Capstone Documentation

Retrieval-Augmented Generation (RAG) for Question Answering

1. Project Overview

This project implements a **Retrieval-Augmented Generation (RAG)** system to answer natural language questions over a small corpus of AI research papers. The system combines **semantic retrieval** with **LLM-based generation** to produce **grounded answers** along with **source attribution** (document name + page/chunk reference).

The corpus includes three AI research papers:

- *Attention Is All You Need*
- *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*
- *Language Models are Few-Shot Learners*

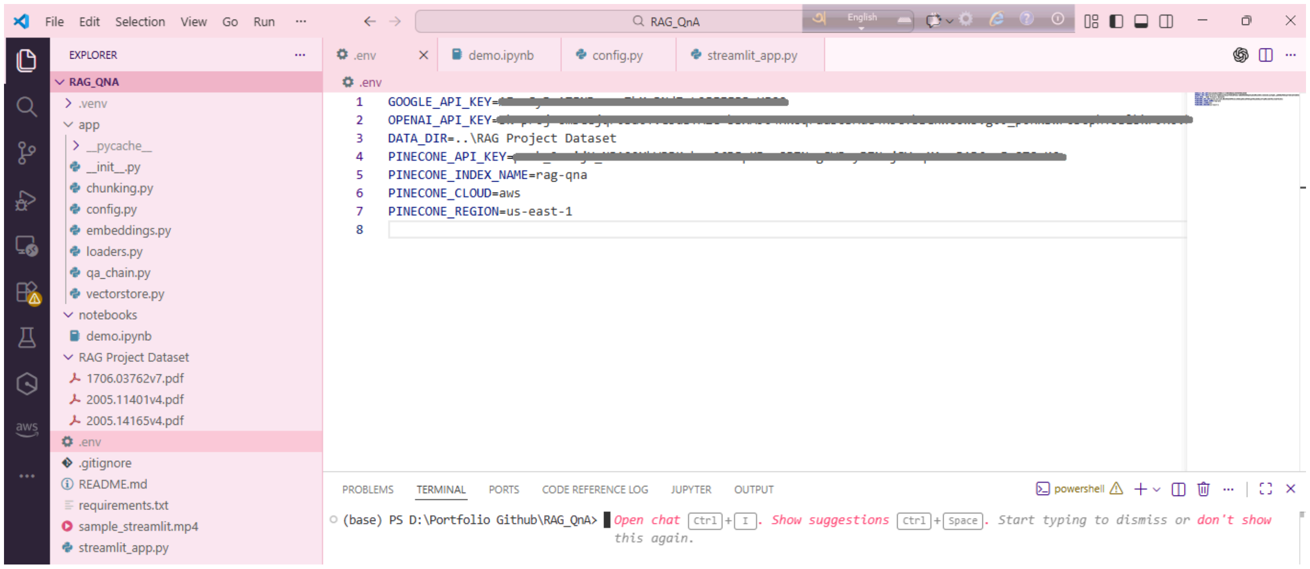
The capstone focuses on end-to-end GenAI application: **document processing → retrieval → prompting → validated output**.

2. Submission Artifacts

The final submission contains:

1. **Demo Notebook** (`.ipynb`)
    - End-to-end RAG pipeline execution
    - Includes ingestion, chunking, embedding, retrieval, QA calls, and example outputs
2. **Python source files (** `.py` **) + Streamlit app files**
    - Modular code for:
        - preprocessing
        - ingestion
        - vector DB upload/query
        - QA interface
    - Streamlit UI to run the QA system interactively
3. **Documentation PDF** (this report)

4. Sample Query Video



```
GOOGLE_API_KEY=[redacted]
OPENAI_API_KEY=[redacted]
DATA_DIR=..\RAG Project Dataset
PINECONE_API_KEY=[redacted]
PINECONE_INDEX_NAME=rag-qna
PINECONE_CLOUD=aws
PINECONE_REGION=us-east-1
```

```
(base) PS D:\Portfolio Github\RAG_QnA> Open chat Ctrl+I. Show suggestions Ctrl+Space. Start typing to dismiss or don't show
                                       this again.
```

All files are zipped and submitted as instructed.

3. System Architecture

The system follows a standard RAG pipeline:

```
PDF Papers
  ↓
Text Extraction
  ↓
Chunking + Metadata (doc name, page, chunk id)
  ↓
Embeddings
  ↓
Vector DB (Pinecone)
  ↓
Retriever (top-k semantic search)
  ↓
Grounded Prompt + LLM (OpenAI)
  ↓
Final Answer + Source Attribution
```

This modular design ensures:

- clean separation between retrieval and generation
- traceable answers with sources
- easy expansion to more papers later

4. Document Processing & Chunking

4.1 PDF Loading

The three research papers are loaded from PDF format and converted into text. Metadata is preserved to support source attribution:

- document name
- page number (when available)
- chunk index / chunk id

4.2 Chunking Strategy

Text is split into overlapping chunks using:

- **Chunk size:** ~800 tokens (or equivalent character-based split)
- **Overlap: 300 tokens**

Rationale (why 300 overlap)

- Research papers often explain one concept across multiple paragraphs.
- Higher overlap reduces the chance that important definitions (e.g., positional encoding, attention equations) get split across chunks and lost during retrieval.
- This increases recall for conceptual questions, which is essential for QA quality.

5. Embeddings & Vector Database

5.1 Embeddings

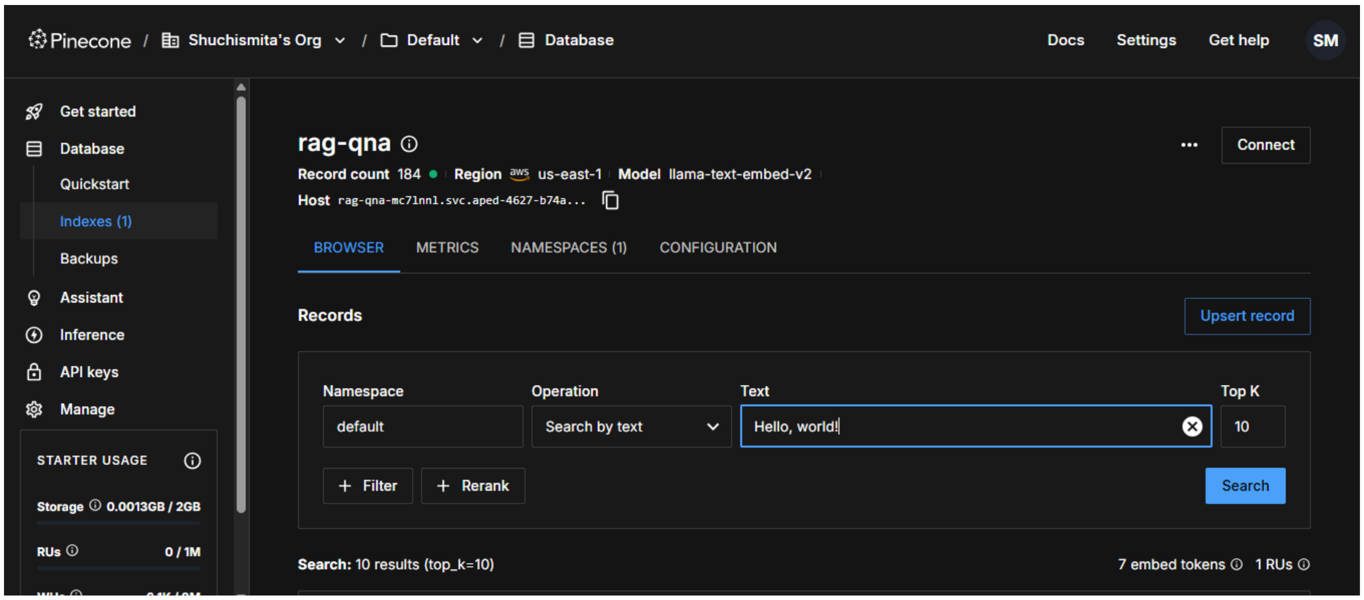
Embeddings are created for each chunk to support semantic retrieval. Embeddings represent chunk meaning in vector form so the system can retrieve relevant content even if the question wording differs from the paper text.

5.2 Vector DB: Pinecone

All embeddings are stored in **Pinecone** as the vector database.
Why Pinecone

- Managed vector DB suitable for production-style RAG
- Fast semantic search and scalable indexing
- Convenient metadata filtering (document/page/chunk ids)
  Stored metadata per chunk includes:
- `doc_name`
- `page`
- `chunk_id`

- `chunk_text`



rag-qna
Record count 184 Region aws us-east-1 Model llama-text-embed-v2
Host rag-qna-mc71nnl.svc.aped-4627-b74a...

Records

Namespace: default
Operation: Search by text
Text: Hello, world!
Top K: 10

Search: 10 results (top_k=10) 7 embed tokens 1 RUs

STARTER USAGE
Storage 0.0013GB / 2GB
RUs 0 / 1M

6. Retrieval System Design

The retriever performs semantic similarity search over Pinecone.

- **Top-k retrieval:** typically `k = 6` (tunable)

Justification

- Multi-part conceptual questions require combining evidence from multiple sections.
- Increasing `k` improves recall and reduces "insufficient context" failures.
- The corpus is small, so `k=6` remains efficient while improving reliability.

7. Answer Generation & Prompt Strategy

7.1 LLM Choice (Important Note)

The original infrastructure guideline recommended Gemini. However, during implementation, **Gemini API calls produced runtime/API errors in the environment**. To ensure completion and stability, the system uses **OpenAI API** for answer generation.
This change was made to:

- keep the pipeline functional end-to-end
- ensure consistent QA outputs during evaluation

7.2 Grounded Prompt Strategy

The prompt is designed to enforce grounding and reduce hallucinations:

Key rules implemented in prompt:

- Use **only** retrieved context
- Combine multiple retrieved chunks when needed
- If context is insufficient → explicitly say so
- Keep answers concise and technical (research-paper style)
- Limit citations to the most relevant sources (1–3)

```
PROMPT_TEMPLATE = """

You are a research assistant answering questions based on academic papers.

Use ONLY the information from the provided context.

You may combine information from multiple context passages.

When answering:

- Be concise and technically precise.

- Avoid overgeneralization beyond what is stated in the papers.

- Cite ONLY the most relevant source passages (maximum 3).



If the answer is not supported by the context, say:

"I could not find sufficient information in the provided documents."



Context:

{context}

Question:

{question}

Answer:

"""
```

This matches the capstone requirement for responsible, validated outputs.

8. Source Attribution

Each answer includes **source attribution**:

- document name (PDF filename)
- page number(s) when available
- chunk id / reference
  To avoid noisy outputs, the system reports only the most relevant supporting chunks (typically top 1–3 sources).

This ensures:

- verifiable responses
- minimized hallucinations
- clear evaluator confidence

9. Limitations & Assumptions

Limitations

- Small corpus (3 papers) — works well for demonstration but limited coverage
- No explicit reranking stage (pure vector similarity retrieval)
- Retrieval quality depends on chunking boundaries and embedding model behavior
- Page-level attribution depends on PDF loader metadata fidelity

Assumptions

- The PDFs are authoritative
- Queries are within scope of the papers
- English-only

11. Conclusion

This capstone demonstrates a complete **RAG pipeline** using:

- robust chunking with high overlap (300) for research content
- **Pinecone** for semantic retrieval with metadata attribution
- **OpenAI LLM** for stable answer generation (Gemini fallback due to API issues)
- grounded prompting and explicit source attribution to ensure reliable QA outputs

The system meets all deliverables: end-to-end implementation, justified design choices, example outputs, and responsible answer generation.