**INTEL® UNNATI INDUSTRIAL TRAINING PROGRAM**

**Project Report**

Problem Statement 4: AI-Powered Interactive Learning Assistant for Classrooms

*Submitted By*

**Sai Rithvik Nama**

B.Tech in Information Technology

Manipal Institute of Technology

rithvik.mitmpl2022@learner.manipal.edu

*Under the guidance of*

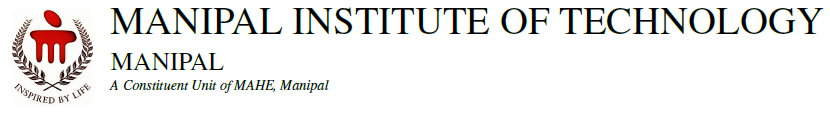
**Mrs. Vibha Prabhu**

Assistant Professor - Senior Scale

Dept. of Information and Communication Technology

Manipal Institute of Technology

vibha.prabhu@manipal.edu



## **Introduction**

As educational content moves to digital platforms, students must interact with diverse formats like text, handwritten notes, diagrams, audio lectures, and slides. Traditional AI tools for question answering and summarization are often limited to cloud-based systems, single-modality inputs, and privacy-invasive setups. There is a growing demand for a responsive, multimodal, privacy-preserving assistant that operates locally, even in low-resource environments.

MnemoCore addresses this need with an end-to-end educational platform. It integrates an OpenVINO-accelerated local LLM, a LangChain-driven retrieval system, and multimodal processing for voice, images, and PDFs. The platform also features a structured course management interface, enabling students to access content, track chapter progress, and generate AI-powered summaries and flashcards. MnemoCore provides a seamless, context-aware self-learning experience without relying on external APIs or GPUs.

## **Background**

Artificial Intelligence is reshaping education by powering smarter, more adaptive, and personalized learning systems. Traditional education often relies on one-size-fits-all content delivery, overlooking students’ unique backgrounds, learning speeds, and preferences for materials like texts, diagrams, audio lectures, or handwritten notes.

AI transforms education by enabling dynamic, interactive, and student-focused engagement, moving beyond passive content consumption. It supports smart classrooms and adaptive learning environments through:

1. Personalised education: AI adapts educational materials to match a student’s progress, knowledge gaps, and preferences, offering a more effective alternative to fixed curricula.

2. Multimodal Content Integration: AI can work with a variety of formats like text, voice and images to generate summaries, explanations, or contextual responses.

3. 24/7 Support: AI serves as a 24/7 assistant, providing clarifications, definitions, or examples tailored to a student’s specific study materials, beyond traditional classroom hours.

5. Optimised Learning: Using techniques such as spaced repetition, AI schedules revisions, tracks progress, and improves long-term retention efficiently.

6. Offline Accessibility: Locally deployed AI models enable students in low-resource or low-connectivity settings to access intelligent support without relying on internet or high-end hardware.

These capabilities form the foundation of smart classrooms, where AI complements teachers by providing real-time, tailored, and multimodal assistance, scaling personalized education across diverse contexts.

MnemoCore reflects these principles as an offline, AI-powered learning assistant. It processes varied educational inputs, guides students through structured content, and delivers context-aware, personalized learning support.

## **Objectives**

The MnemoCore project was designed with the following technical objectives:

1. To deploy a fully functional large language model (LLM) locally on CPU using Intel's OpenVINO runtime with minimal latency and resource footprint.
2. To design a multimodal input pipeline capable of extracting and integrating context from:  
   1. Text documents (e.g., PDFs, lecture slides).
   2. Scanned images or handwritten notes (via OCR).
   3. Voice input (via microphone or MP3 uploads).
3. To implement Retrieval-Augmented Generation (RAG) using LangChain and a local vector store, allowing question-answering grounded in user-uploaded content.
4. To support conditional prompt augmentation, ensuring context is injected only when it is semantically relevant to the query.
5. To build an educational content companion UI capable of managing multiple courses and chapters, rendering embedded PDFs, and enabling AI-based summarization and flashcard generation.
6. To provide an integrated revision planner using spaced repetition visualizations.

# **Methodology and Implementation**

The MnemoCore platform was designed with the goal of supporting fully local, multimodal educational assistance on consumer-grade hardware. The implementation is divided into two core modules: (1) the Multimodal Chat Assistant, which handles all forms of user queries and input processing, and (2) the AI Course Companion, which manages structured learning content, revision planning, and interactive summarization tools. This section details the methodology behind each module, the technologies and models used, and the system-level optimizations applied to ensure reliable and performant local operation.

## **Model Selection and Optimization with OpenVINO**

The backbone of the assistant is the microsoft/Phi-3-mini-128k-instruct language model, selected for its compact size (∼1.8B parameters), instruction tuning, and competitive performance in CPU-only environments. To optimize inference efficiency, the model was quantized to INT8 and converted to the OpenVINO Intermediate Representation (IR) format using the optimum-intel toolchain. This involved exporting to ONNX, applying post-training quantization, and converting to OpenVINO IR. The model is executed on CPU via a LangChain-compatible wrapper with no GPU dependency.

## **Contextual Augmentation via RAG (Retrieval-Augmented Generation)**

MnemoCore uses LangChain’s Chroma vector store for RAG. Uploaded PDFs are chunked and embedded using sentence-transformer models and indexed for similarity search. Conditional context injection is used, where retrieved chunks are only included in prompts if their similarity exceeds a configurable threshold. This avoids unnecessary context pollution.

## **Multimodal Input Preprocessing**

MnemoCore supports PDF, image, and audio inputs. PDFs are parsed with PyPDFLoader; images are processed via Tesseract OCR; audio is transcribed using faster-whisper. Extracted text from all modalities is indexed and used for RAG.

## **Streamlit-Based User Interface**

The application uses Streamlit for its frontend. The chat assistant (app.py) supports text and multimodal queries, session-based persistence, and file previews. The course companion (streamlit\_lms.py) enables chapter navigation, embedded PDF viewing, summary and flashcard generation, and spaced repetition-based revision planning.

## **Flashcard and Summary Generation**

Flashcards are generated from PDFs using a structured prompt and regex parsing of LLM output. Summaries are generated using few-shot instruction-tuned prompts. Both outputs are cached and stored per chapter.

## **Session Management and Persistence**

Sessions are saved using a timestamped session ID. Each session has an isolated vector store, a history log, and links to associated file uploads, ensuring reproducibility and modular context handling.

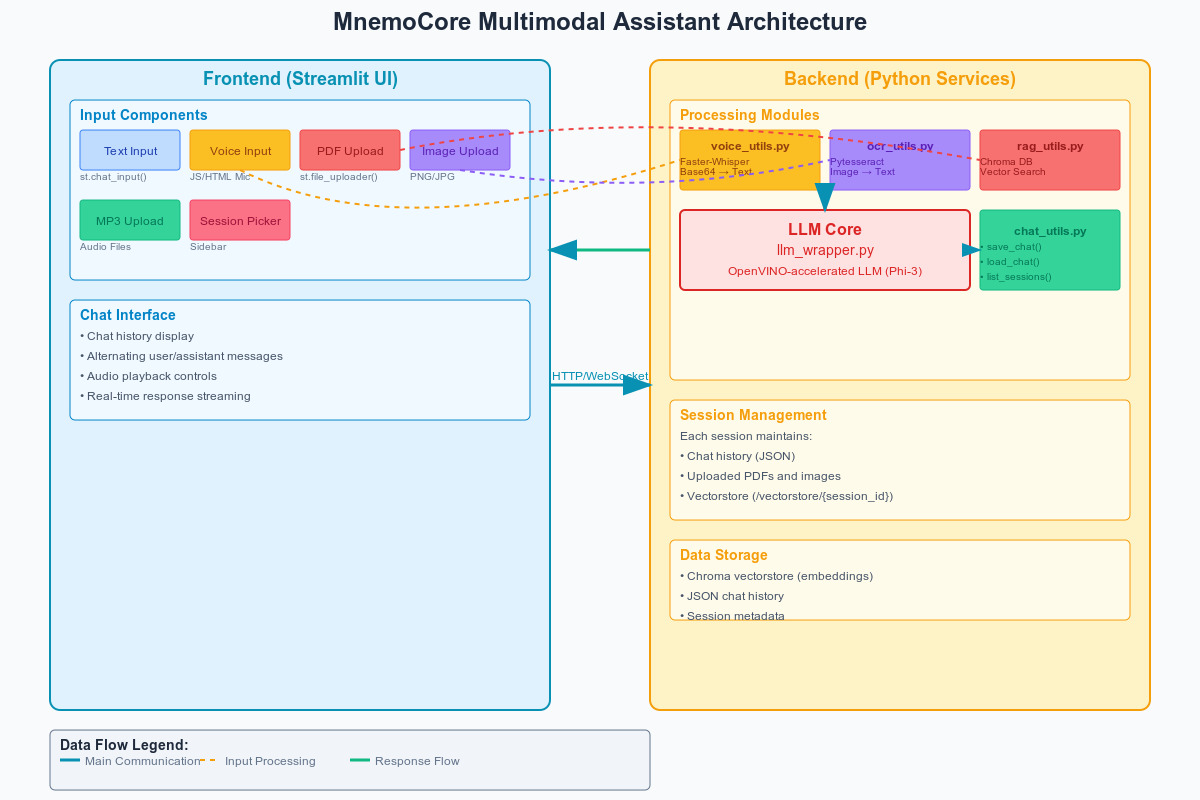
## 

## **Performance Considerations**

Optimizations include OpenVINO INT8 quantization, chunk caching, on-demand loading, and context relevance filtering.

For the purpose of comparison, the Course Companion was built using the Phi-3-Mini model with Ollama.

# **Architecture**



# 

# 

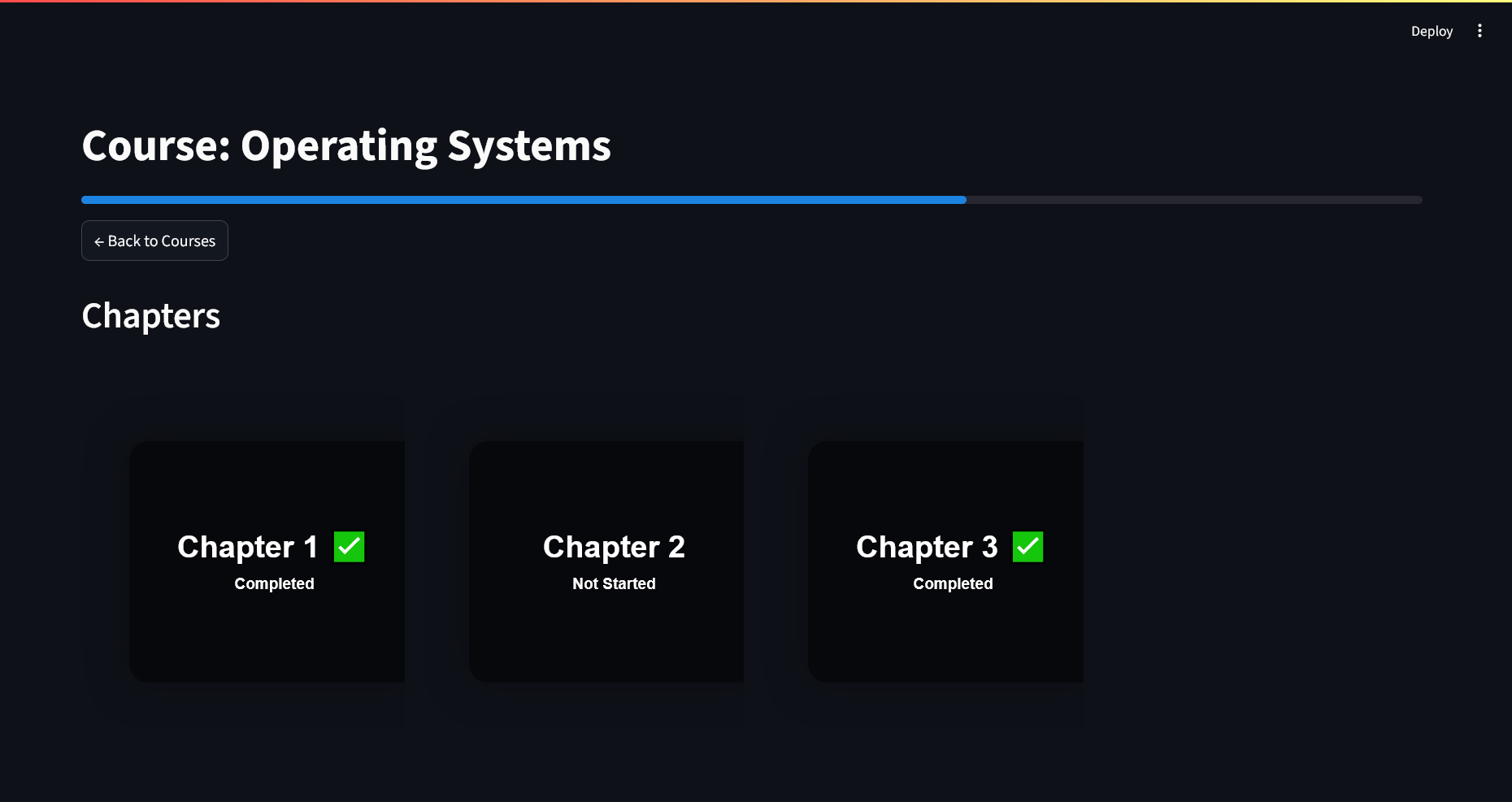
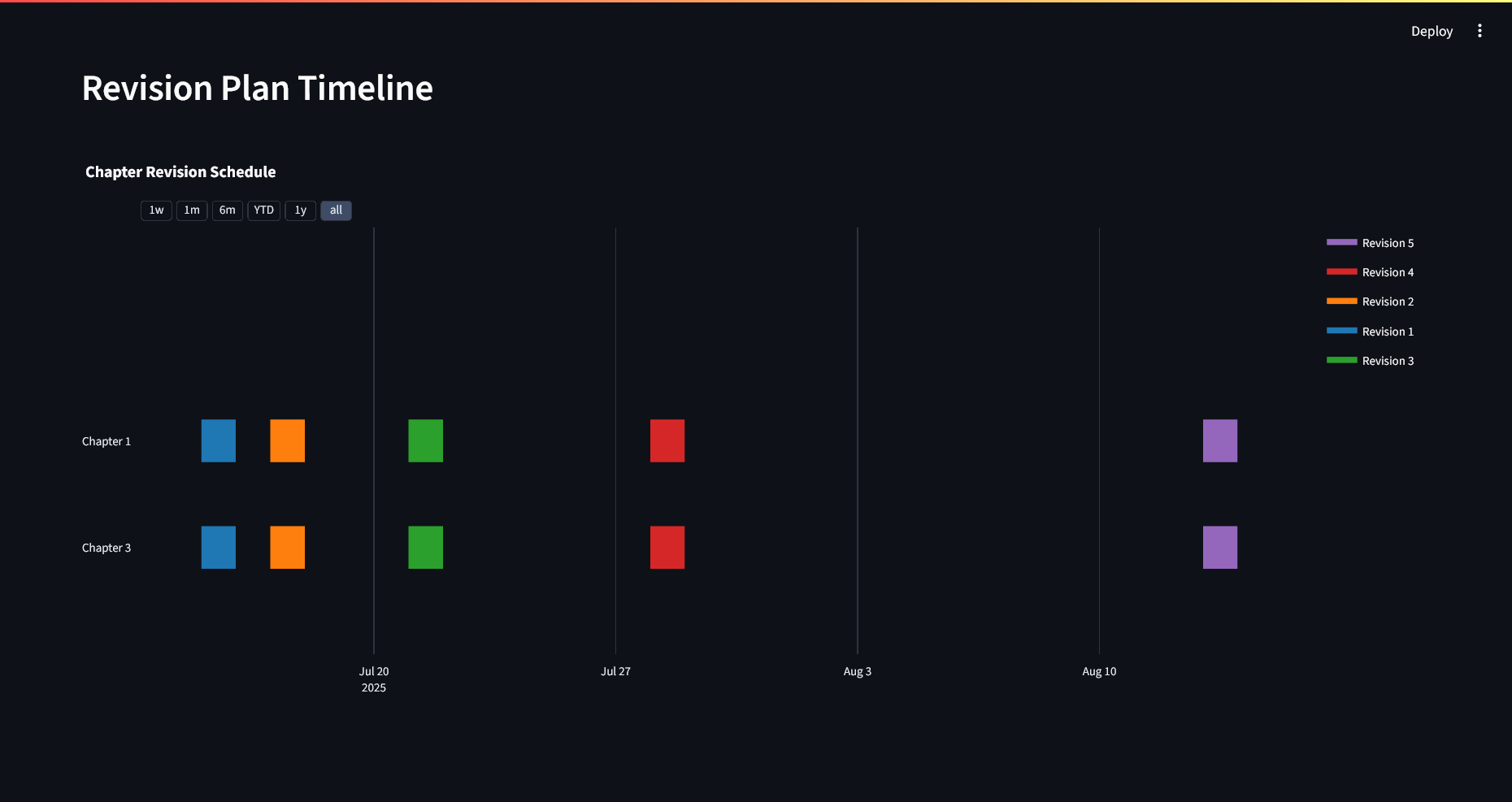
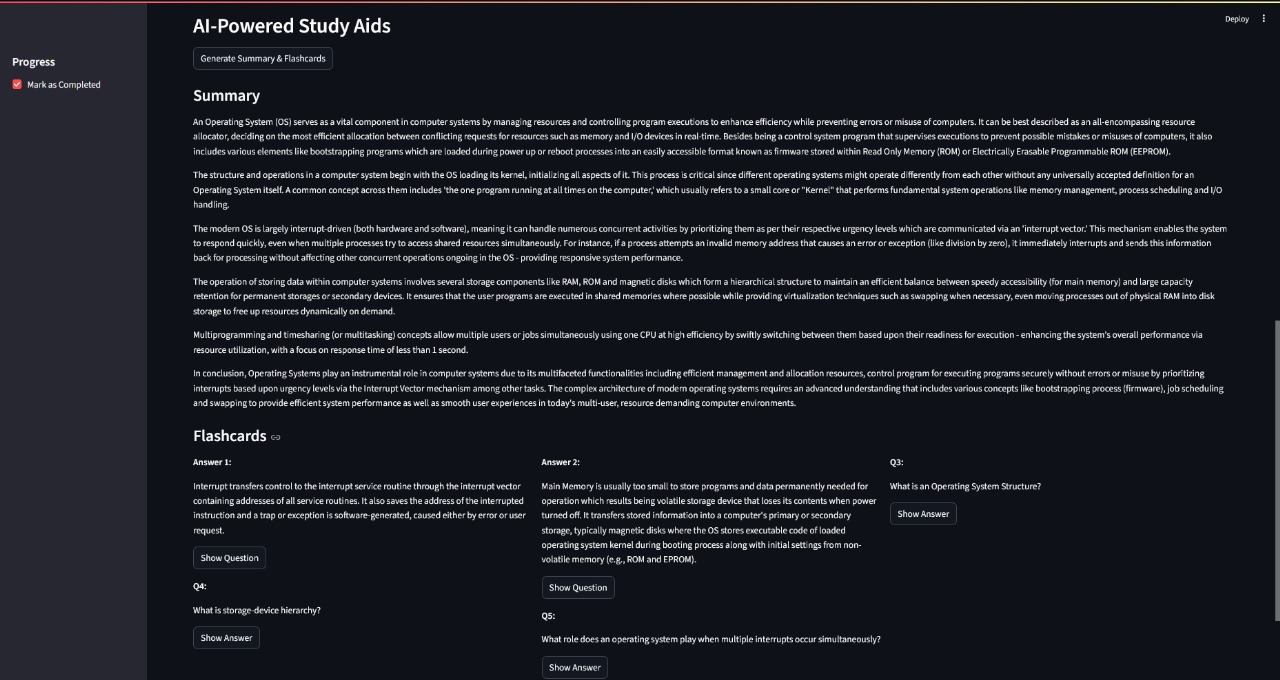
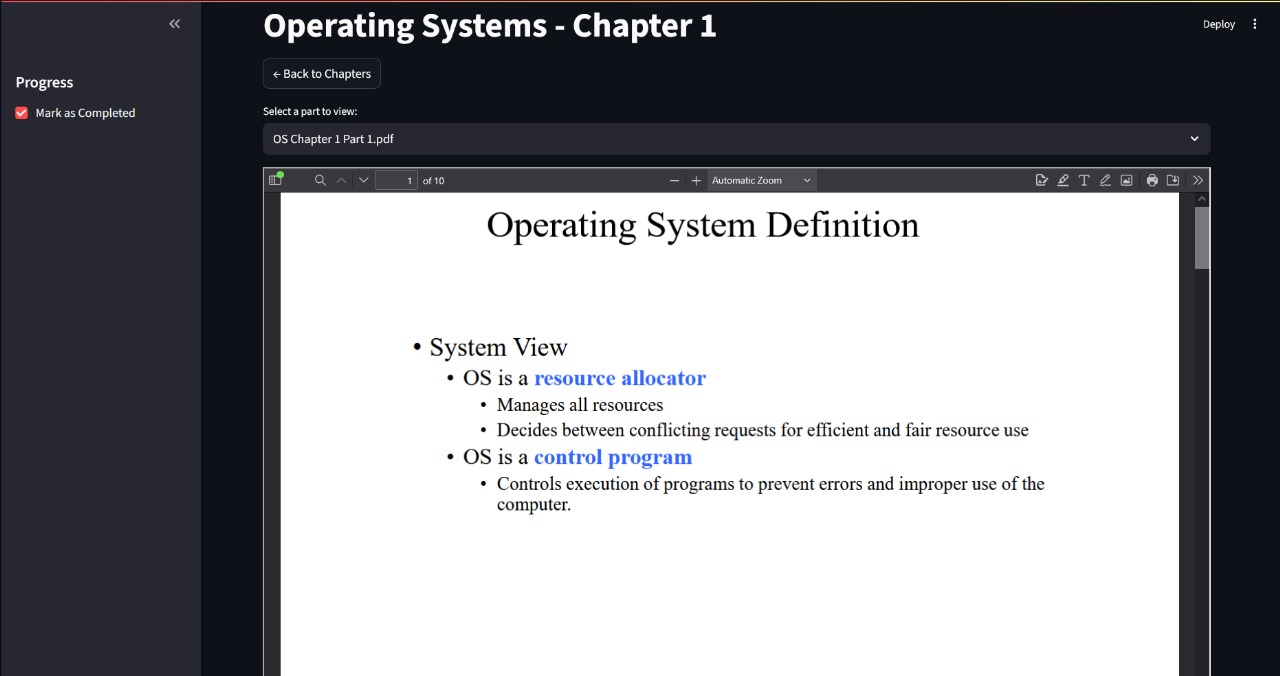
# **Results**

## **Demo Videos**

1. MnemoCore Course Companion: [**https://youtu.be/sjx1kb9Jvbs**](https://youtu.be/sjx1kb9Jvbs)
2. MnemoCore Assistant Demo 1: [**https://youtu.be/3Gyc1SHJx\_w**](https://youtu.be/3Gyc1SHJx_w)
3. MnemoCore Assistant Demo 2: [**https://youtu.be/lPkAdUeulIs**](https://youtu.be/lPkAdUeulIs)
4. MnemoCore Assistant Demo 3: [**https://youtu.be/84NxpD5FUWw**](https://youtu.be/84NxpD5FUWw)

## **Screenshots**

### **MnemoCore Course Companion**



### **MnemoCore Assistant**

### 

## **Benchmarks**

A python script was used to test latency and memory usage for the ollama model and the openVINO model used in the Course Companion and Multimodal Assistant respectively. The results are presented below. Testing was performed on a system with an 8th generation Intel i7-8550u and 12gb of RAM:

| Metric | OLlama (Phi-3-Mini) | OpenVINO (Phi-3-Mini) |
| --- | --- | --- |
| Latency (ms) | 104139.05 | 63421.85 |
| Memory Usage (MB) | 6281.64 | 5205.34 |

Benchmarking shows that the OpenVINO-optimized Phi-3-Mini model outperformed the Ollama deployment in terms of inference speed, achieving approximately 39% lower latency (63s vs 104s) on CPU. OpenVINO also showed better memory efficiency, consuming about 17% less RAM (5.2 GB vs 6.3 GB).

The results show that openVINO is a more suitable option especially in use-cases where access to higher-performance or better-specced systems is less likely, such as in classrooms.

# 

# 

# 

# 

# 

# 

# **Future Improvements**

## **Short-Term Objectives**

### **Multilingual Query Support**

1. Integrate Whisper large model (CPU-optimized) for Indian language transcription.
2. Add IndicNLP or AI4Bharat embeddings for context-aware RAG in Hindi and other regional languages.

### **Improved PDF Chunking + Metadata-Aware Retrieval**

1. Implement semantic chunking using sentence boundaries and headings.
2. Tag document chunks with metadata (e.g., subject, topic) for scoped search and filtered retrieval.

### **Model Caching & Session Performance Optimization**

Add persistent session-level caching for:

1. Retrieved chunks (via LangChain's retriever cache)
2. Tokenized inputs for repeated prompts
3. Reduce redundant inference overhead in long sessions.

## **Longer-Term Implementation**

### **Facial Emotion-Based Engagement Detection**

Integrate OpenCV + FER or DeepFace models to detect:

1. Disengagement
2. Confusion or boredom
3. Use feedback loop to alter explanation style or slow down delivery.

### **Adaptive Teaching Assistant Loop**

1. Track query success rate, user feedback score, and query categories.
2. Implement RLHF-like feedback system to prioritize:
   1. Easier analogies for weak topics
   2. More examples for high-error categories

### **Curriculum Alignment**

1. Create modular curriculum packs per grade level (K-12 or university courses).
2. Include topic classification models for automatic alignment with uploaded PDFs.

### **Teacher Analytics Dashboard**

1. Backend:
   1. Store chat metadata, session usage logs, and topic frequency in a lightweight SQLite or NoSQL DB.
2. Frontend:
   1. Display heatmaps for common student doubts.
   2. Material upload interface with tagging and bulk ingestion

### 

### **Edge-Optimized Multimodal Pipeline**

1. Quantize OCR + captioning + LLM models using NNCF or OpenVINO INT8 pipelines.
2. Integrate Intel NPU (if available) and test power-efficiency vs CPU fallback.

# **Conclusion**

MnemoCore represents a practical and efficient approach to deploying intelligent learning systems locally, without reliance on high-end hardware or internet access. By combining OpenVINO-accelerated inference with robust input pipelines and a structured user interface, the system offers multimodal educational assistance that is both fast and extensible.

Its technical design emphasizes modularity, allowing future additions such as regional language support, group collaboration, and adaptive testing. The integration of RAG, OCR, Whisper, and OpenVINO into a single cohesive workflow showcases the power of local inference when combined with modern retrieval and interface technologies.

MnemoCore demonstrates that real-world, student-centric AI tools can be built with open tools, run efficiently on edge devices, and adapt to the complex and varied formats in which knowledge is consumed today.

# **Contributions**

This project was built in its entirety by Sai Rithvik Nama, with guidance and help from Mrs. Vibha Prabhu and Mr. Debdyut Hajra, the academic and industrial mentors for this project respectively.

# 

# **Acknowledgements**

1. Mrs. Vibha Prabhu, my mentor for this project from MIT.
2. Intel OpenVino *(https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html)*
3. Hugging Face *(https://huggingface.co/)*
4. Tesseract OCR *(https://github.com/tesseract-ocr/tesseract)*
5. LangChain *(https://github.com/langchain-ai/langchain)*