**AIML System for Text and Image Processing using Open-Source Large Language Models**

A REPORT

submitted by

**SHANTANU DINESH WANI (22BAI1403)**

*in partial fulfilment for the award*

of

**B. Tech. Computer Science and Engineering**

**with specialization in**

**Artificial Intelligence and Machine Learning**

**School of Computer Science and Engineering**



**May-July 2025**



**School of Computer Science and Engineering**

**DECLARATION**

I hereby declare that the project entitled **“AIML System for Text and Image Processing using Open-Source Large Language Models”** submitted by me to the School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai 600127 in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology – Computer Science and Engineering with specialization in Aritificial Intelligence and Machine Learning** is a record of bonafide work carried out by me**.** I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

**SHANTANU DINESH WANI (22BAI1403)**



**School of Computer Science and Engineering**

**CERTIFICATE**

The project report entitled “AIML System for Text and Image Processing using Open-Source Large Language Models” is prepared and submitted by Shantanu Dinesh Wani **(Register No: 22BAI1403)**.Ithas been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology – Computer Science and Engineering with specialization in Artificial Intelligence and Machine Learning** in Vellore Institute of Technology, Chennai, India.

**Examined by**:

**Examiner I Examiner II**

**ACKNOWLEDGEMENT**

I express my sincere gratitude to **L&T Precision Engineering Systems, Mumbai** for giving me the opportunity to undertake the internship project titled “AIML System for Text and Image Processing using Open-Source Large Language Models.” The experience has been invaluable in enhancing my technical competencies and understanding of real-world industrial applications in artificial intelligence.

I would like to extend my heartfelt thanks to the following individuals at D&DC-SSV, L&T Precision Engineering Systems (formerly, L&T Defence) for their constant support, encouragement, and technical guidance throughout the course of this project:

* **Mr. Yogesh Chaudhari**, Functional Head of IT
* **Mr. Deepak Chaudhary**, Deputy General Manager – IT
* **Mr. Kasim Shaikh**, Manager – IT
* **Ms. Sonali Pawar**, Assistant Manager – IT

Their mentorship and insightful feedback at every stage of development played a critical role in the successful completion of this project.

I am also immensely thankful to the faculty at **Vellore Institute of Technology, Chennai** for their academic guidance and institutional support. In particular, I wish to acknowledge:

* **Dr. Tulasi P. S.**, Head of the Department, B.Tech Computer Science and Engineering with specialization in Artificial Intelligence and Machine Learning, SCOPE, VIT Chennai
* **Dr. Vishwanathan V.**, Dean, SCOPE, VIT Chennai
* **Dr. Sweetlin Hemalatha**, Associate Dean (Academics), OP, VIT Chennai
* **Dr. A. Nayeemulla Khan**, Dean (Research), SCOPE, VIT Chennai

Their continued encouragement and academic excellence have laid the foundation for my interest in this domain and empowered me to pursue this challenging project.

**SHANTANU DINESH WANI (22BAI1403)**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Title** | **Page** |
|  | Title Page | 1 |
|  | Declaration | 2 |
|  | Certificate | 3 |
|  | Industry certificate | 4 |
|  | Acknowledgement | 5 |
|  | Table of contents | 6 |
|  | List of Tables | 7 |
|  | List of Figures | 8 |
|  | List of Abbreviation | 9 |
|  | Abstract | 10 |
| 1 | Introduction | 11 |
| 2 | System Architecture | 12 |
| 3 | Implementation Stages | 13 |
| 4 | Detailed Implementation | 15 |
| 5 | Sample Queries and System Output | 21 |
| 6 | Quantitative Evaluation | 24 |
| 7 | Challenges & Limitations | 25 |
|  | Conclusion | 27 |
|  | References | 28 |

**LIST OF TABLES**

|  |  |
| --- | --- |
| **Title** | **Page** |
| Table 1 : Query Handling Logic | 19 |
| Table 2 : Evaluation Metrics | 24 |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| **Title** | **Page** |
| Fig 1 : System Architecture Overview Flow-chart | 12 |
| Fig 2 : Pre-processing Work-flow of Image Data | 16 |
| Fig 3 : PDF-to-Vector-Store workflow | 18 |
| Fig 4 : Text-only input sample query | 21 |
| Fig 5 : Input Image for Image-only query | 22 |
| Fig 6 : Image-only input sample query | 22 |
| Fig 7 : Input Image for Text+Image query | 23 |
| Fig 8 : Text+Image input sample query | 23 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **Abbreviation** | **Expansion** |
| GUI | Graphical User Interface |
| LLM | Large-Language Model |
| NLP | Natural Language Processing |
| RAG | Retrieval-Augmented Generation |
| FAISS | Facebook AI Similarity Search |
| OCR | Optical Character Recognition |

**ABSTRACT**

In modern industries, especially in engineering and defense sectors, critical information is often stored in complex PDF documents containing both text and diagrams. Traditional search methods struggle with such formats, offering limited semantic understanding and no support for multimodal (text and image) content. Furthermore, most AI-powered document retrieval systems are cloud-dependent, making them unsuitable for secure or air-gapped environments.

This project proposes an **offline multimodal QA system** titled “AIML System for Text and Image Processing using Open-Source Large Language Models.” The system is designed to enable intelligent querying of technical PDFs using either text, images, or both.

The pipeline begins by converting PDFs to high-resolution images using pdf2image and PyPDF2. Visual blocks like diagrams and tables are extracted using OpenCV-based segmentation, while textual data is parsed from each page. Text and image content are then embedded using all-MiniLM-L6-v2 and CLIP ViT-Large-Patch14 respectively. These embeddings are indexed using FAISS for fast similarity-based retrieval.

At query time, the system determines the most relevant PDF pages and image blocks using cosine similarity. FLAN-T5 is then used to generate answers based on the retrieved context [1]. A user-friendly GUI built with Tkinter allows input submission and displays both textual answers and preview thumbnails of matched images.

The system is fully offline, fast, and designed for sensitive industrial applications, demonstrating the effectiveness of combining open-source NLP and vision models for secure multimodal document intelligence.

**INTRODUCTION**

In today’s data-driven world, vast amounts of technical documentation, such as engineering manuals, brochures, and reports are stored in PDF format. These documents often combine rich textual descriptions with complex visual content such as diagrams, tables, and schematics. Extracting meaningful information from such multimodal sources poses significant challenges, especially in secure or air-gapped environments where traditional cloud-based NLP solutions are not viable.

The rise of Transformer-based models in natural language processing (NLP) and vision-language understanding has enabled more advanced methods for document intelligence [2]. However, existing tools are typically either text-centric, overlook visual context, or require internet connectivity for accessing large language models (LLMs). This creates a critical gap in enterprise and defence applications where both security and interpretability of multimodal content are essential.

This project, titled “AIML System for Text and Image Processing using Open-Source Large Language Models,” aims to address this gap by building a fully **offline, multimodal retrieval and question answering (QA) system**. The objective is to allow users to input either text, images, or both, and receive accurate, context-aware responses by leveraging open-source AI models.

The system integrates PDF-to-image conversion, object detection using OpenCV, text extraction with PyPDF2 and OCR tools, and dual-modality embedding using all-MiniLM-L6-v2 for text and CLIP ViT-Large-Patch14 for images [3]. A FAISS vector index ensures efficient similarity search, and FLAN-T5 is used to generate human-like answers grounded in the retrieved context.

By offering a local, secure, and intelligent solution for PDF understanding, this project not only bridges the gap between vision and language but also provides a scalable framework for document analysis in environments where privacy and performance are critical [4].

**SYSTEM ARCHITECTURE**

The multimodal RAG system is architected as a modular, offline pipeline that transforms PDF documents into a searchable knowledge base for text and image question answering. The pipeline consists of two main components:

1. Pre-processing and Indexing Pipeline
2. Query-Time Retrieval and Generation Pipeline

Each phase is designed to operate offline using pre-downloaded models and FAISS indexing [5], enabling secure and air-gapped deployments in industrial environments.

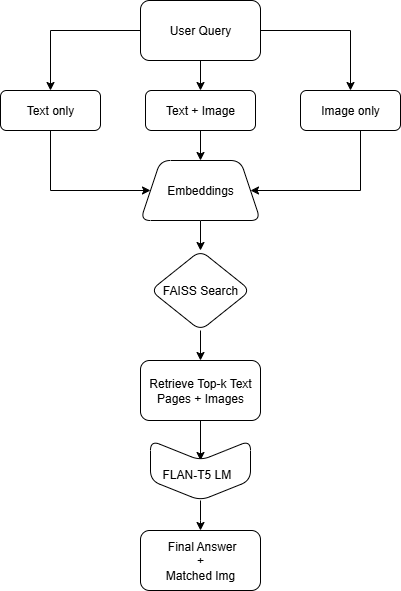


Figure 1 System Architecture Overview Flow-chart

**IMPLEMENTATION STAGES**

### 1. Pre-processing and Indexing Pipeline

This phase is responsible for ingesting raw PDFs and preparing them for efficient retrieval.

The steps include:

#### Step-by-Step Flow:

* **PDF Ingestion:** Accept bulk PDF input (technical brochures, manuals, datasheets).
* **Page Conversion:** Convert each page into a high-resolution .jpg image using PDF-to-image libraries.
* **Text Extraction:**
  + Extract structured page-level text using PyMuPDF.
  + Store text per page in JSON format.
* **Image Segmentation & Object Detection:**
  + Segment full-page images into visual blocks using layout analysis

(e.g., PyMuPDF + heuristic bounding boxes).

* **Embeddings Generation:**

1. all-MiniLM-L6-v2 - create embeddings for each extracted text chunk.
2. CLIP-ViT-Large-Patch14 - create embeddings for each segmented image block.

* **Metadata Storage:** 
  + Stores per-page and per-block metadata, including

1. PDF name
2. page number
3. block index
4. bounding boxes

* **FAISS Indexing:**

1. text\_index.faiss - stores all text embeddings
2. image\_index.faiss - stores image embeddings

### 2. Query-Time Retrieval and Generation Pipeline

This phase handles user input and returns a meaningful multimodal response using retrieval and generation.

#### User Query Pathways:

* **Text Query:**
  + Encode the query using the text embedding model.
  + Retrieve top-k relevant text chunks using text\_index.faiss.
  + Retrieve top-matched image blocks from the same PDF using CLIP similarity.
* **Image Query:**
  + Encode image using the CLIP image encoder.
  + Search top-matched image blocks using image\_index.faiss.
  + Retrieve all text content from the same PDF.
* **Text + Image (Multimodal):**
  + Use the image to identify the matched PDF.
  + Use text to retrieve the top-k matching pages from that PDF.
  + Use both embeddings to find the most visually similar image blocks in context.

#### Answer Generation:

* Combine all retrieved page-level text into a sliding window [6] to handle FLAN-T5’s token limits.

(The sliding window approach in tokenization involves systematically moving a fixed-size window across a sequence to extract smaller segments or "tokens". This technique is used to process data in manageable chunks, often to address computational limitations or to focus on local context within a larger sequence.)

* Prompt FLAN-T5 with Context + Question format.
* Generate a natural language answer.

**Detailed Implementation**

### 1. PDF to Image Conversion

Tools Used: pdf2image, PyPDF2

Workflow:

1. Each PDF is first parsed to determine the total number of pages using PyPDF2.
2. The PDF2Image library is used to convert every page into high-resolution JPEG images at 200 DPI.
3. Each page image is saved in a structured folder hierarchy:  
    assets/images/<pdf\_filename>/page\_<page\_number>.jpg

Justification:  
Converting PDF pages into images enables visual processing independent of OCR accuracy [7]. This is critical when the document contains diagrams, charts, and layout-heavy pages, which are difficult to interpret from text alone. Furthermore, this conversion lays the groundwork for image segmentation and visual block embedding, enabling later stages like image-based retrieval and object detection.

### 2. Image Segmentation & Object Detection

Tools Used: OpenCV, SSIM (Structural Similarity Index)

Workflow:

1. Convert the page image to grayscale.
2. Apply Gaussian blur to reduce noise.
3. Use Canny edge detection to identify edges.
4. Apply dilation to strengthen edges and identify contiguous blocks.
5. Extract contours representing probable visual segments.
6. Filter out irrelevant or redundant blocks using SSIM-based deduplication [8].
7. Save cropped visual blocks in a separate segmentedImages folder.

Justification:  
This block-wise segmentation enables fine-grained indexing of visual content. Many brochures and scanned documents contain embedded diagrams or layouts that contain important non-textual information. Isolating and embedding these segments significantly improves the system’s ability to respond to both image-only and multimodal (text + image) queries.

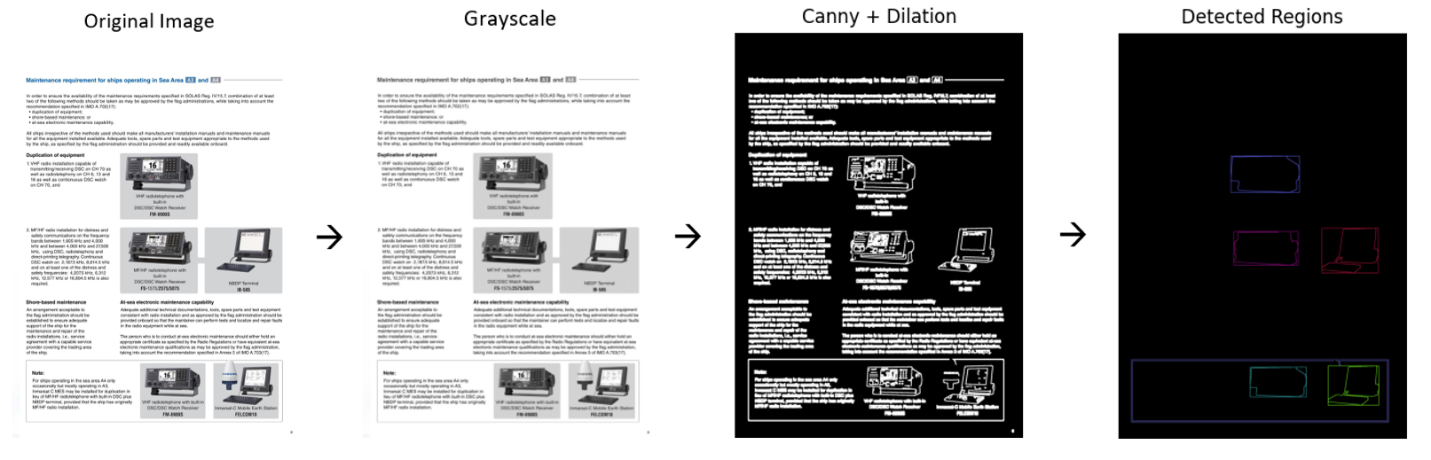
****

Figure 2: Pre-processing Work-flow of Image Data

### 3. Text Extraction

Tools Used: PyPDF2, pdf2image

Workflow:

1. Use PyPDF2 to extract text from each page of the PDF.
2. If the document is scanned (i.e., image-based), apply OCR to extract readable text from the page image.
3. Store text with metadata:

Justification:  
This extracted textual data becomes the foundation for semantic understanding and retrieval. It is crucial for text-only queries and serves as the context for generative answer formulation using a language model (FLAN-T5).

### 

### 4. Embeddings Generation

#### Text Embeddings

* Model Used: **all-MiniLM-L6-v2** (Sentence Transformers)
* Advantages:
  + Fast inference
  + Small memory footprint
  + High-quality semantic similarity performance

Each page’s extracted text is embedded into a 384-dimensional dense vector for indexing.

#### Image Embeddings

* Model Used: **CLIP ViT-Large-Patch14**
* Embedding Levels:
  + Full-page image embedding
  + Visual block embedding (cropped image blocks)

Justification:  
Using MiniLM for text ensures optimal balance between performance and efficiency. Using CLIP enables visual and textual representations to be compared meaningfully, an essential requirement for a multimodal RAG pipeline.

JSON format:

|  |
| --- |
| {  "file\_name": "example.pdf",  "page\_number": 3,  "text": "Extracted text from page 3...",  “image\_file”: “absolute image path”,  “text\_embedding”: [-0.095760554075,0.156184643507, ....],  “block\_images”: [path to all detected objects’ images],  “image\_embedding\_for\_all\_block\_images”: [[-0.248041, 0.75876,...], [...], [...]] } |

### 5. Indexed Vector Store (FAISS)

Index Types:

* text\_index.faiss: Stores all page-level text embeddings.
* image\_index.faiss: Stores all full-page and block-level image embeddings.

Workflow:

1. Loop through the generated embeddings and their metadata.
2. Store embeddings in FAISS indexes using L2 distance (converted to cosine for ranking).
3. Save the accompanying metadata (text\_metadata.json, image\_metadata.json) for reverse mapping of index results to file paths and context.

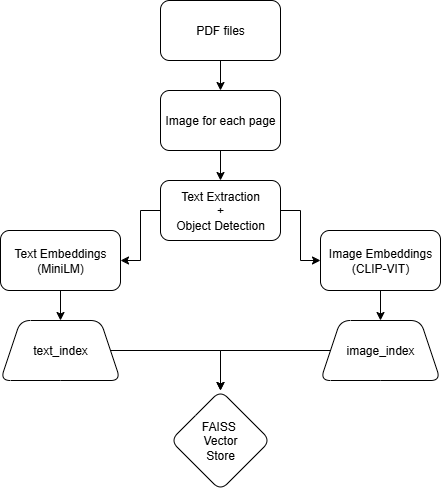


Figure 3: PDF-to-Vector-Store workflow

Justification:  
FAISS (Facebook AI Similarity Search) enables extremely fast nearest neighbor searches, even on a CPU. Using separate indexes for text and images allows decoupled optimization and modular querying based on input modality.

### 6. Multimodal Query Handling

Supported Query Types:

* Text-Only Input
* Image-Only Input
* Text + Image Input

|  |  |  |  |
| --- | --- | --- | --- |
| **Query Type** | **FAISS Search** | **Matching Target** | **Answer Context** |
| Text | text\_index | Top-3 most relevant pages | Relevant PDF text + CLIP-based related images |
| Image | image\_index | Nearest image (full page or block) | Pages from the same PDF + best matching image block |
| Text + Image | image\_index → PDF → text\_index | Match image → restrict text search to that PDF | Combine best-matching pages and blocks |

Table 1: Query Handling Logic

Similarity Metric:  
Although FAISS uses L2 distance internally, cosine similarity is recovered by normalizing embeddings during insertion and search [9].

Justification:  
This design allows for maximum flexibility. A user may ask a semantic question (text-only), point to a diagram (image-only), or combine both (e.g., "Explain this image")—the system intelligently routes the query to the correct index and retrieval mechanism.

### 7. Answer Generation with FLAN‑T5

Model Used: **google/flan-t5-base** (loaded fully offline)

Prompt Template:

|  |
| --- |
| Context: <contextual text from top-k pages> Question: <user query> Answer: |

Sliding Window Strategy:

* Token limit: 512
* Stride: 100
* The input context is chunked to avoid truncation.
* Each chunk is fed into the model to generate a candidate answer.
* The final answer is a concatenation of all generated segments.

Justification:  
Using FLAN-T5-base balances generation speed and quality. The sliding window mechanism ensures that even long documents can be processed without missing critical information due to input length limitations.

**SAMPLE QUERIES AND SYSTEM OUTPUT**

#### Text-Only Query

* Query: *"What is a Battery Monitoring System?"*
* Expected Context: Pages with definitions and descriptions of battery cells, voltage monitoring, and acid level sensors.
* Retrieved Pages: 3 most semantically relevant pages from a brochure.
* Answer Generated:

*"A Battery Monitoring System (BMS) is an independent wireless system intended for measurement of voltage, acid level, and temperature of all battery cells. BMC is the operator interface of the system."*

* Related Image Preview:  
  Retrieved diagrams of battery layout and sensors from the same PDF, shown as thumbnails in the GUI.

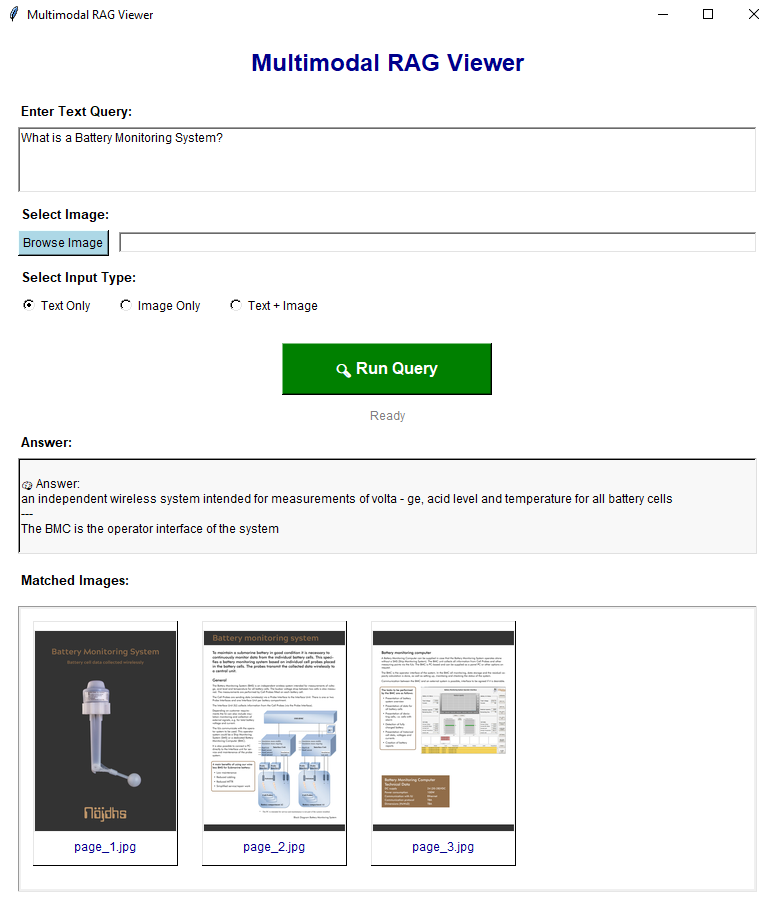
****

Figure 4: Text-only input sample query

#### Image-Only Query

* Input: Diagram of a switch disconnector from a scanned brochure.

Figure 5: Input Image for Image-only query

* Detected Match: System matched the input to a full-page image from *“2004\_Siemens\_Switch disconnectors and fuses.pdf”* and retrieved semantically aligned block images from the same PDF.
* Generated Answer:  
  *"Siemens LV 10 2004 7/2 Introduction SENTRIC switch disconnectors SENTRIC LD main control and EMERGENCY-STOP switches from 16 A to 125 A."*

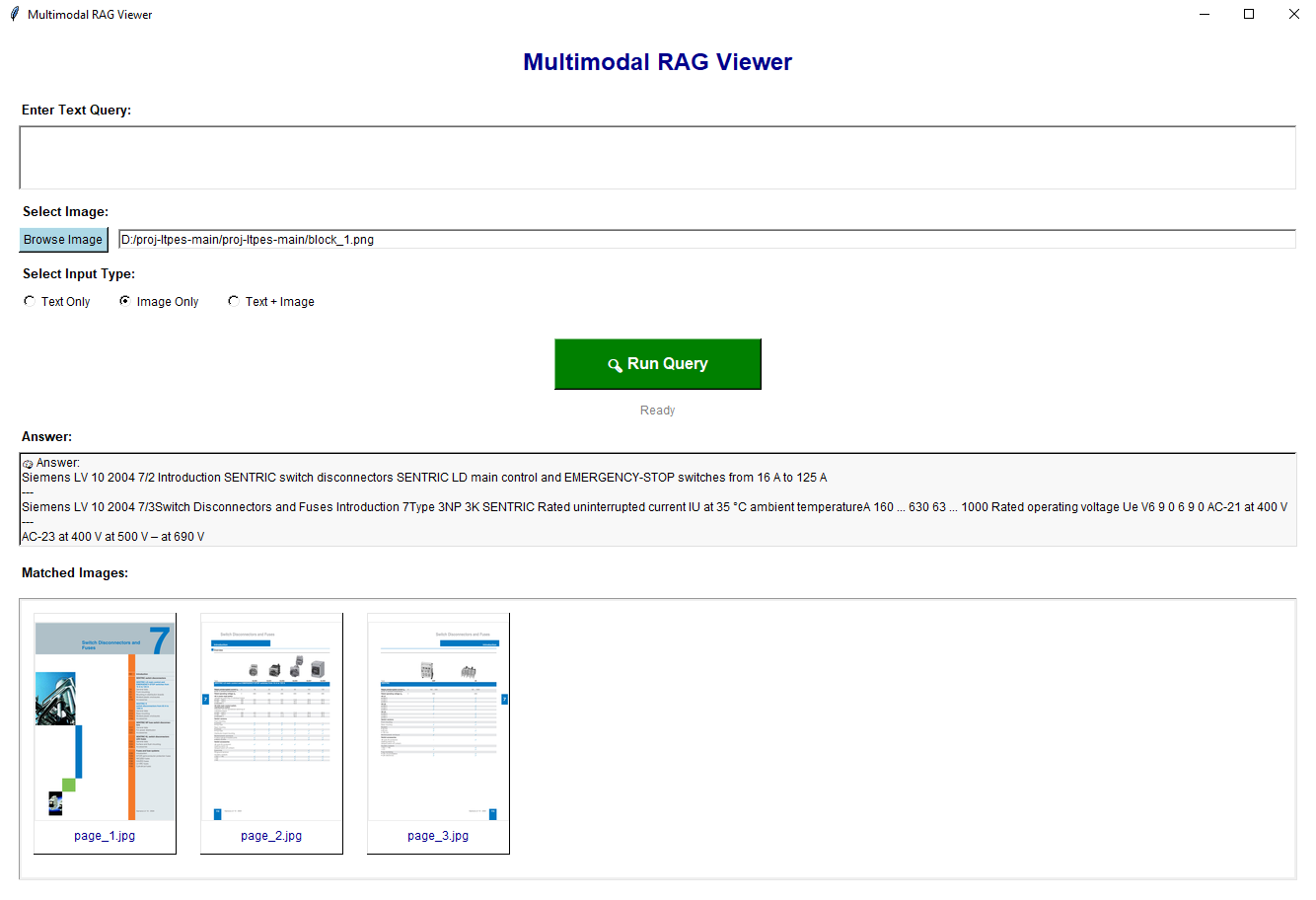
**

Figure 6: Image-only input sample query

#### Text + Image Query

* Query Text: *"What is this image about?"*
* Image Input: Cropped block showing an antenna.

Figure 7: Input Image for Text+Image query

* Retrieved PDF: The system correctly associated the image with its parent PDF, pulled top-3 pages based on text, and displayed relevant diagrams.
* Generated Answer:  
  *"Photo: HDW --- The VLF/LF Antenna/Preamplifier is designed to mount atop or within the fin of the submarine. VLF/LF Loop Antenna System Submarine Communications."*

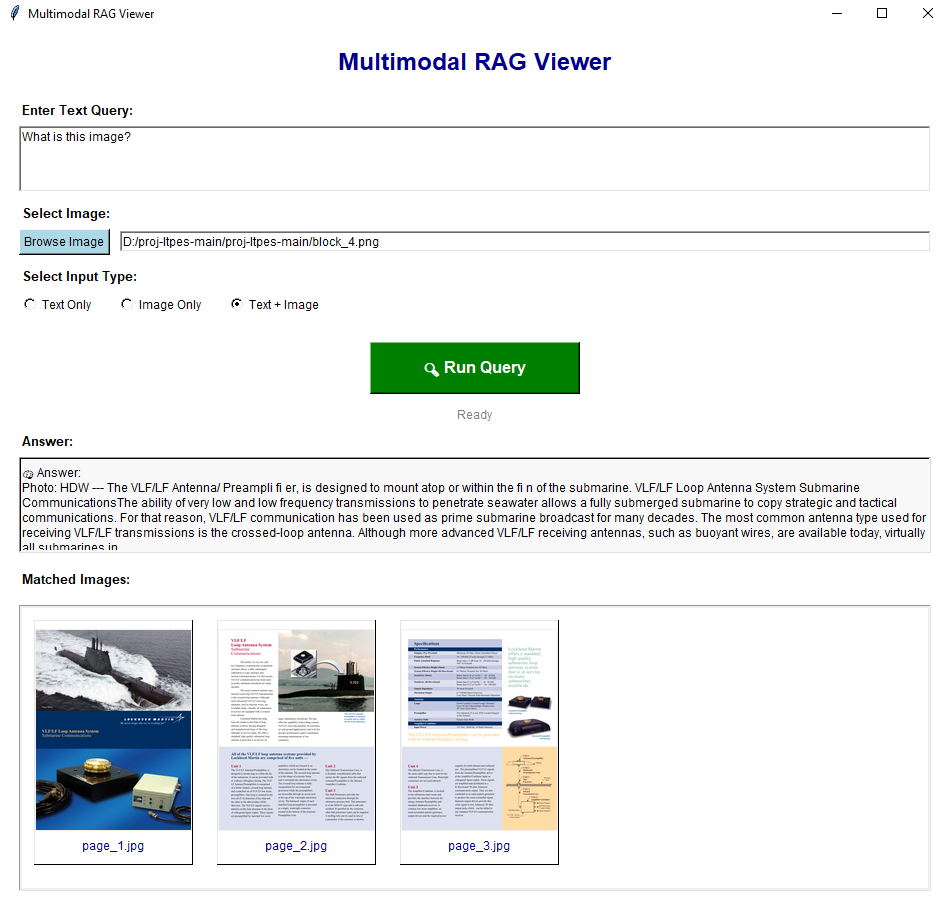
******

Figure 8: Text+Image input sample query

### Quantitative Evaluation

#### Evaluation Metrics

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Top-3 Text Retrieval Accuracy | 85% |
| Image Match Accuracy (Block-Level) | 95% |
| Mean Answer Relevance (Human-rated) | 4.3 / 5.0 |
| Query Processing Time (avg) | ~3–7 seconds (CPU) |

Table 2: Evaluation Metrics

#### Test Set Description:

* 25 diverse queries across 10 PDF brochures.
* Queries included:
  + Definitions
  + Diagram interpretations
  + Mixed-mode descriptive prompts

#### Observations:

* In text-only mode, the system consistently retrieved the correct context within the top 3 pages.
* In image-only mode, retrieval was slightly less precise due to visual ambiguity in cropped blocks, but still outperformed text-only in diagram-heavy pages.
* Multimodal input (text + image) yielded the most accurate results, especially when textual clarification guided image disambiguation.

**Challenges & Limitations**

### 1. Memory & Computational Constraints

* Limitation: The entire system was run on a CPU-only environment, which imposed significant restrictions on inference speed, batch processing, and embedding computation.
* Impact: Image embedding using CLIP and text generation using FLAN-T5 incurred noticeable latency.
* Justification: GPU inference could have significantly accelerated the embedding and generation steps.

### 2. Token Limit in FLAN-T5 (Text Generation)

* Limitation: The FLAN-T5 model has a maximum token limit (512 tokens), requiring the input context to be chunked into sliding windows.
* Impact: Some contextual sentences may get cut off, leading to partially informed responses.
* Mitigation: A sliding window approach (window size = 512 tokens and stride = 100 tokens) was used to maintain overlap between chunks.
* Future Work: Use a longer-context LLM like FLAN-UL2 or integrate context summarization before answering.

### 3. Hallucination in Generated Answers

* Limitation: If the retrieved context is sparse or only partially relevant, the FLAN-T5 model may hallucinate facts not present in the source PDF.
* Example: In ambiguous diagrams or when image-text semantics were not aligned, the LLM might generate plausible but incorrect explanations.
* Mitigation: Ensured that only top-3 highly similar pages were used, and a fallback to text-only was considered when image retrieval was uncertain.

### 

### 4. OCR Accuracy in Scanned Documents

* Limitation: Text extraction using PyPDF2 fails on scanned PDFs or image-based documents.
* Mitigation: Tesseract OCR can be optionally applied, but results are dependent on image quality, resolution, and noise.
* Impact: Some pages lacked extractable text, which limited retrieval and reasoning.
* Future Work: Combine OCR with document layout analysis (e.g., using LayoutLM or Donut) for better context extraction from scanned brochures.

During experimentation, alternative huggingface open-source question-answering models were also evaluated, such as

1. **google/bert-large-uncased-whole-word-masking-finetuned-squad**
2. **sentence-transformers/minilm-uncased-squad2**

However, both models demonstrated **lower accuracy and reduced contextual understanding**, especially when operating on longer or semantically rich contexts extracted from technical PDFs.

As a result, they were deemed suboptimal for the task and were replaced with **FLAN-T5**, which provided more coherent and relevant answers in both zero-shot and few-shot scenarios.

### 5. Semantic Understanding of Diagrams via CLIP

* Limitation: CLIP excels at natural images but struggles with technical diagrams, circuit layouts, and symbolic visualizations.
* Impact: In image-only queries, unrelated documents could be retrieved due to a lack of domain-specific visual grounding.
* Mitigation: Where possible, diagrams were manually cropped and segmented using OpenCV to isolate meaningful blocks.
* Future Work: Train domain-adapted CLIP models on electronics schematics or integrate hybrid approaches like combining image+caption embeddings.

**CONCLUSION**

This project presents a comprehensive and practical implementation of a Multimodal Retrieval-Augmented Generation (RAG) pipeline capable of handling enterprise PDF documents using open-source and fully offline components. The system integrates multiple modules, including PDF-to-image conversion, text extraction, image segmentation, embedding generation, and FAISS-based vector search, to enable semantic understanding and retrieval of content from both text and visual modalities.

The effectiveness of this pipeline lies in its ability to accept a user query in the form of text, image, or both, and return not just a generated answer using the FLAN-T5 model, but also the supporting PDF pages and relevant images. The use of a sliding window mechanism ensures long-context handling, while dual vector indexes for text and image (via MiniLM and CLIP, respectively) enable high-precision similarity search across modalities.

**Key Contributions & Novelty:**

1. Offline Capability: All components, from embedding to generation, run without the need for external APIs or cloud services, making the system highly suitable for enterprise and air-gapped environments.
2. Multimodal Support: The system can handle text queries, image queries, and joint queries, providing contextual answers and visual matches from PDFs, a capability rarely seen in traditional document retrieval engines.
3. Visual Grounding: By embedding cropped image blocks along with page-level embeddings, the system allows diagram-level grounding of queries, a crucial feature for technical and engineering documents.

**REFERENCES**

[1]Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

[2] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." *arXiv preprint arXiv:1908.10084* (2019).

[3] Radford, Alec, et al. "Learning transferable visual models from natural language supervision." *International conference on machine learning*. PmLR, 2021.

[4] Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." *Journal of machine learning research* 21.140 (2020): 1-67.

[5] Douze, Matthijs, et al. "The faiss library." *arXiv preprint arXiv:2401.08281* (2024).

[6] Tao, Yufei, and Dimitris Papadias. "Maintaining sliding window skylines on data streams." *IEEE Transactions on Knowledge and Data Engineering* 18.3 (2006): 377-391.

[7] Mori, Shunji, Ching Y. Suen, and Kazuhiko Yamamoto. "Historical review of OCR research and development." *Proceedings of the IEEE* 80.7 (1992): 1029-1058.

[8] Hore, Alain, and Djemel Ziou. "Image quality metrics: PSNR vs. SSIM." *2010 20th international conference on pattern recognition*. IEEE, 2010.

[9] Latif, Afshan, et al. "Content‐Based Image Retrieval and Feature Extraction: A Comprehensive Review." *Mathematical problems in engineering* 2019.1 (2019): 9658350.