**Project Title**

**Ontology-Driven Knowledge Retrieval System Using Generative AI and RAG**

**Business Goal**

To enable intelligent navigation and semantic search over large textual knowledge bases by leveraging automated ontology generation and Retrieval-Augmented Generation (RAG).

**Problem Statement**

Organizations often deal with massive, unstructured knowledge bases—such as books, documents, and internal wikis—that lack formal structure.

Building ontologies manually for such datasets is labour-intensive, error-prone, and does not scale.

This project aims to automate the ontology creation process using Generative AI techniques and employ it to enhance search within the corpus using a RAG-based architecture.

**Solution Architecture Overview**

The end-to-end system will follow the following logical flow:

**Text → Ontology Generation → Vector Store Ingestion → User Query → RAG-Based Answer Generation**

1. **Text Processing**: Ingest raw text from a freely available public-domain book.
2. **Ontology Creation**: Use LLMs to automatically infer and generate an ontology of key entities, relationships, and concepts from the corpus.
3. **Embedding and Storage**: Chunk and embed the content using state-of-the-art embedding models. Store the vectors in a scalable vector database.
4. **Query Handling**: Accept natural language queries from the user through an interactive UI.
5. **RAG Pipeline**: Retrieve relevant information using vector similarity and ontology data and generate final answers using a language model.
6. **UI Layer**: Implemented via Streamlit, allowing real-time interaction with the system.
7. Python 3.11.9

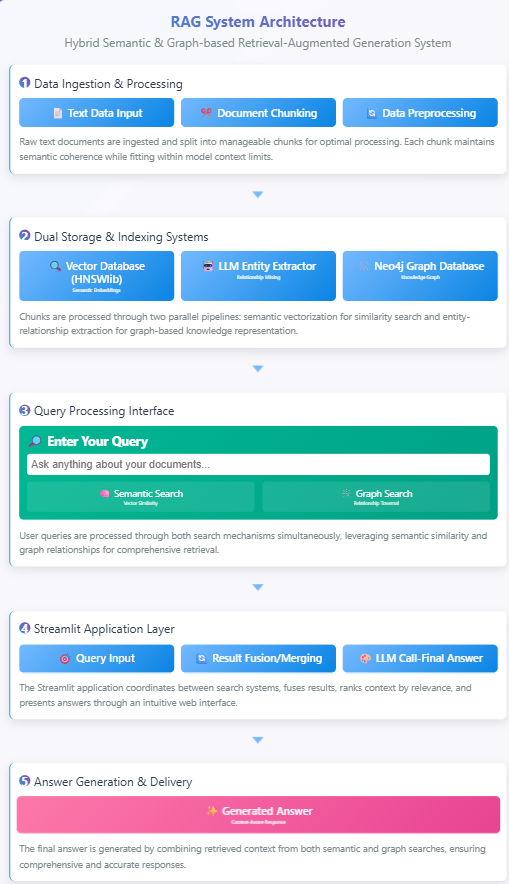
**Business Impact**

* Significantly improved ability to extract and search contextual information from large unstructured texts.
* Reduces the manual overhead of taxonomy and ontology creation.

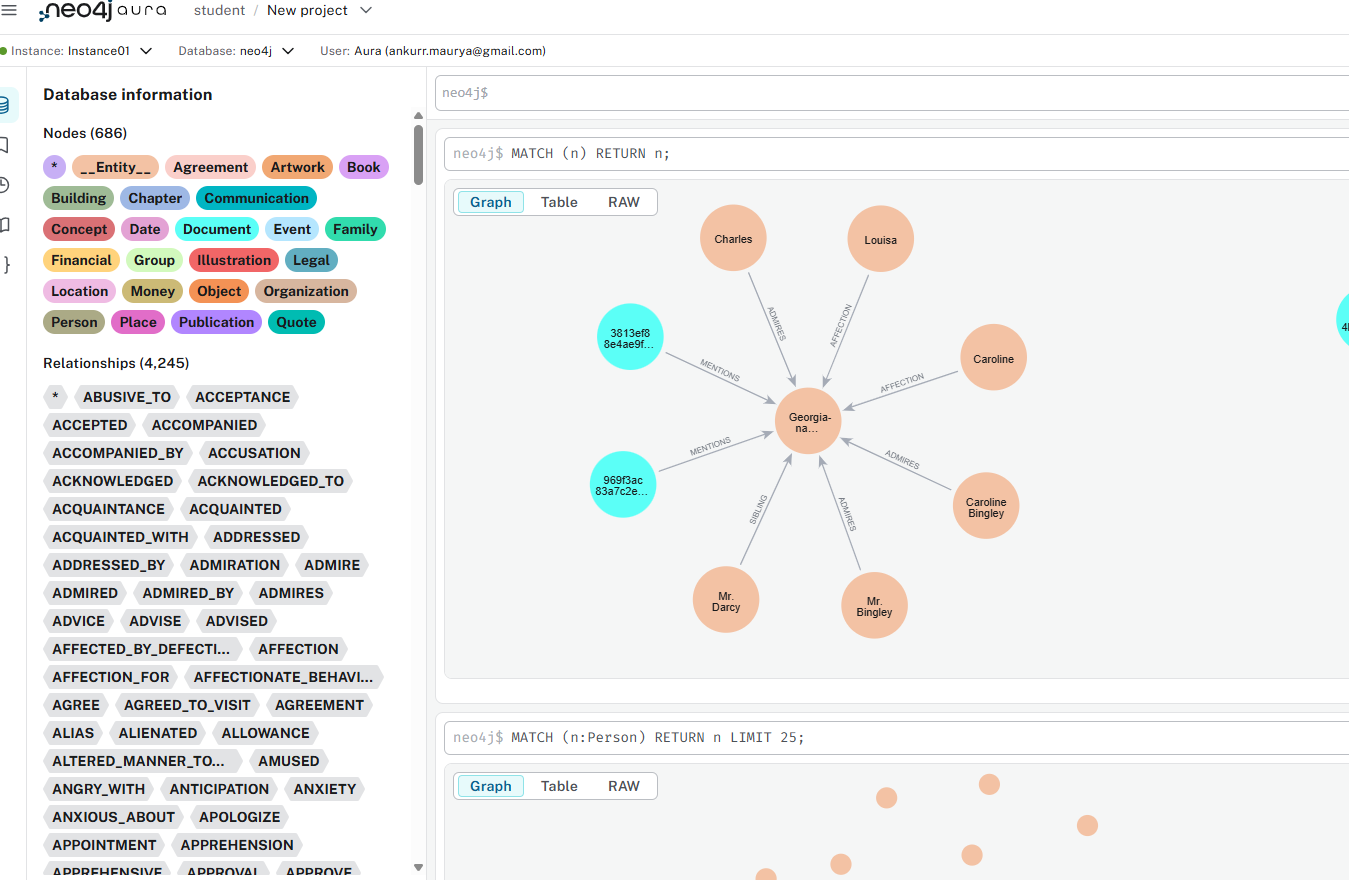
**Key Features:**

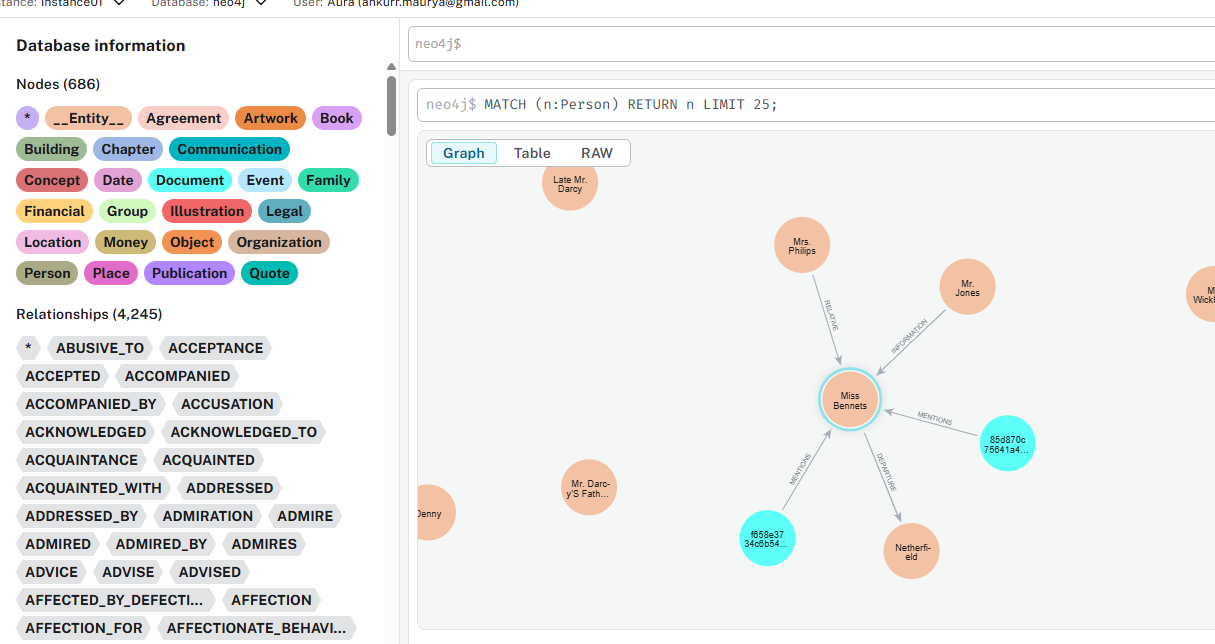
* **Data Ingestion**: Text processing and chunking
* **Dual Storage Systems**: Vector database (HNSWlib) for semantic search and Neo4j for graph-based relationships
* **Query Processing**: Interactive interface supporting semantic, graph, and hybrid search
* **Streamlit Application**: Orchestration layer that coordinates between systems
* **Answer Generation**: Final ANSWER.

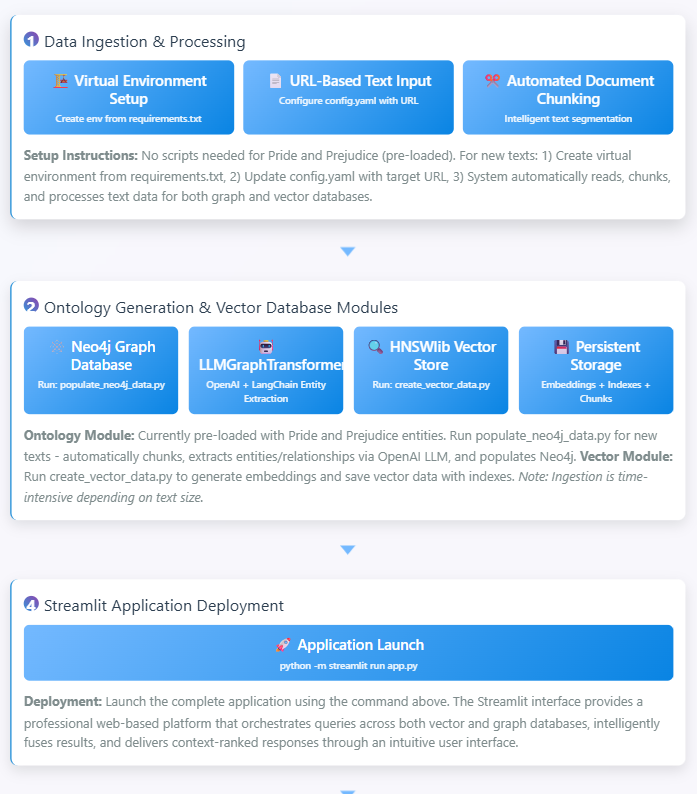
.



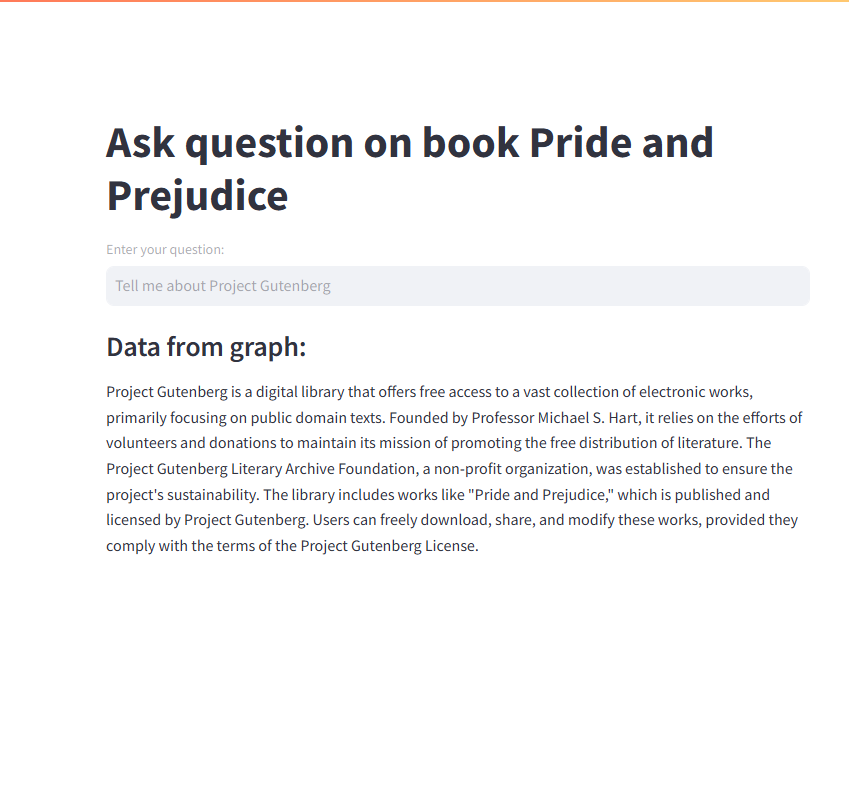
Neo4j Database (real data):

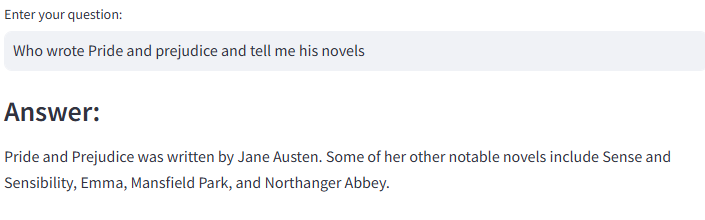


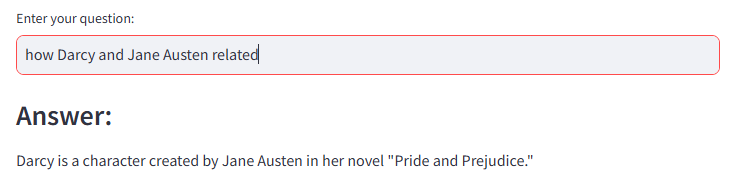


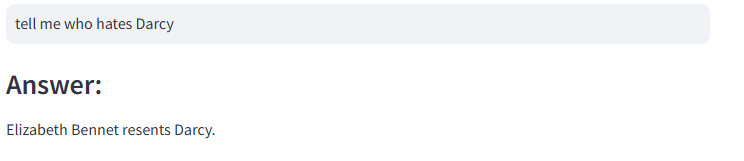


Few question and answer based application









\*For OpenAI credential please use your data and update YAML configuration accordingly.

Neo4j data was created using GPT-4o

Query answer was is done using gpt-4o-mini

**Challenges**: We can work on this.

### 1. ****Graph Retriever****

**Challenge**:  
It’s hard to find the right information from the ontology (graph of knowledge) that matches the user’s question.

Sometimes, the system looks at too many unrelated things (irrelevant node expansion), or not enough (shallow search), which brings in confusing or useless information. This makes it harder for the AI to give a clear and accurate answer.

Imagine this graph:

Instagram → owned\_by → Meta

Meta → has\_CEO → Mark Zuckerberg

Meta → founded\_by → Zuckerberg

Instagram → founded\_by → Kevin Systrom

**User asks**: “Who is the CEO of the company that owns Instagram?”

Without understanding, the system might return Kevin Systrom (he founded Instagram, but isn’t CEO now).

With query understanding and controlled traversal:

* First go from Instagram → owned\_by → Meta
* Then from Meta → has\_CEO → Mark Zuckerberg  
  Correct answer: Mark Zuckerberg.

### 2. ****Type of Question - Converting Question to Meaningful Graph Query****

**Challenge**:  
When people ask questions, they use natural language (like how we talk or type every day).

But the graph system needs the question in a special format to search it properly — like a graph query or a pattern (e.g., Cypher query).

If the system misunderstands the question, it ends up looking in the wrong part of the graph and gives the wrong answer.

**Example:**

**User asks**: “Who founded Google?”

But the system misunderstands and searches:  
**“Where is Google located?”**  
Now it gives the location instead of the founders. That’s poor question interpretation.

We fix this in two steps:

1. **Understand the type of question**  
   The system first figures out:
   * Is the user asking for a person (like a founder)?
   * A date, place, organization, or relationship?

So, “Who founded Google?” → Looking for a **person** connected to Google through a **founded\_by** relationship.

1. **Convert to a graph query**  
   The system turns the question into a graph-specific format.

For example:

* + From the natural question → create this graph query:  
    MATCH (p:Person)-[:FOUNDED]->(c:Company {name: "Google"}) RETURN p

1. **Add smart linking and disambiguation**  
   The system makes sure “Google” is correctly linked to the company (not the search engine or another product), and “founded” matches the right relation in the graph.

.

### 3. ****Semantic Search****

**Challenge**:  
Semantic search uses meanings of words (not exact words) to find answers. It uses things called **embeddings** — smart number codes that represent the meaning of text.

But sometimes, it finds things that sound similar but are not actually connected in the graph. This can confuse the system and lead to wrong or unrelated answers.

### ****Example of the Problem:****

**User asks**: “What drugs treat high blood pressure?”

Semantic search might return:

* “Aspirin” (because it's related to medicine)
* “Ibuprofen” (also related to treatment)

But these aren't used for blood pressure, so they are contextually similar but **wrong in structure** (not related through the right edges in the graph).

Solution:  
**Use hybrid search**  
Combine two things:

* **Semantic search**: Finds things with similar meaning
* **Graph-based search**: Checks if those things are actually connected the right way in the graph

**Apply filters**  
Add rules from the ontology (graph structure) — like:

* Only include answers that are connected through a relation like treats
* Ignore nodes that are just “medically similar” but not relevant

.

\*MANY content is generated by CHATGPT.

Target Python 3.11.9