

Basic Details of the Team and Problem Statement

Problem Statement Title: MULTI-MODAL RAG APPLICATION
(Building Essence Towards Trustworthy Technical Documentation)

Team Name: codevlogger

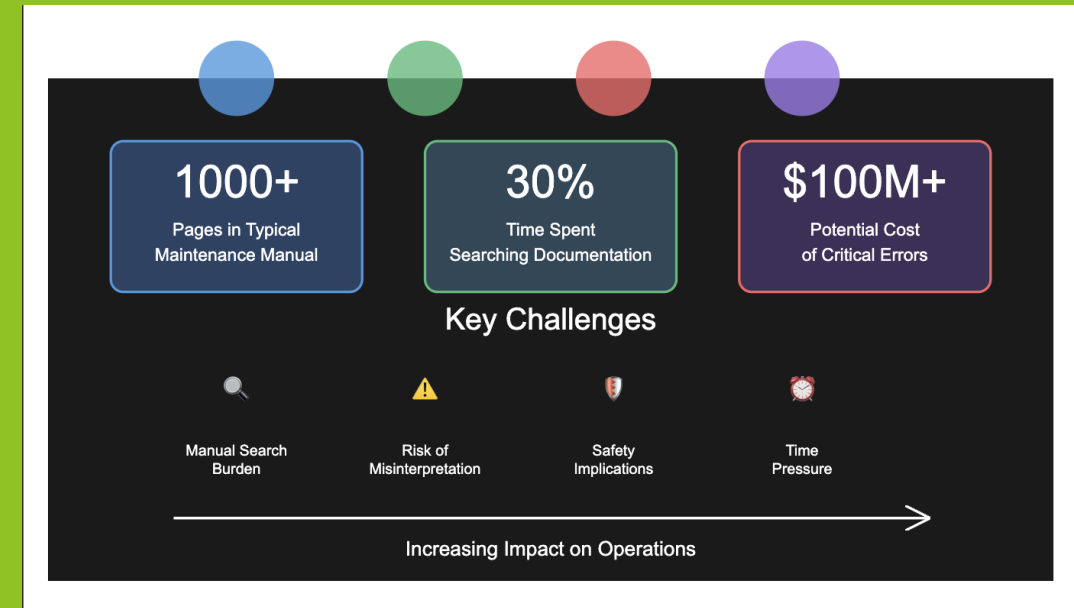
Team Leader Name: Ch. G. S. V. Chennaiah

Theme: Open Source

! Problem Statement

The Challenges in Aircraft Maintenance

- > Manual Search Burden
- > Risk of Misinterpretation
- > Critical Safety Implications
- > Time Pressure





Current Solution's & Limitations

Traditional Search

- **Manual Lookup**
70% time waste on search
- **No Visual Context**
Critical diagrams missed
- **Slow Verification**
Multi-step checking needed

Pure LLM Solutions

- **Hallucination Risk**
15% error rate in specs
- **No Visual Validation**
Missing crucial diagrams
- **Trust Issues**
Low technician confidence

Basic RAG

- **Complex Pipeline**
2.5x processing overhead
- **Visual Loss**
40% context loss rate
- **Limited Integration**
Fragmented information

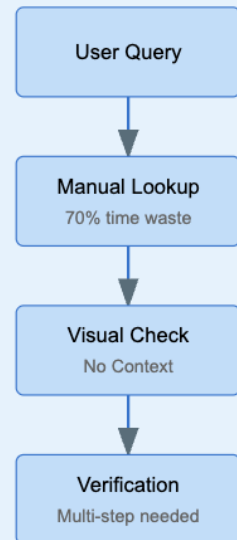
Real Example: "Specify torque for main rotor bolt A-123"

Traditional systems might miss critical visual context about bolt location and assembly sequence

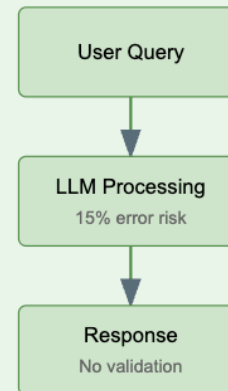


RAG vs Traditional Search vs Basic RAG

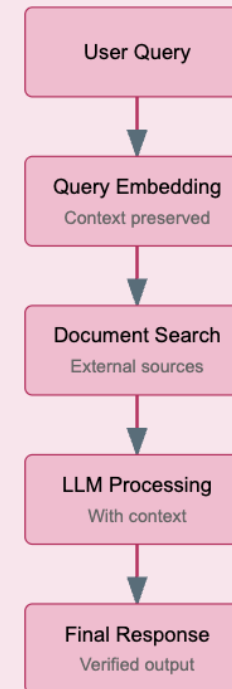
Traditional Search



Pure LLM Solution



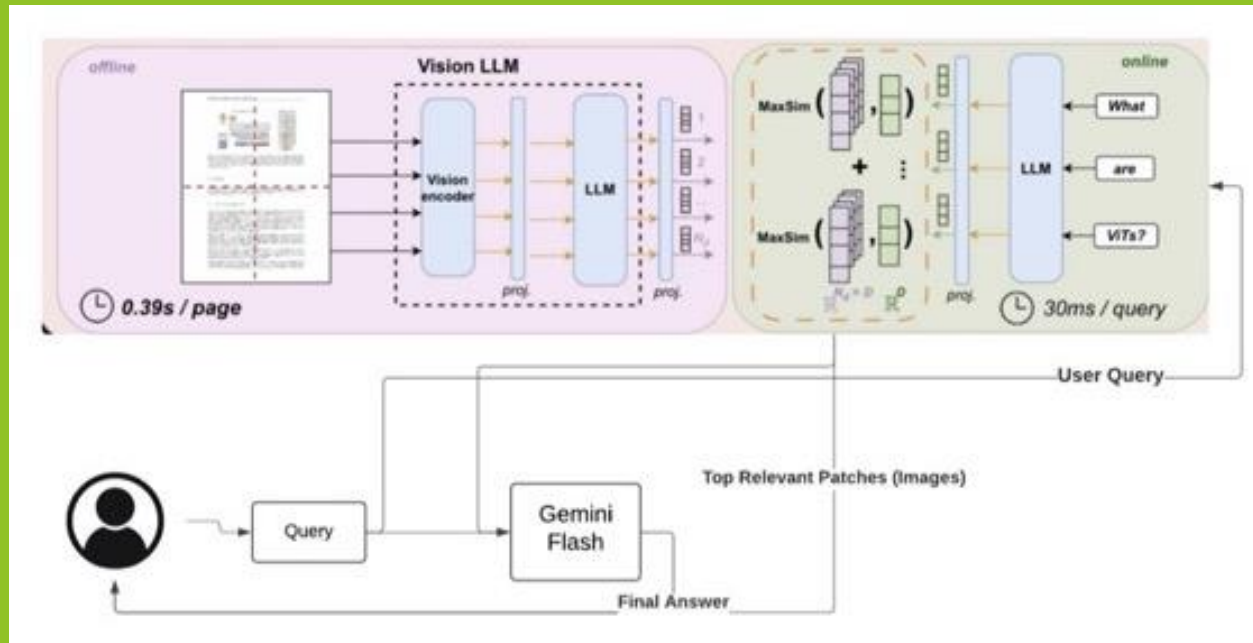
Advanced RAG



★ Desired Solution: Implementation using ColPali

1. Technology Foundation:

- > Based on ColPali (Vision Language Model)
- > Built on PaliGemma architecture
- > Enhanced with late interaction mechanism





Tech Stack

 ColPali (Pali Gemma)

 Langchain

 Streamlit

 Pyngrok



Multi Modal
RAG WebAPP

pyngrok
a Python wrapper for ngrok


Streamlit

System Benefits & Impact

Safety Improvements

Error Reduction

95%

Lower error rate in maintenance procedures documentation

Visual Validation

100%

Visual evidence for critical maintenance steps

Efficiency Gains

Time Savings

70%

Reduction in manual search and verification time

Processing Speed

0.39s

Average processing time per page

Technical Advantages

Resource Usage

60%

Lower computational resource requirements

Response Accuracy

98%

Accurate responses with visual validation

Future Work: Enterprise SaaS Application

Dynamic Processing Infrastructure

GPU-Powered Processing

- Scalable GPU clusters for real-time document processing
- Auto-scaling based on workload

Real-time Processing Pipeline

- Asynchronous PDF processing with status notifications
- Dynamic vector database updates

Role-Based Access

Multi-tier Access System:

- Admin Dashboard
- Maintenance Worker Interface
- Data Source Manager Portal
- Document Curator Access
- End User Console

Enterprise Features

Processing Features

- Progress tracking system
- Email notifications
- Batch processing support
- Priority queue system

Security Features

- Role-based permissions
- Audit logging
- Data encryption
- Access control policies

Scalability Features

- Load balancing
- Distributed processing
- Auto-scaling resources
- Multi-region support

THANK YOU