AIPHAISTOS Project - Video Script 1: Demo Presentation

Hi, my name is Sanjay Gokhale. I'm currently working on a major AI project under the guidance of my professor, Dr. Stefan Heinemann. This project is titled AIPHAISTOS, and it aims to solve a very critical problem: understanding and extracting meaningful answers from highly technical, architectural, and contractual documents using AI.

Let me show you the demo of my current backend system.

This is built using FastAPI and is designed to work through a Swagger UI. Users can upload a variety of documents—PDFs, DOCX files, images, even CAD files—and our system automatically extracts the text, chunks it, performs OCR if needed, stores vector embeddings using FAISS, and creates a hybrid BM25 index.

When I ask a question related to the uploaded document, our backend retrieves the most relevant chunks using FAISS and BM25, reranks them with a local reranker, and then feeds the top results into a local LLM like Zephyr using Ollama. The answer is generated contextually based on the document content.

The system is smart enough to choose the right model: for blueprints it uses a specialized model, for general PDFs it uses a compact Zephyr LLM, and we have fallback mechanisms using OpenAI if needed. It also logs every query and the matched chunks for debugging.

This is the foundation of our AI assistant, which mimics ChatGPT-like behavior but uses your documents as the knowledge base. Now let's move to the second part to explain how it works in code.

AIPHAISTOS Project - Video Script 2: Backend Overview and Limitations

Hi, this is Sanjay Gokhale again. In this video, I will walk you through the technical backend of our AIPHAISTOS project in Visual Studio Code.

All logic is implemented using Python in FastAPI. We have modularized processors for different file types—PDF, DOCX, TXT, DWG, and images. We also use Tesseract and YOLO for OCR and layout parsing in technical drawings. Every upload goes through virus scanning and safe processing before chunking and embedding.

The query pipeline does semantic and lexical search together (FAISS + BM25), reranks chunks using an LLM, and dynamically selects a model—Zephyr, Mistral, Yi, TinyLlama, or specialized blueprint QA models. This helps us make the answers more intelligent and grounded.

But we are facing a bottleneck. To make this assistant truly accurate, especially for complex documents like blueprints and contracts, we need domain-specific fine-tuning. This requires:

1. A large volume of real-world technical data.
2. High computational power—preferably GPUs—for LoRA fine-tuning.

Unfortunately, due to limited GPU access and data availability, we're restricted in how smart the answers can be right now. But our system is ready for fine-tuning. Once we gain access to more data and GPU resources, we can significantly improve the performance.

This is our AI journey with AIPHAISTOS—building a real assistant that understands real

documents. Thank you, and special thanks to my professor, Dr. Stefan Heinemann, for supporting this project.