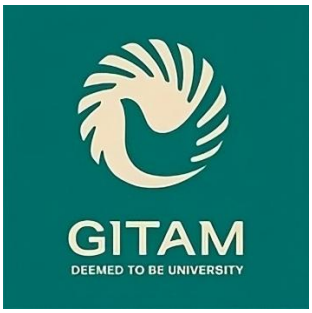
Intel Unnati Industrial Training Program

Problem Statement - 4: AI – Powered Interactive Learning Assistant for classrooms.



Department: Computer Science and Engineering

Team - Spartans:
Sathwik Yellapragada (VU22CSEN0101112)

Karri Sai Chetan (VU22CSEN0100827)

Bhargav Padala (VU22CSEN0100298)

Mentoring Faculty:
Dr. K E Naresh Kumar

SAGE - Smart AI Guide for Education

Project Report

Links

- **GitHub Repository**: https://github.com/sathwik-y/AI-Tutor-Intel
- **Demo Video**: **https://drive.google.com/file/d/1FrbZW1xjyyt4xpzJavFZ2tZgrUFx_PMq/view?usp=sharing**

Abstract

SAGE (Smart AI Guide for Education) is a full-stack AI-driven intelligent tutoring system designed to provide an immersive, multimodal, and personalized educational experience. By combining powerful AI models with modern web technologies, SAGE offers features such as real-time voice interaction, document/image analysis, context-aware Q&A, and automated attendance tracking.

---

1. Introduction

The rapid development of artificial intelligence has transformed the landscape of education. SAGE leverages this evolution by providing an interactive, AI-first platform that adapts to individual learners' needs. Built as part of the Intel Unnati Industrial Training Program, the system integrates React, FastAPI, and OpenVINO-optimized Qwen 2.5 models to deliver highly efficient and intelligent learning support.

---

2. Objectives

- Build an interactive, multimodal tutoring assistant.
- Integrate advanced language models for contextual learning.
- Optimize AI inference on standard hardware using Intel OpenVINO.
- Provide seamless frontend-backend integration for real-time use.

---

3. System Architecture

3.1 Overview

SAGE consists of three primary layers:

- **Frontend**: React 18 application with Tailwind CSS and Vite for an interactive UI.
- **Backend**: FastAPI REST service with integrated AI services.
- **Model Optimization**: OpenVINO INT8 quantization for optimized model inference.

3.2 Folder Structure

```
SAGE/
├── sage-frontend/
├── sage-backend/
├── OpenVino/
└── setup.py
```

---

4. Features

4.1 Conversational AI

- Built using the Qwen 2.5 language model optimized via Intel OpenVINO.
- Allows learners to engage in contextual, educational conversations.
- Integrated with a Retrieval-Augmented Generation (RAG) system for document-based responses.

4.2 Voice Communication

- Utilizes Faster Whisper for low-latency speech-to-text transcription.
- Implements a TTS pipeline using edge-optimized synthesizers for real-time vocal feedback.
- Enables accessibility and enhances user engagement via voice-first interfaces.

4.3 Document Analysis

- PDF files are parsed using PyPDF2 and NLP summarization pipelines.
- Document content is embedded using Sentence Transformers for semantic understanding.
- Supports follow-up Q&A based on uploaded materials.

4.4 Image Interpretation

- OpenCV handles basic image processing for head count tracking.
- Integrates image-to-text OCR via Tesseract to read educational material from photos.
- Uses Blip for understanding Image context

4.5 RAG (Retrieval-Augmented Generation)

- Embeds and indexes uploaded document data using FAISS.
- Context from documents is retrieved at query time and passed to the Qwen model.
- Supports grounded, contextual, and accurate answer generation.

5. Technology Stack

5.1 Frontend

- **React 18**: For component-based, high-performance rendering.
- **Vite**: Ensures fast build times and hot-reloading during development.
- **Tailwind CSS**: Used for utility-first, responsive styling.
- **Additional Tools**:
    - **React Router**: Client-side routing.
    - **Axios**: API interaction.
    - **Framer Motion**: Animations and transitions.
    - **Lucide & Tabler Icons**: Visual iconography.

5.2 Backend

- **FastAPI**: High-speed Python web framework built on ASGI.
- **Uvicorn**: ASGI server for asynchronous processing.
- **Core Libraries**:
    - **PyTorch**: Model inference.
    - **Transformers**: Language model wrappers.
    - **FAISS**: Vector store for semantic document search.
    - **OpenCV & MediaPipe**: Face detection and image processing.
    - **Faster Whisper**: Optimized ASR engine.
    - **PyPDF2 & Tesseract**: PDF and image document parsing.

5.3 Model Optimization

- **Intel OpenVINO**: Converts models to optimized Intermediate Representation (IR).
- **Optimum Intel**: A Hugging Face utility to export models in OpenVINO-compatible format.
- **Model Used**: Qwen 2.5 7B-Instruct with INT8 quantization for reduced memory usage and improved inference speed.

---

6. Implementation Details

6.1 Backend Setup

- Python 3.8+
- Install dependencies:

```
pip install -r sage-backend/requirements.txt
```

- Run backend server:

  ```
  python sage-backend/run.py
  ```

6.2 Frontend Setup

- Node.js 18+
- Install dependencies:

  ```
  cd sage-frontend && npm install
  ```

- Launch development server:

  ```
  npm run dev
  ```

6.3 FFmpeg and Tesseract Setup

- Install and configure paths in backend services for:
  - **FFmpeg** (for audio conversion)
  - **Tesseract OCR** (for image-to-text recognition)

6.4 Model Directory Structure

- Place the Qwen model in:

  ```
  sage-backend/models/qwen-2.5-optimized-int8/
  ```

  Ensure all Hugging Face files are downloaded into this directory.

6.5 setup.py

- A custom `setup.py` file in the root directory automates full environment setup:
  - Verifies system prerequisites
  - Installs frontend/backend dependencies
  - Launches both frontend and backend services
  - Checks model and tool availability

---

7. Role-Based Access

SAGE implements a **role-based interface** to tailor functionality based on user type. The two main user roles **Teacher** and **Student** access the system through distinct portals within the same frontend application.

Note: The login and signup are mock functionalities as there is no production.

7.1 Student Portal

- Interact with the AI assistant via voice or text

- Upload and query PDF documents as part of contribution
- Receive contextual responses based on document content
- Access past learning history and previous queries

7.2 Teacher Portal

Inherits all features of the Student Portal and additionally includes:

- **Automated Head-Count System**:
    - Real-time detection using webcam
    - Student count and presence logging
    - Option to export attendance data
- **Analytics Dashboard**:
    - Visual breakdown of system usage
    - Attendance records and class interaction stats
- **Knowledge Base Management**:
    - Upload and organize multiple PDFs
    - Update or delete outdated documents

Both portals share the same design language and user experience, offering a cohesive and immersive interaction tailored to the needs of each role.

---

8. Core AI Components

- **Text Generator**: Qwen 2.5 + OpenVINO + FAISS (RAG-based document-aware inference).
- **Audio Services**: Faster Whisper (STT), pyttsx3 or edge-optimized synthesizer (TTS).
- **Vision Module**: OpenCV + MediaPipe for face tracking, OCR via Tesseract.

---

9. API Endpoints

Endpoint	Description
`POST /api/text/generate`	Generate contextual text response
`POST /api/audio/transcribe`	Convert speech to text
`POST /api/tts/generate`	Convert text to speech
`POST /api/image/analyze`	Process image for OCR or attendance
`POST /api/upload/process`	Analyze and summarize PDF
`POST /api/attendance/detect`	Face detection + attendance marking

10. Performance Evaluation

10.1 Optimization Metrics

- **Model Size Reduction**: ~53.25%
- **Inference Latency Improvement**: ~72.52%
- **OpenVINO Runtime**: Efficient on standard CPUs

---

11. Use Cases

- **Tutoring**: AI-assisted, multimodal learning sessions.
- **Homework Help**: Contextual document Q&A.
- **Attendance Logging**: Automated student presence via camera.
- **Accessible Learning**: Voice-based input and feedback.

---

12. Conclusion

SAGE demonstrates the immense potential of AI-powered education tools. Its seamless integration of multimodal AI, real-time interaction, and optimized backend performance makes it a powerful learning companion, especially on hardware-constrained systems.

---

13. Team Member Contributions

Sathwik: Conversational AI & Knowledge Systems

- Developed chat interface, knowledge base browser, and document upload features.
- Integrated Qwen 2.5 model and implemented RAG system with FAISS.
- Managed PDF parsing, vector embedding, and query contextualization.
- Added backend features for AI context filtering and response post-processing.
- Key Contribution: Document-aware response system using OpenVINO-optimized language model.

Chetan: Voice & Model Optimization Systems

- Created voice interaction pipeline, including STT and TTS services.
- Implemented real-time WebSocket communication and audio controls.
- Optimized the Qwen model using OpenVINO with INT8 quantization.
- Key Contribution: 72% latency reduction via model compression and inference tuning.

Bhargav: Vision & Analytics Systems

- Built the automated attendance feature using OpenCV and MediaPipe.
- Integrated image-to-text analysis (OCR) and BLIP-based vision processing.
- Created the analytics dashboard for visualizing usage and attendance.
- Key Contribution: Seamless visual pipeline with camera-based student detection.

---

14. Collaborative Responsibilities

Shared Contributions Across All Members:

- **System Integration**: Endpoint validation, integration testing, error handling
- **Model Optimization**: OpenVINO conversion, benchmarking, inference tuning
- **UI/UX Development**: Responsive layout, animations, cross-browser support
- **Deployment & QA**: Setup scripts, documentation, security checks, testing

---

15. References

- OpenVINO Toolkit
- Optimum Intel
- Qwen 2.5 Model
- FastAPI
- React