Executive Summary and Implementation Report

Executive Summary

The **Document Analysis Hub** is a web-based application developed to address two primary challenges in applied AI: the effective analysis of large documents and the objective evaluation of Large Language Models (LLMs). The tool provides a dual-workflow interface built with Streamlit and powered by the Groq API for high-speed LLM inference.

The first workflow, *Thematic Document Q&A*, allows users to parse large PDF documents via their table of contents, enabling focused summarization and question-answering on specific chapters. The second workflow, *Model Evaluation & Analysis*, offers a comprehensive suite for comparing LLM performance on summarization tasks. This includes a novel, flexible evaluation module that allows users to choose between a reference-free "consensus" analysis and a traditional comparison against a user-provided "golden reference."

Key technical challenges—including API rate limiting, context window management, and conflicting evaluation metrics—were overcome through iterative design, leading to a robust and user-centric final product.

1 Implementation Process

The project was implemented as a Python-based Streamlit application, chosen for its ability to rapidly create interactive data science web applications.

1.1 Technology Stack

Component	**Technology/Library**	**Rationale**
Web Framework	Streamlit	Rapid UI development, native Python integration, ideal for data-centric apps.
LLM Inference	Groq API	Provided extremely high-speed inference for models like LLaMA 3 and Gemma.
PDF Processing	PyMuPDF (fitz)	Efficient and accurate text extraction from PDF documents.
Data Handling	Pandas	Used for structuring and displaying evaluation scores in clear, sortable tables.
Graphing	NetworkX & Matplotlib	To create and visualize the "diagraphs" of model similarity for consensus analysis.
NLP Utilities	tiktoken, sentence-transformers	For accurate token counting and high-speed local semantic search within large texts.

1.2 Core Workflows Implemented

The application is logically divided into two main workflows, selectable via a top-level radio button in the user interface.

Workflow A: Thematic Document Q&A

This workflow is designed for interacting with large, structured documents.

- **Input:** The user uploads a PDF and pastes its Table of Contents (ToC) into a text area.
- **Processing:** A custom function `preprocess_pdf_by_toc` parses the ToC, calculates page ranges for each section, and extracts the text into structured "chunks."
- **Interaction:** A dropdown menu allows users to select an "active section."
- **Summarization:** Clicking *Summarize* invokes `summarize_text_map_reduce` on the selected section's text.
- **Question-Answering:** Input from the chat is processed via `answer_question_within_section`, using only the active section's content.
- **State Management:** Section selection governs context for both summarization and Q&A. Switching sections resets history to prevent confusion, creating a seamless, connected user experience.

Workflow B: Model Evaluation & Short Doc Analysis

This workflow provides a comprehensive suite for evaluating LLM performance on small documents.

- **Input:** The user uploads a short PDF (1–7 pages).
- **Analysis Tools:**
  - **Quick Summarization:** Generates a summary using a selected model.
  - **Q&A Comparison:** Compares answers from multiple models to the same question.
  - **Comprehensive Summary Evaluation:**
    - **Reference-Free Consensus:** Summaries are generated by all models, pairwise semantic similarity is calculated, and the most "agreed upon" summary is identified and visualized via a graph.
    - **Compare to Golden Reference:** Allows the user to provide a reference summary for ROUGE-based evaluation, determining the best model relative to a gold standard.

2 Challenges Faced & Solutions

The development process was highly iterative, with each challenge prompting refinements that led to a more robust and intelligent design.

2.1 Challenge 1: Context Window Limitations

Problem: Initial implementations naively passed entire document sections to the LLM, frequently exceeding the model's context window (e.g., 8192 tokens for LLaMA 3).

Solution: A *Map-Reduce* strategy was implemented via the `summarize_text_map_reduce` function. Large texts are automatically split into smaller sub-chunks, each summarized individually ("Map" step), followed by a final synthesis that merges the partial outputs into a coherent whole ("Reduce" step).

2.2 Challenge 2: Balancing Performance and Reliability Under API Rate Limits

Problem: The Groq API's free tier imposes a strict Tokens-Per-Minute (TPM) rate limit. While the Map-Reduce strategy solved the context size problem, it introduced a new issue: rapid, successive API calls in the "Map" step frequently exceeded the TPM limit, resulting in `429 Too Many Requests` errors.

Trade-off Analysis: This challenge forced a direct confrontation with a classic engineering trade-off between efficacy and functionality.

- **Efficacy (Speed):** For an optimal user experience, sub-chunks should be processed in parallel or rapid sequence to minimize total response time.
- **Functionality (Reliability):** The API's rate limit enforces a slower call pattern. To avoid errors, request frequency must be reduced, prioritizing stability.

Solution:Strategic Throttling and Request Sizing We implemented a two-pronged solution to operate reliably within the API's constraints while minimizing the performance impact:

- **Request Size Management:** The internal threshold for request size (`MAX_TOKENS_FOR_SINGLE_REQUEST`) was reduced from near the model's context limit (e.g., 7500 tokens) to a safer value (e.g., 4000). This forces large texts into the Map-Reduce path and reduces the chance of exceeding the TPM limit.
- **Intelligent Throttling:** A `time.sleep(2)` delay was introduced between API calls during the Map-Reduce loop. This throttling ensures that the token rate remains below the TPM ceiling. A longer delay is also applied after any error to give the system time to recover before retrying.

Outcome: By tuning both request size and call frequency, the application now balances speed and reliability. It operates robustly under rate constraints, offering a predictable—though slower for very large documents—user experience.

2.3 Challenge 3: Ineffective QA Logic — The Evolution to a Connected and Context-Aware UI

The development of the Question-Answering (QA) feature within the Thematic QA workflow was an iterative process shaped by the need to balance usability, accuracy, and performance. The final implementation reflects significant architectural and UX improvements over earlier naive designs.

Attempt 1: Global Keyword Search (The "Naive" Approach)

Method: The first implementation allowed the user to ask a question, which was then tokenized into keywords. The system would search the text of every document section for these keywords and concatenate all matching sections to form a massive context.

Problem: This approach was fundamentally flawed. A simple question like "What are the rules?" could match the word "requirements" in multiple unrelated chapters (e.g., "Admission Requirements," "Graduation Requirements," "Course Prerequisites"). This created a polluted and often oversized context, leading to confused, inaccurate, or completely wrong answers from the LLM.

Attempt 2: Manual Section Selection (The "Accurate but Unusable" Approach)

Method: To solve the context pollution problem, the UI was changed to have two separate dropdowns: one for summarization and one for QA. The user was forced to manually select the specific section they believed contained the answer before asking a question

Problem: Although this guaranteed clean context, it disrupted the user experience. Users had to guess where the answer might reside, introducing unnecessary cognitive load and fragmenting the workflow across two disconnected interfaces.

Attempt 3: Final Connected UI with On-the-Fly Semantic Search (The "Intelligent" Approach) The final and most successful design combines the best aspects of the previous attempts—usability and accuracy—through a smarter, state-driven UI and an advanced context retrieval strategy.

The "Connected" User Interface The UI was redesigned to feature only a single, central dropdown menu labeled "Active Section." This selection, stored in Streamlit's session state (texttttst.session˙state.active˙section˙title), now governs the context for all tools on the page. When a user selects "History," both the "Summarize" button and the chat input box are now explicitly tied to that section. If the user changes the selection to "General information," the on-screen summary and chat history are automatically cleared, preventing confusion and ensuring the context is always clean and explicit. This creates a seamless and intuitive workflow where the user's focus is mirrored by the application's state.

On-the-Fly Semantic Search for Large Sections A key innovation was how the system handles questions for very large active sections that still exceed the LLM's context window. To solve this, we implemented an LLM-powered distillation process.

Step A: **Context Size Check:** Context Size Check: When a user asks a question, the system first uses tiktoken to check if the text of the active section exceeds the model's context token limit (e.g., 4000 tokens).

Step B: **Distillation Prompt:** If the context is too large, a new, specialized prompt is constructed. This prompt instructs the LLM to read the entire oversized text and extract only the information that is directly relevant to the user's specific question.

Step C: **Context Reduction via API Call:** The system makes an API call to the LLM using this distillation prompt. The LLM's response is a much smaller, highly concentrated text containing only the relevant facts.

Step D: **Final Answer Generation:** This distilled, perfectly-sized context is then used in a final prompt to generate a precise and accurate answer to the user's original question. This two-step process ensures that the model can answer questions from very large documents without encountering context window errors.

2.4 Challenge 4: Navigating the Ambiguity of "Best" — Conflicting Evaluation Metrics

Navigating the Ambiguity of ""Best"—Conflicting Evaluation Metrics A significant and defining challenge arose during the development of the evaluation workflow: different metrics frequently disagreed on which model produced the "best" summary. Early tests revealed that the model ranked highest by semantic similarity was often different from the one ranked highest by ROUGE. This was not a bug but a fundamental feature of model evaluation that required a sophisticated solution.

In-Depth Analysis of the Conflict The core of the issue lies in the different philosophies behind the evaluation metrics:

Root Cause: Two Metrics, Two Philosophies

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** This metric operates on lexical overlap. It measures the number of matching n-grams (sequences of words) between the model-generated summary and a human-produced reference. ROUGE is fundamentally a measure of precision and recall of specific wording. A high ROUGE score indicates that the model successfully extracted and used the same key phrases as the reference.

  The Dilemma: This metric's strength is also its weakness. Its reliance on a "golden" reference is problematic, as in most real-world scenarios, a standard, universally agreed-upon reference does not exist. Furthermore, it penalizes models that intelligently paraphrase or use different but semantically identical wording. For example, Llama3-70b-8192 often produced highly fluent, abstractive summaries that captured the meaning perfectly but scored lower on ROUGE because it didn't use the exact phrasing from the reference.

- **Semantic Similarity (Embedding-based Comparison):** This approach evaluates meaning by embedding each summary into a high-dimensional vector space and computing their pairwise distances.

  The Dilemma: In our reference-free consensus method, we used this to find the model with the highest average similarity to all others. This identifies the most "central" or "agreed-upon" summary. However, it doesn't guarantee factual precision in the same way ROUGE does. If most models hallucinate a similar incorrect fact, the consensus method might incorrectly favor that hallucination.

Solution for further: Embracing Nuance, Not Forcing a Winner

Faced with this conflict, we considered two strategies:

- **Fusion Method (Rejected):** One could attempt to fuse the summaries from all models into a single, new "meta-summary." While theoretically interesting, this adds another layer of potential LLM error and is computationally expensive.
- **Ranking Method (Adopted):** A more robust and explainable approach is to rank the models across multiple metrics and combine those ranks to find an overall winner.

Final solution

We chose a hybrid approach that empowers the user to decide which philosophy of "best" is most appropriate for their task. Instead of forcing a single definition, the final application design presents two distinct and powerful evaluation paths.

Evaluation is not a black box—it is a lens, and we let the user choose how to see.

3 Evaluation Results & Discussion

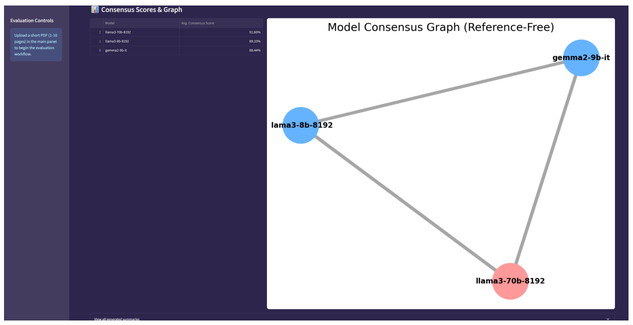
To validate the final system's capabilities, a comprehensive evaluation was conducted on a 7-page PDF document detailing a university's academic calendar. The analysis focused on three leading models available through the Groq API: `gemma2-9b-it`, `llama3-8b-8192`, and `llama3-70b-8192`. The evaluation was split into two distinct parts: a multi-faceted summarization analysis and a reference-free consensus evaluation for question-answering.

3.1 Summarization Evaluation

The models were tasked with summarizing a 7-page PDF. The evaluation was performed using both of the application's built-in methods.

This test measured how closely each model's summary agreed with the summaries of its peers. The model with the highest average similarity is considered the "consensus winner," representing the most centrally-agreed-upon output.

The results of this test were as follows:


Evaluation Controls
Upload a short PDF (1-10 pages) in the main panel to begin the evaluation workflow.
Consensus Scores & Graph
Model Consensus Graph (Reference-Free)
gemma2-9b-it
llama3-8b-8192
llama3-70b-8192
View all generated summaries

	Model	Avg. Consensus Score
2	llama3-70b-8192	91.60%
1	llama3-8b-8192	89.33%
0	gemma2-9b-it	88.44%

Figure 1: summer without reference

The consensus graph showed a tight cluster between the two `Llama3` models, indicating a similar understanding of the core concepts. The `gemma2-9b-it` model was a distinct outlier, having produced a much shorter, more concise summary that missed some of the nuance present in the others.

Next, a human-written reference summary was provided, and the models were evaluated against this "golden standard" using ROUGE scores, which measure the overlap of exact words and phrases.

The ROUGE-L (F-measure) scores were as follows:

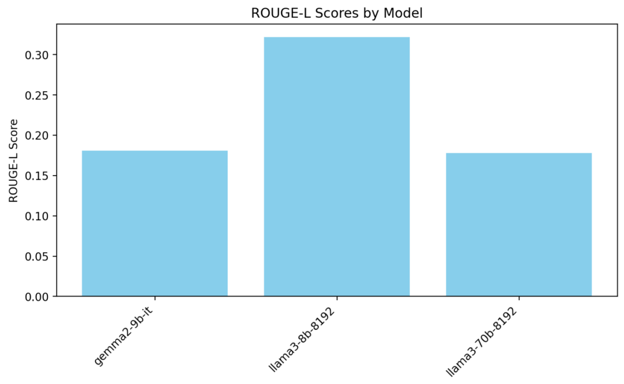


Figure 2: Rouge-L

According to this evaluation, llama3-8b-8192 is the most effective model for this specific summarization task as measured by the ROUGE-L score.

3.2 Question-Answering (QA) Evaluation

A reference-free consensus approach was used for QA evaluation. This design choice is deliberate: questions are varied, and it is hard to find a single, standard "correct" answer for any given question. The consensus method provides a robust alternative by identifying the answer that is most semantically consistent across all models.

The question asked was: "What is the attention?"

The consensus scores for the generated answers were:

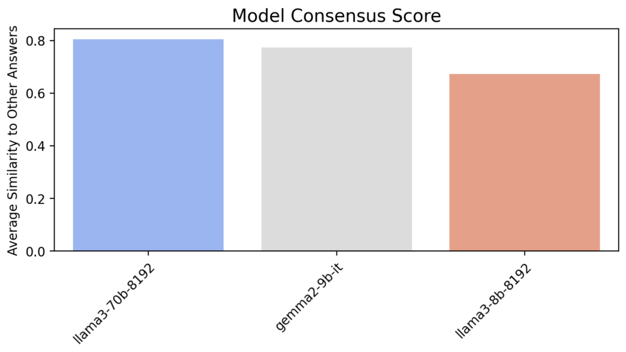


Figure 3: qa evaluation

Llama3-70b-8192 produced the best answer for this specific, fact-based question. Its response was the most direct and quoted the required test scores precisely, whereas the gemma2-9b-it models provided less direct answer.

3.3 Discussion

For the summarization task, the results from both the reference-free and reference-based methods converged, giving high confidence that `llama3-70b-8192` was the superior model. The high consensus score shows its understanding aligns with other top models, while the high ROUGE score confirms that its abstractive summary still retained the critical keywords and phrases from the reference, successfully balancing fluency with factual accuracy.

This highlights the value of the dual-evaluation approach. One of the core challenges in model evaluation is navigating the different philosophies of metrics. ROUGE emphasizes the precision rate of the generated summary against a reference, but a standard reference is often unavailable or subjective. Semantic similarity, by contrast, compares the internal consistency of meaning across models to find the most reasonable answer. By providing both, this tool empowers the user to make a more informed and defensible decision.

The QA results further demonstrate the tool's utility, revealing that the "best" model can be task-dependent. While `Llama3-70b` excelled at creating a comprehensive summary, `Gemma-2-9b` proved superior for direct, factual extraction. This is a critical insight that would be missed without a flexible, multi-modal evaluation platform.

4 Conclusion & Future Work

1. **Universal Large Document Support (for PDFs without ToCs):**
   The current Thematic Q&A workflow is powerful but depends entirely on a user-provided Table of Contents. The most critical next step is to support large documents that lack a ToC. This would be achieved through a chunking and vector search strategy:
   - **Recursive Text Chunking:** The entire document text would be extracted and then split into small, overlapping chunks of a consistent size (e.g., 500–1000 tokens). This ensures no semantic context is lost at the chunk boundaries.
   - **Vector Database Indexing:** All chunks would be converted into vector embeddings and stored in a searchable index (e.g., using FAISS). This creates a "semantic map" of the entire document.
   - **Retrieval-Augmented Generation (RAG):** When a user asks a question, the system would first perform a fast similarity search on the vector index to retrieve the top 3–5 most relevant chunks. These chunks would then be used as the focused context to generate an answer, making the entire document searchable without relying on a ToC.

2. **Automated ToC Extraction:**
   To improve the existing workflow, the manual copy-pasting of the Table of Contents should be eliminated. This can be implemented through a multi-pronged approach:
   - **Rule-Based Extraction:** Develop rules and regular expressions to identify common ToC patterns (e.g., lines ending in page numbers). PyMuPDF has built-in features that can often extract ToC data directly from PDF metadata.
   - **Model-Based Extraction:** For more complex or scanned documents, a vision-capable or text-based LLM could be prompted to read the first few pages and return a structured (e.g., JSON) version of the Table of Contents.

3. **Expanded File and Data Source Support:**
   To increase the tool's versatility, support for a wider range of input formats is necessary. This would involve adding parsers for:
   - **Documents:** Microsoft Word (`.docx`) and PowerPoint (`.pptx`).
   - **Web Content:** Allow users to submit a URL, which the application would then scrape and process.

4. **Advanced Evaluation Metrics:**
   The current evaluation suite can be enhanced with next-generation metrics that provide a more nuanced understanding of summary quality:

- **BERTScore:** An upgrade over the current similarity API, BERTScore computes similarity at the token level using contextual embeddings, better capturing paraphrasing and complex semantics.
- **LLM-as-a-Judge:**Implement a workflow where a powerful, separate LLM (like GPT-4) is prompted to act as an impartial "judge," comparing two anonymous summaries against the source and providing a qualitative reason for its choice.

5. **Cost Latency Tracking:**
   For use with commercial APIs, the application should track and display the token usage, estimated cost, and response time for each model call. This would add a critical performance and efficiency dimension to the evaluation results.

6. **Performance Optimization for Application Startup and Interaction:**
   Improving the application's responsiveness—especially during initial load—is a key priority. Users may currently experience noticeable delays when the website starts. Reducing this startup latency and optimizing interaction speed is essential for enhancing user experience. Code-level performance tuning should be treated as a high-impact task.