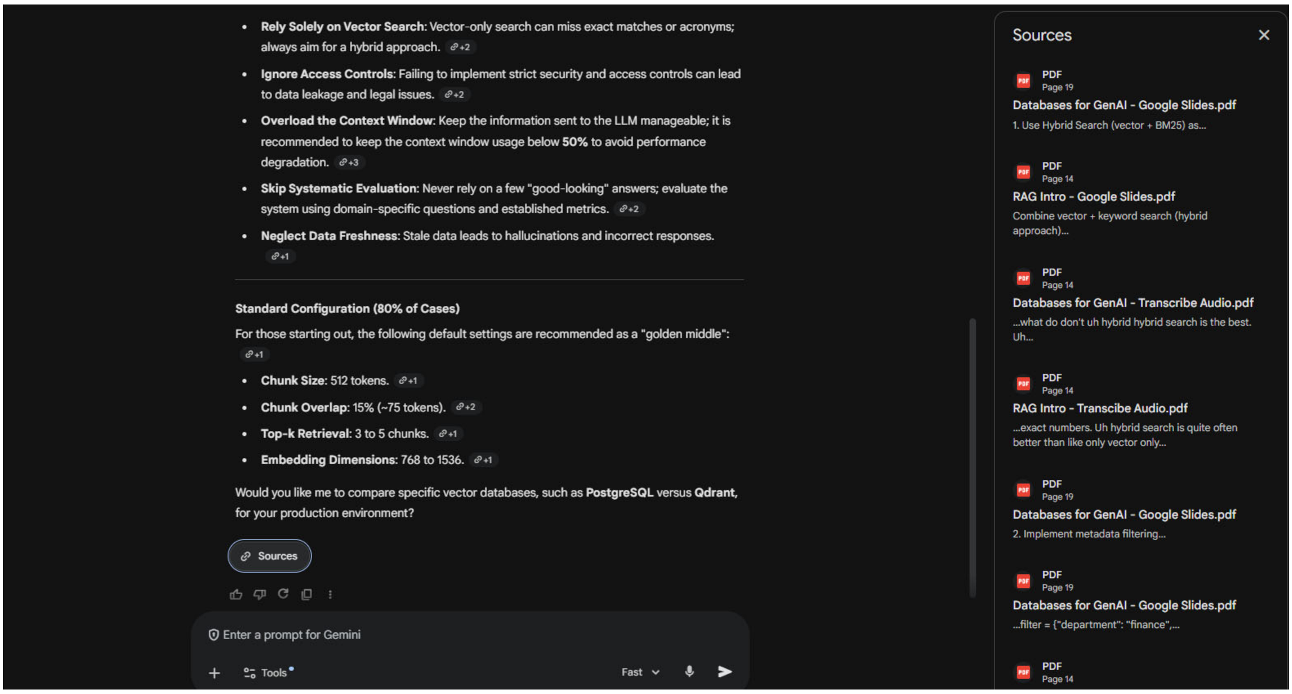
➔ Creating My Gem



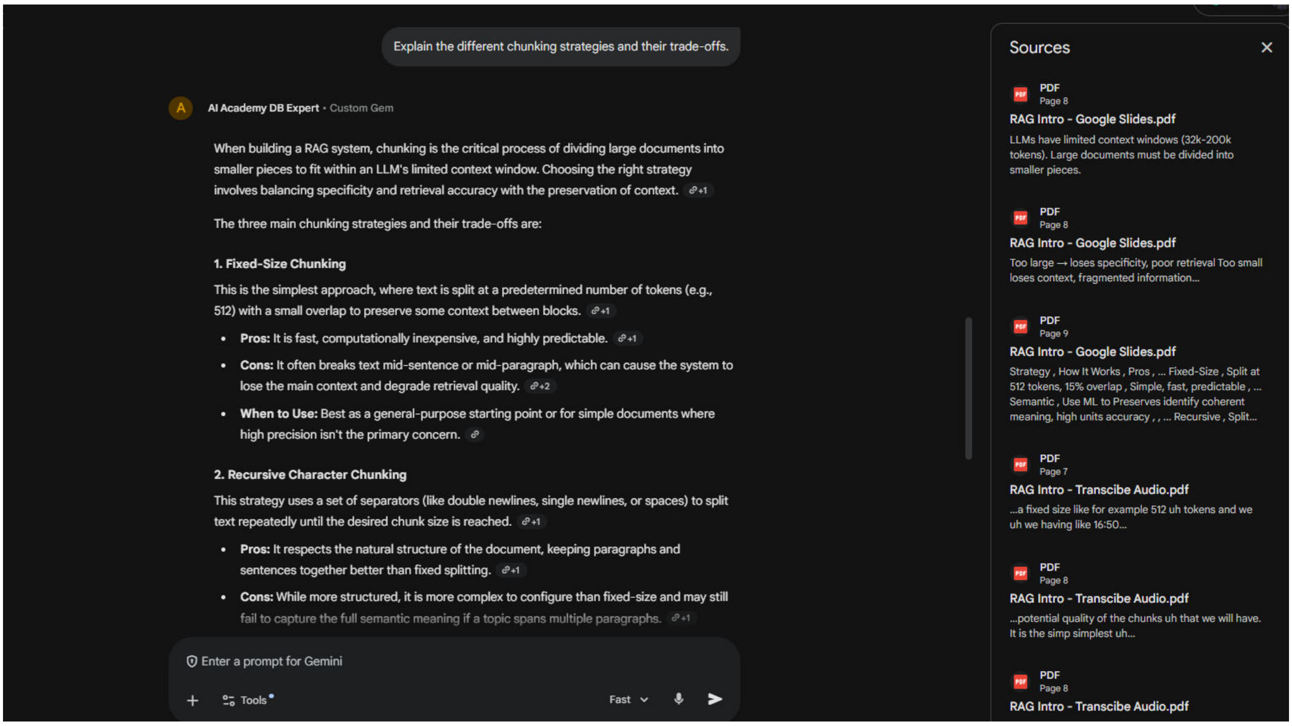
➔ Started with simple question

1. What are the production Do’s and Don’ts of building a RAG system?





2. Explain the different chunking strategies and their trade-offs.



Explain the different chunking strategies and their trade-offs.

AI Academy DB Expert • Custom Gem

When building a RAG system, chunking is the critical process of dividing large documents into smaller pieces to fit within an LLM's limited context window. Choosing the right strategy involves balancing specificity and retrieval accuracy with the preservation of context. +1

The three main chunking strategies and their trade-offs are:

1. Fixed-Size Chunking

This is the simplest approach, where text is split at a predetermined number of tokens (e.g., 512) with a small overlap to preserve some context between blocks. +1

- **Pros:** It is fast, computationally inexpensive, and highly predictable. +1
- **Cons:** It often breaks text mid-sentence or mid-paragraph, which can cause the system to lose the main context and degrade retrieval quality. +2
- **When to Use:** Best as a general-purpose starting point or for simple documents where high precision isn't the primary concern.

2. Recursive Character Chunking

This strategy uses a set of separators (like double newlines, single newlines, or spaces) to split text repeatedly until the desired chunk size is reached. +1

- **Pros:** It respects the natural structure of the document, keeping paragraphs and sentences together better than fixed splitting. +1
- **Cons:** While more structured, it is more complex to configure than fixed-size and may still fail to capture the full semantic meaning if a topic spans multiple paragraphs. +1

Enter a prompt for Gemini

Tools Fast

Sources

PDF Page 8
RAG Intro - Google Slides.pdf
LLMs have limited context windows (32k-200k tokens). Large documents must be divided into smaller pieces.

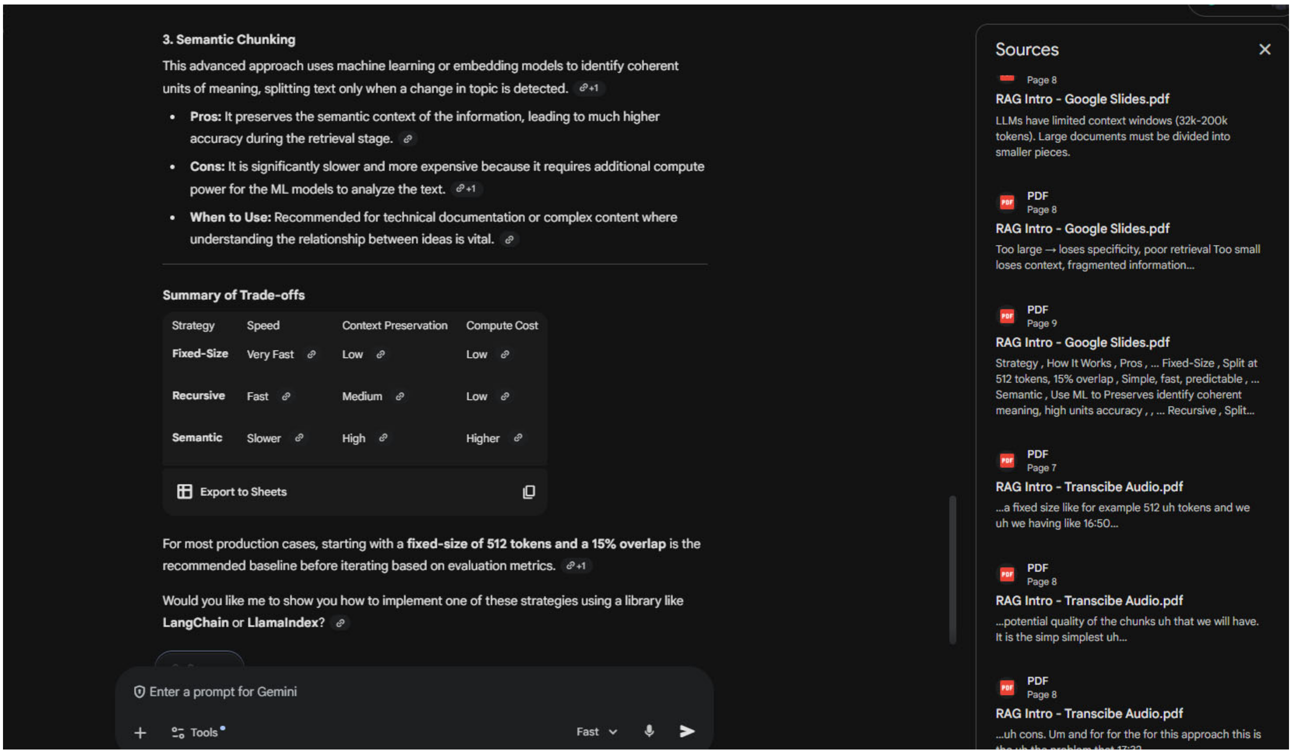
PDF Page 8
RAG Intro - Google Slides.pdf
Too large → loses specificity, poor retrieval Too small loses context, fragmented information...

PDF Page 9
RAG Intro - Google Slides.pdf
Strategy , How It Works , Pros , ... Fixed-Size , Split at 512 tokens, 15% overlap , Simple, fast, predictable , ... Semantic , Use ML to Preserves identify coherent meaning, high units accuracy , , ... Recursive , Split...

PDF Page 7
RAG Intro - Transcibe Audio.pdf
...a fixed size like for example 512 uh tokens and we uh we having like 16:50...

PDF Page 8
RAG Intro - Transcibe Audio.pdf
...potential quality of the chunks uh that we will have. It is the simp simplest uh...

PDF Page 8
RAG Intro - Transcibe Audio.pdf



3. Semantic Chunking

This advanced approach uses machine learning or embedding models to identify coherent units of meaning, splitting text only when a change in topic is detected. +1

- **Pros:** It preserves the semantic context of the information, leading to much higher accuracy during the retrieval stage.
- **Cons:** It is significantly slower and more expensive because it requires additional compute power for the ML models to analyze the text. +1
- **When to Use:** Recommended for technical documentation or complex content where understanding the relationship between ideas is vital.

Summary of Trade-offs

Strategy	Speed	Context Preservation	Compute Cost
Fixed-Size	Very Fast	Low	Low
Recursive	Fast	Medium	Low
Semantic	Slower	High	Higher

Export to Sheets

For most production cases, starting with a **fixed-size of 512 tokens and a 15% overlap** is the recommended baseline before iterating based on evaluation metrics. +1

Would you like me to show you how to implement one of these strategies using a library like **LangChain** or **LlamaIndex**?

Enter a prompt for Gemini

Tools Fast

Sources

Page 8
RAG Intro - Google Slides.pdf
LLMs have limited context windows (32k-200k tokens). Large documents must be divided into smaller pieces.

PDF Page 8
RAG Intro - Google Slides.pdf
Too large → loses specificity, poor retrieval Too small loses context, fragmented information...

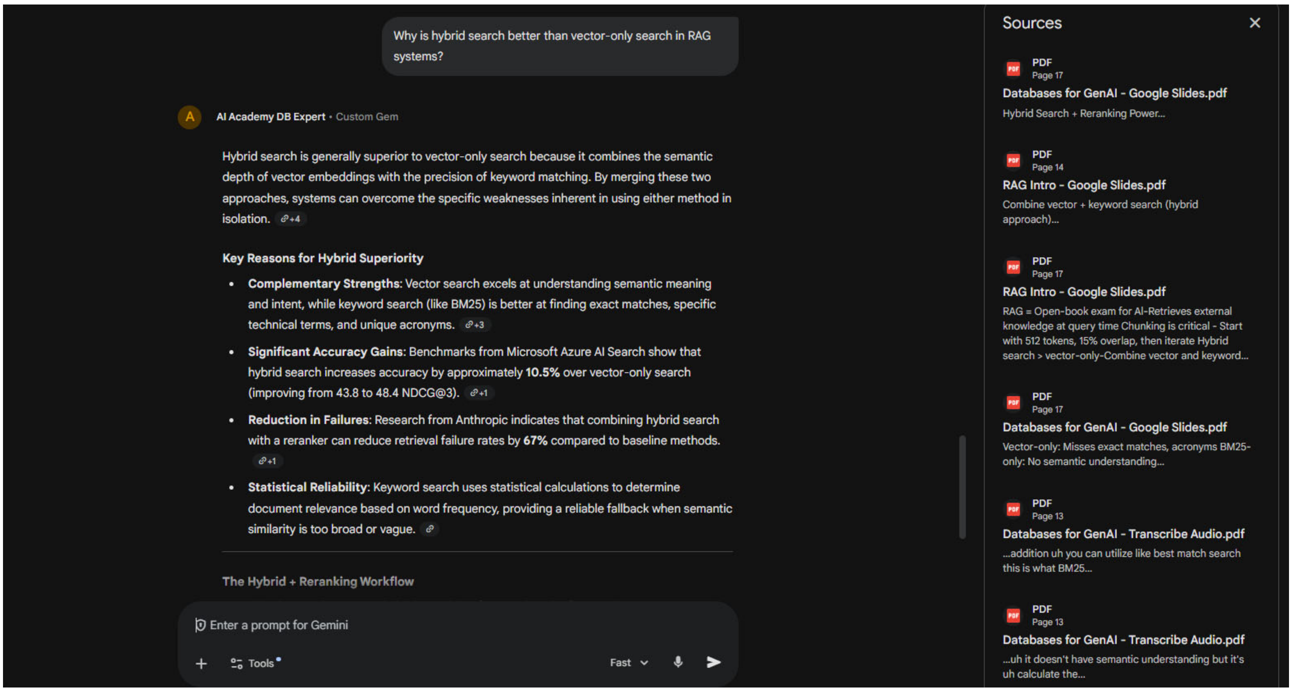
PDF Page 9
RAG Intro - Google Slides.pdf
Strategy , How It Works , Pros , ... Fixed-Size , Split at 512 tokens, 15% overlap , Simple, fast, predictable , ... Semantic , Use ML to Preserves identify coherent meaning, high units accuracy , , ... Recursive , Split...

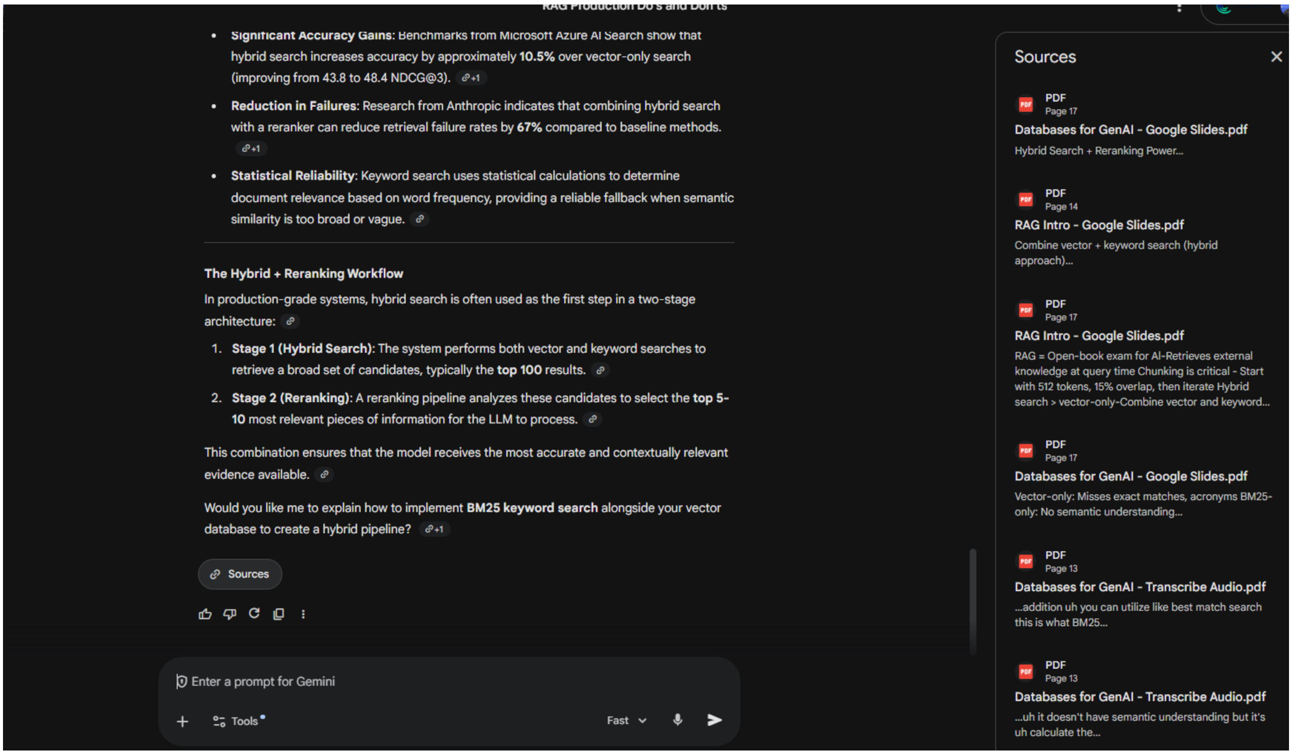
PDF Page 7
RAG Intro - Transcibe Audio.pdf
...a fixed size like for example 512 uh tokens and we uh we having like 16:50...

PDF Page 8
RAG Intro - Transcibe Audio.pdf
...potential quality of the chunks uh that we will have. It is the simp simplest uh...

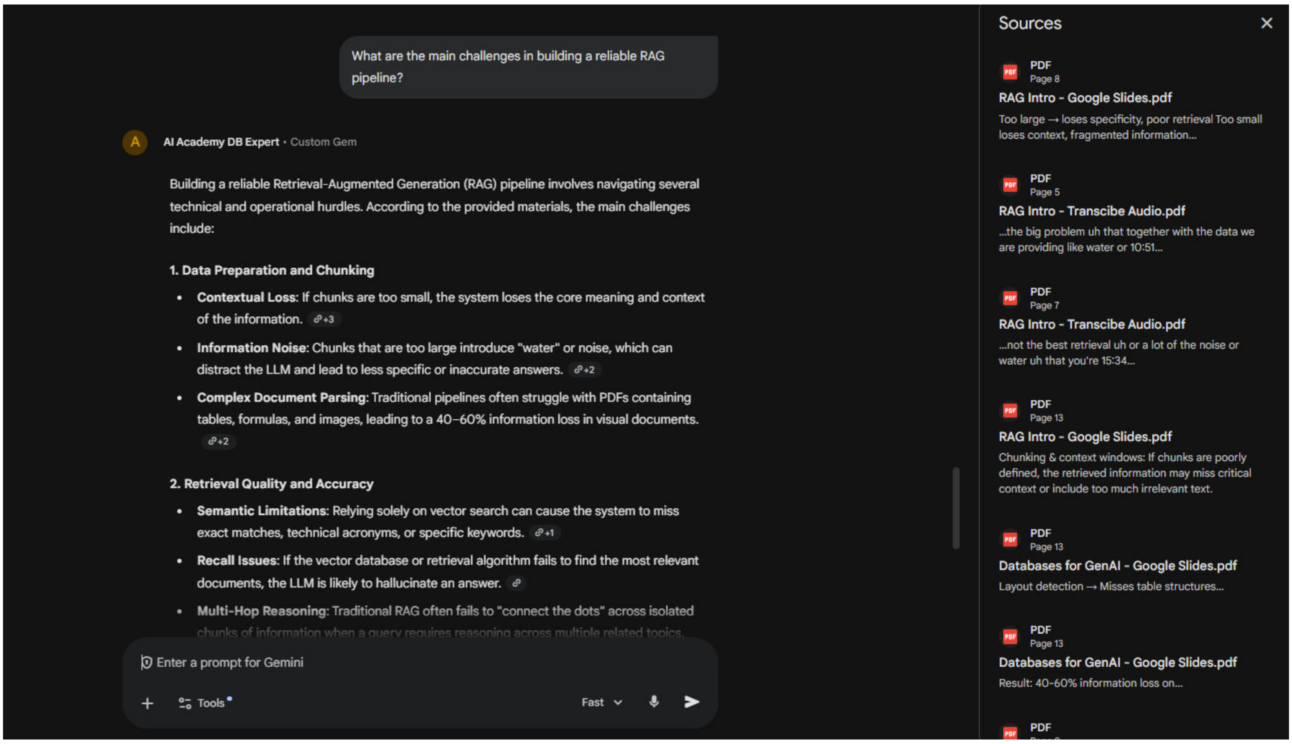
PDF Page 8
RAG Intro - Transcibe Audio.pdf
...uh cons. Um and for for the for this approach this is

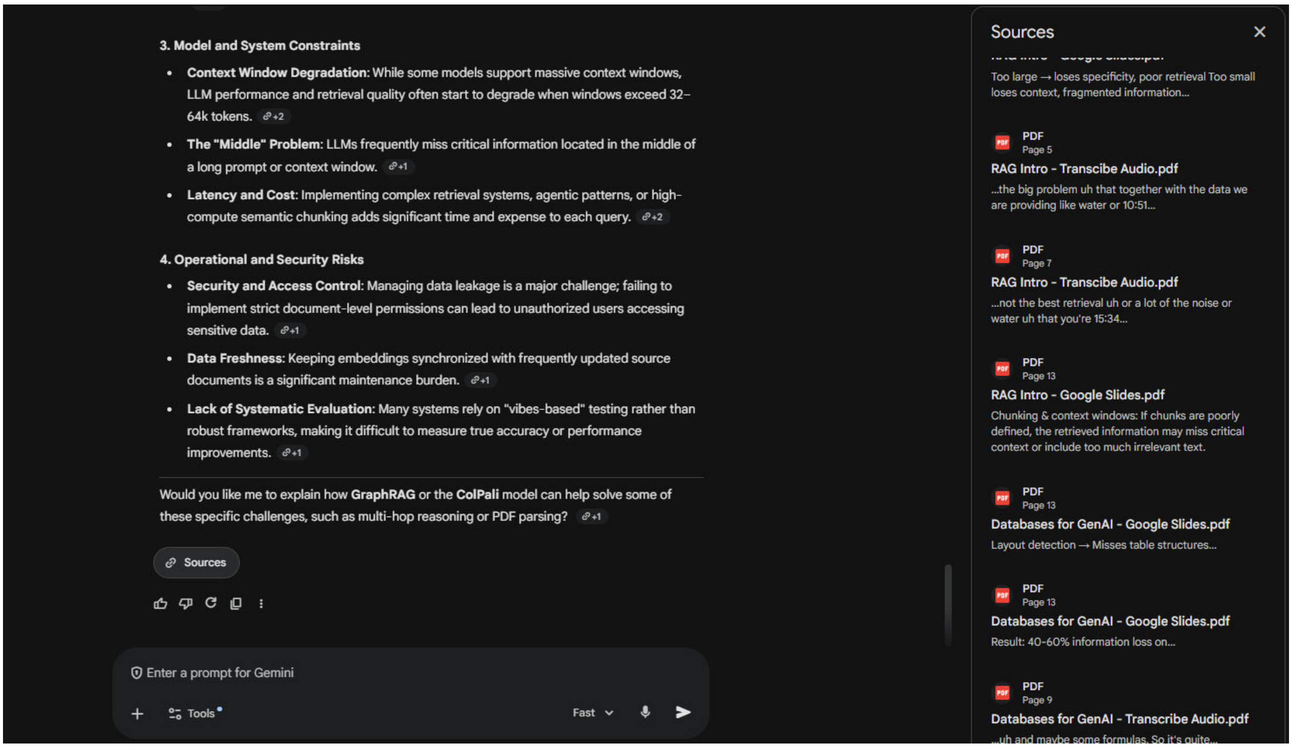
3. Why is hybrid search better than vector-only search in RAG systems?





4. What are the main challenges in building a reliable RAG pipeline?





Reflection on Gem Performance

1. How well did your Gem answer the questions? Was it accurate?

My Gem answered the questions accurately and successfully retrieved relevant information from the uploaded PDFs. It clearly explained topics like chunking strategies, hybrid search, and RAG architecture using correct details from the materials. However, even after instructing it to keep responses concise and include citations, it did not always strictly follow those formatting rules. Despite this, the answers remained relevant and grounded in the provided documents.

2. What were the limitations of providing knowledge by copying and pasting it into the instructions?

Copying and pasting the knowledge into the instructions limited flexibility and scalability. While the Gem could recall factual information, it sometimes struggled with maintaining concise formatting or properly citing sources. Managing large volumes of text in the instruction field can also become difficult and less structured. This approach works for smaller datasets but may not be ideal for more complex or dynamic knowledge bases.

3. What kind of “prompt engineering” did you have to do in the instructions to make the Gem behave correctly?

I clearly defined the Gem’s role as a “GenAI RAG Database Expert” and instructed it to only use the uploaded PDFs. I added rules to avoid hallucinations, provide concise answers (maximum five lines), and cite document sections when possible. I also specified the tone and response style. These structured instructions helped improve focus and reduce irrelevant or overly long responses.

4. How is this approach different from just asking the standard Gemini model the same questions?

This customized Gem provided focused, document-specific answers based strictly on the uploaded materials. In contrast, the standard Gemini model would respond using its broader general knowledge, which may be less aligned with the specific course content. The Gem approach ensures better grounding and relevance to the assignment materials.