1. High-Level Scope & Objectives

- Implement a Retrieval-Augmented Generation (RAG) platform to handle support queries by referencing PDF documentation.

Primary Outcomes:

1. Document ingestion (PDF parsing and embedding).
2. LLM configuration for context-aware responses.
3. User-friendly interface for document uploads and query handling.
4. Conduct human evaluation of the system's performance based on key metrics:
    - **Accuracy**: Alignment with the provided documentation.
    - **Relevance**: Direct applicability to the query.
    - **Clarity**: Ease of understanding and actionability.
    - **Context Adherence**: Restriction to the uploaded documentation.

---

2. Work Breakdown

1. Document Ingestion

- Parse PDF documents.
- Generate vector embeddings.
- Store embeddings in a vector database.

2. LLM Configuration

- Integrate with LLM Models with the vector search pipeline.
- Use prompt engineering to ensure context-restricted responses.

3. Interface Design

- Develop a simple UI (web or local) for:
    - Uploading PDF documents.
    - Submitting queries.
    - Viewing LLM responses.

4. Evaluation Against Competitor

- Use identical queries on both platforms.
- Rate responses on accuracy, relevance, clarity, and adherence to provided docs.

5. Metrics & Analysis

- Human evaluation of:
    - **Accuracy** (alignment with documentation),
    - **Relevance** (directly addresses the question),
    - **Clarity** (easy to understand, actionable),
    - **Context Adherence** (sticks to the doc).

---

3. Assumptions & Constraints

- **PDF-Only Support**: During Pre-PoC, only standard text-based PDFs are in scope.
- **Document Size Limit**: Up to 5 MB per PDF.
- **Text-Focused**: Emphasis on textual data; image-heavy documents are out of scope for this phase.
- **Compute Resources**: Adequate infrastructure needed.
- **Continuous Improvement**: LLM performance will improve through iterative updates:
    - Prompt refinement.
    - Possible fine-tuning.
    - Hybrid retrieval methods.

---

4. Implementation Options

Option 1: Full-Blown Solutions with Pre-Built RAG Systems (e.g., Open WebUI, Dify)

- **Pros**:
  - Rapid setup with existing pipelines.
  - Minimal custom development (focus on configuration & testing).
- **Cons**:
  - Possible licensing/cost constraints.
  - Limited flexibility for deep customization.
  - Long-term feasibility depends on vendor lock-in and roadmap.

Links:
Open WebUI
Dify

Option 2: Develop UI, Backend, and Use Open-Source RAG Tools (e.g., Unstructured, RAGFlow, R2R, LangChain)

- **Pros**:
  - Greater control over customization.
  - Active open-source community support.
- **Cons**:
  - Accuracy and performance depend on vendor RAG.

Option 3: Full End-to-End Development from Scratch

- **Pros**:
  - Complete ownership of codebase and architecture.
  - Highly tailored to specific needs and workflows.
- **Cons**:
  - Significant development time and resources.
  - Requires specialized expertise in LLMs, RAG, and full-stack development.
  - Demands high compute resources.

---

5. Key Challenges

1. Compute Resources:

- Embedding generation and vector storage can be compute-intensive, potentially affecting cost and performance.

2. Accuracy & Continuous Improvement:

- **Prompt Engineering**: Iterative approach to refine instructions for best results.
- **Hybrid RAG**: Combining vector-based retrieval with traditional keyword searching.
- **Model Training**: If domain-specific fine-tuning is needed, it increases complexity and resource requirements.

---

6. Conclusion

Pre-PoC Focus:

The scope of this phase is to develop the basic system, evaluate its functionality, and understand its capabilities compared to an existing system. The aim is to set up the pipeline, identify challenges, and evaluate response quality and metrics like accuracy, relevance, clarity, and context adherence.

Next Phase (POC):

- Add features.
- Improve performance.
- Fine-tune accuracy.

I look forward to further discussions.