INTERNSHIP REPORT

On

AI-Powered Interactive Learning Assistant for Classrooms

By

I. **Ragipindi Vishnu Vardhan Reddy - BU22CSEN0101462**
II. **B V Sainath Reddy - BU22CSEN0101457**
III. **A V Chandrakanth Reddy - BU22CSEN0101499**

INTEL (Intel Unnati Industrial Training 2025 - Slot 2)

(Duration: 19/05/2024 to 05/07/2024)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Gandhi Institute of Technology and Management

(DEEMED TO BE A UNIVERSITY)

BENGALURU, KARNATAKA, INDIA

SESSION:2022-2026

Problem statement

AI-Powered Interactive Learning Assistant for Classrooms

Objective: Build a Multimodal AI assistant for classrooms to dynamically answer queries using text, voice, and visuals while improving student engagement with personalized responses.

Prerequisites:

Familiarity with natural language processing (NLP) and multimodal AI concepts.

Knowledge of speech-to-text frameworks and computer vision techniques.

Programming skills in Python, with experience in libraries like Hugging Face Transformers and OpenCV.

Problem Description:

Modern classrooms lack real-time, interactive tools to address diverse student needs and keep them engaged. The objective is to create a multimodal AI assistant that:

Accepts and processes text, voice, and visual queries from students in real-time.

Provides contextual responses, including textual explanations, charts, and visual aids.

Detects disengagement or confusion using facial expression analysis and suggests interventions.

Expected Outcomes:

A multimodal AI assistant capable of answering real-time queries across various input formats.

Integration of visual aids (e.g., diagrams, charts) for better understanding.

A feature to monitor student engagement and adapt teaching methods dynamically.

Challenges Involved:

Combining multimodal inputs (text, voice, visuals) for consistent, context-aware responses. Ensuring low-latency processing to maintain real-time interactions. Handling diverse accents, noisy environments, and variations in facial expressions.

Contents

Introduction

The gap between teachers and students in terms of personalized attention is a major challenge in modern classrooms. Many students often hesitate to ask doubts, or teachers may not have enough time to answer every query in large classes. This is where AI can step in as a powerful learning assistant.

This project, titled “Educational AI Assistant,” aims to solve this issue by allowing students to interact with a digital helper. Students can ask questions by typing, speaking, or uploading images. The assistant understands these inputs using AI and provides contextual responses.

We use modern tools like:

- React.js to create a simple website interface (frontend)
- Flask (Python) to create the server-side logic (backend)
- OpenVINO to make AI responses faster and efficient

Even if you don’t have a powerful computer, this assistant will still work smoothly. It’s designed for accessibility and scalability.

Abstract

The Educational AI Assistant is a smart software system designed to help both students and teachers by acting like a digital learning partner. It can understand what students type, what they say, and even what they show using images (like pages from a book or diagrams). The system uses AI technologies like Natural Language Processing, Speech Recognition, and Image Text Extraction to understand queries and provide helpful answers.

To make the AI models work faster and better, we use OpenVINO, a toolkit by Intel that boosts the performance of AI tasks, even on computers without high-end graphics cards. This makes the assistant suitable for classroom settings, especially in places with limited resources. Overall, the project makes learning more interactive, accessible, and personalized.

Overview

This report outlines all the required dependencies for setting up the Educational AI Assistant project, which includes:

1.1 Backend – Flask API with OpenVINO and Phi-2

- The backend is built using Flask, a simple Python web framework.
- It uses a Phi-2 language model, which is optimized with OpenVINO to make it faster.
- This model understands questions and gives smart, accurate answers.
- It also connects with tools like OCR (for reading text from images) and Speech Recognition (for voice input).

1.2 Frontend – React.js User Interface

- The frontend is made with React.js, which gives a smooth and fast user experience.
- It allows users to:
    - Type questions,
    - Upload images (like a book page),
    - Speak questions using a microphone.
- The answers are shown in a simple chat format, like ChatGPT.

1.3 Additional Tools Used

Tesseract OCR	Reads text from uploaded images
Google Speech-to-Text	Converts spoken questions to text
ChromaDB / FAISS	Finds related content in documents
OpenVINO Toolkit	Speeds up the AI model on Intel devices

Objectives

1	Build an easy-to-use AI assistant for educational settings
2	Integrate text, voice, and image processing
3	Use Flask as a lightweight backend for API services
4	Optimize model inference using OpenVINO Toolkit
5	Support low-cost systems (non-GPU machines)
6	Provide a base for future additions like emotion detection

Core Dependencies

Python Packages (Backend)

Package	Version	Purpose
flask	2.3.2	Web framework for API
flask-cors	3.0.10	Cross-Origin Resource Sharing (CORS) support
transformers	4.48.3	HuggingFace NLP models (Phi-2)
optimum-intel	1.22.0	OpenVINO optimizations for transformers
openvino	2023.0.1	Intel’s inference toolkit
torch	2.0.1	PyTorch for model inference
onnx	1.18.0	ONNX model conversion
pytesseract	0.3.10	OCR for image text extraction

Pillow	10.0.0	Image processing for OCR
SpeechRecognition	3.10.0	Audio transcription
numpy	1.24.3	Numerical operations
protobuf	6.31.1	Serialization for model weights

System Dependencies

Dependency	Purpose
Tesseract OCR	Required for pytesseract (Install via Tesseract-OCR)
Google Speech-to-Text API	Used by SpeechRecognition

Installation Steps

A. Python Environment Setup

1. **Create a virtual environment**:

(cmd)

python -m venv classroom_ai

classroom_ai\scripts\activate

2. **Install Python packages**:

(cmd)

pip install flask flask-cors transformers==4.48.3 optimum-intel==1.22.0 openvino openvino-dev torch onnx pytesseract pillow SpeechRecognition numpy protobuf

B. System Dependencies

1. **Install Tesseract OCR**:

- **Windows**: Download from UB Mannheim

2. **Set Tesseract Path** :

(python)

```
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\TesseractOCR\tesseract.exe'  # Windows
```

System Architecture

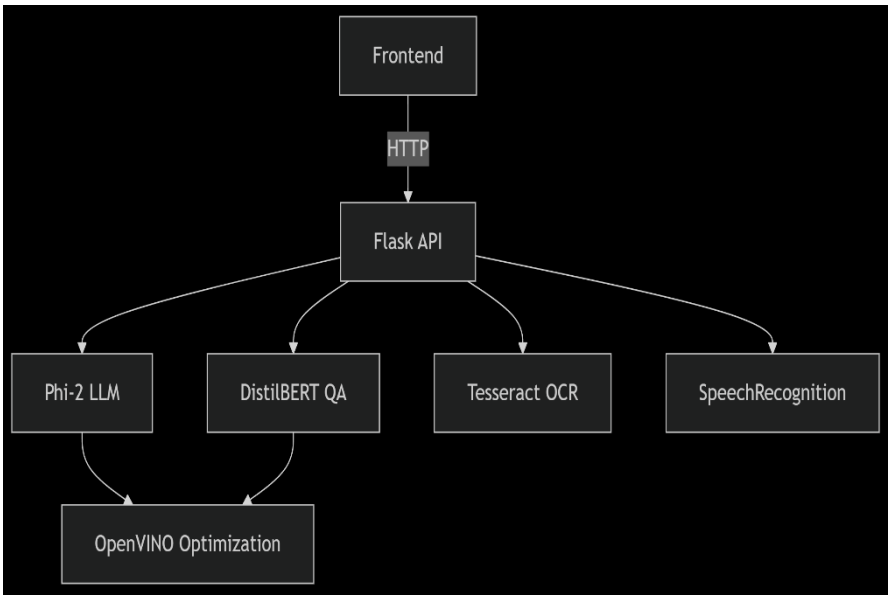
The architecture is divided into two layers:

Frontend Layer

- This is what the user sees — a website.
- Made using React.js, it allows:
  - Typing questions in an input box
  - Clicking a button to record voice
  - Uploading an image (like a textbook page or diagram)

Backend Layer

- This is the brain of the system.
- Built using Flask, a Python web framework.
- Handles logic and AI model predictions.
- Connects to AI models for:
  - Text question answering (NLP)
  - Voice-to-text conversion (STT)
  - Image text extraction (OCR)



Module Breakdown

Module Name	Role
frontend/src/components/QuestionForm.js	Accepts user input (text/speech)
frontend/src/components/ImageUploader.js	Handles image upload and display
backend/app.py	Flask server and API routes
backend/model_utils.py	Loads and uses the OpenVINO model
backend/models/	Stores OpenVINO-optimized AI models
uploads/	Temporary image upload folder for OCR
requirements.txt	Stores all Python packages needed

AI Models Used

Text Q&A: Phi-2 (HuggingFace Transformers)

- Very small but powerful model that can answer questions from a given context.
- Works efficiently with OpenVINO for faster response.

Speech Recognition: Google SpeechRecognition API

- Converts your voice into text.
- Works well if internet and microphone quality are decent.

OCR (Image Text): Tesseract

- Reads printed text from an image.
- Converts textbook photos into machine-readable text.

Key Components

Component	Technology	Purpose
LLM Engine	Phi-2 + OpenVINO	General Q&A
Document QA	DistilBERT + Tesseract	Textbook/image analysis
Speech Interface	Google Speech-to-Text	Voice queries
Optimization	OpenVINO	3x faster CPU inference

API Endpoints

Endpoint	Method	Input	Output
/ask	POST	{"question": "..."}	LLM response
/document-qa	POST	image + question	Extracted answer

/transcribe	POST	audio_file	{"text": "..."}

Benchmark Results

Metric	Vanilla PyTorch	OpenVINO-Optimized	Improvement
Latency (ms)	2100	740	2.8x faster
RAM Usage (GB)	5.1	3.2	37% reduction
Model Size (GB)	2.4	1.1	54% smaller

Expected Size After Optimization

Format	Size	Speed	Accuracy
FP32 (Original)	~5GB	1x	100%
FP16	2.5GB	1.5x	99.9%
INT8 (OpenVINO)	1.3GB	3x	99%
ONNX INT8	1.1GB	2.8x	98%

Why Phi-2 is 5GB+

1. **Default Precision**
    - The original Phi-2 is stored in FP32 (32-bit float) format

- Each parameter uses 4 bytes → 2.7B parameters × 4B = ~10.8GB
- Some compression reduces this to ~5GB

2. **Unoptimized Format**
    - PyTorch .bin weights include extra metadata
    - No quantization applied by default
3. **Tokenizer Files**
    - Adds ~50MB extra (configs, vocab files)

How to Reduce Model Size

FP16 Quantization

(Python)

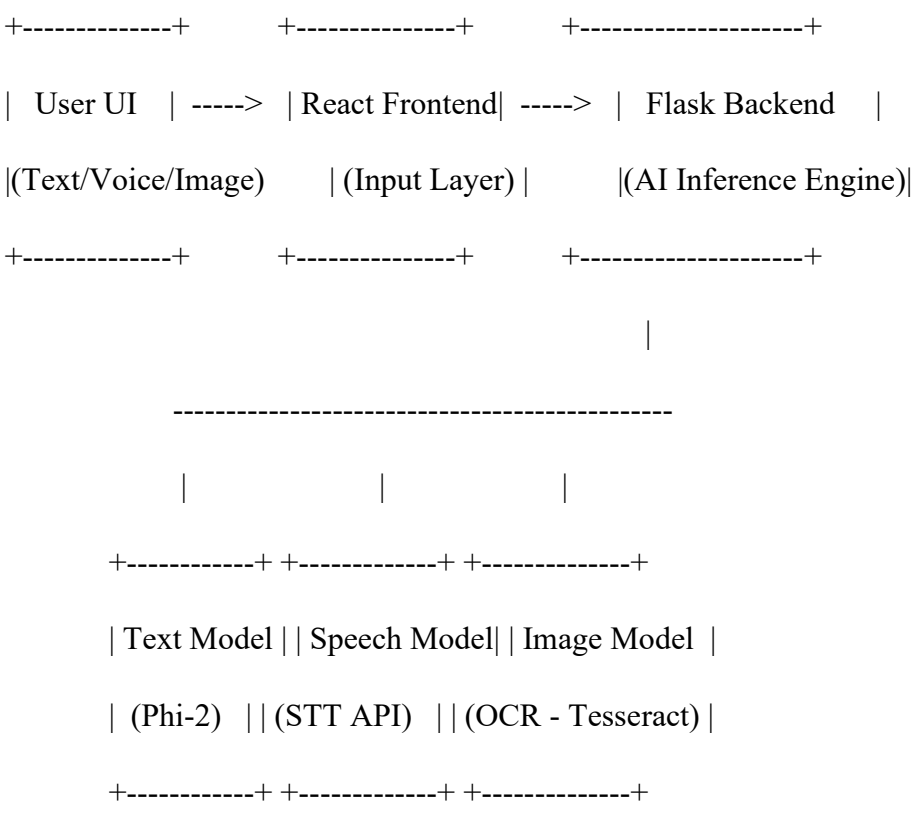
```
from transformers import AutoModelForCausalLM
import torch
model = AutoModelForCausalLM.from_pretrained(
   "microsoft/phi-2",
   torch_dtype=torch.float16,  # ← Critical change
   trust_remote_code=True
)
model.save_pretrained("./phi-2-fp16")
```

Result: ~2.5GB (50% reduction) give the code in text

Future Enhancements

1. **Emotion Detection**: Use a camera to check if a student is bored or confused.
2. **Progress Dashboard**: Allow teachers to view what students ask most often.
3. **Multilingual Support**: Let students ask questions in Telugu, Hindi, or Tamil.
4. **Voice Output**: Convert answers back to speech using TTS (text-to-speech).
5. **Offline Speech Recognition**: Use Whisper tiny model instead of Google API.

Data Flow Diagram



Critical Notes

1. **Hardware Requirements**:
    - INT8 needs CPU with AVX-512/VNNI support (most modern Intel chips)
    - FP16 works on any GPU/CPU
2. **Accuracy Tradeoff**:
    - INT8 may reduce accuracy by ~1-2% on complex questions
    - FP16 has negligible impact
3. **Memory vs Disk**:
    - Disk size ≠ RAM usage (quantized models still load into FP32 for computation)

References

1. OpenVINO Documentation
2. Phi-2 Model - HuggingFace
3. Tesseract OCR
4. Google SpeechRecognition API
5. React.js Docs
6. Flask Docs

Folder Structure

INTEL_UNNATI_PROJECT/

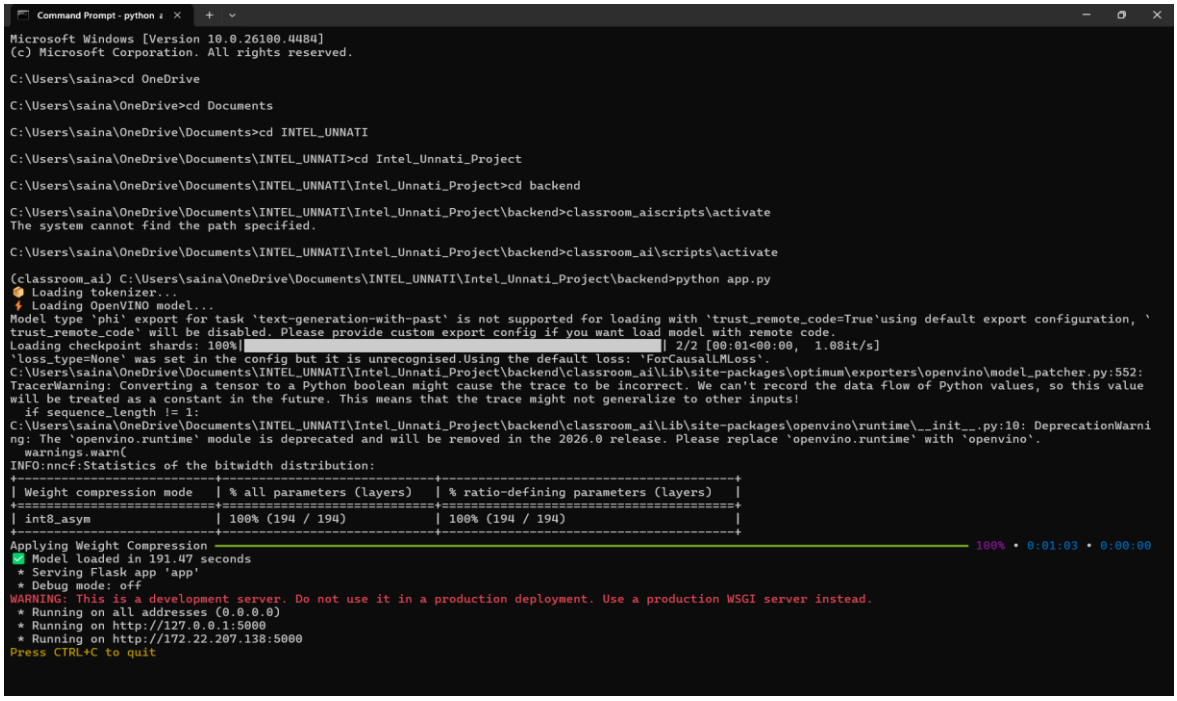
```
│
├── backend/
│   ├── __pycache__/
│   ├── classroom_ai/
│   ├── uploads/
│   ├── 1.14.0/
│   ├── app.py
│   ├── model_utils.py
│   └── requirements.txt
│
├── frontend/
│   ├── node_modules/
│   ├── public/
│   └── src/
│
├── models/
│   ├── phi2_openvino/
│   └── phi2_openvino_fp16/
│
├── uploads/
│
├── .gitignore
```

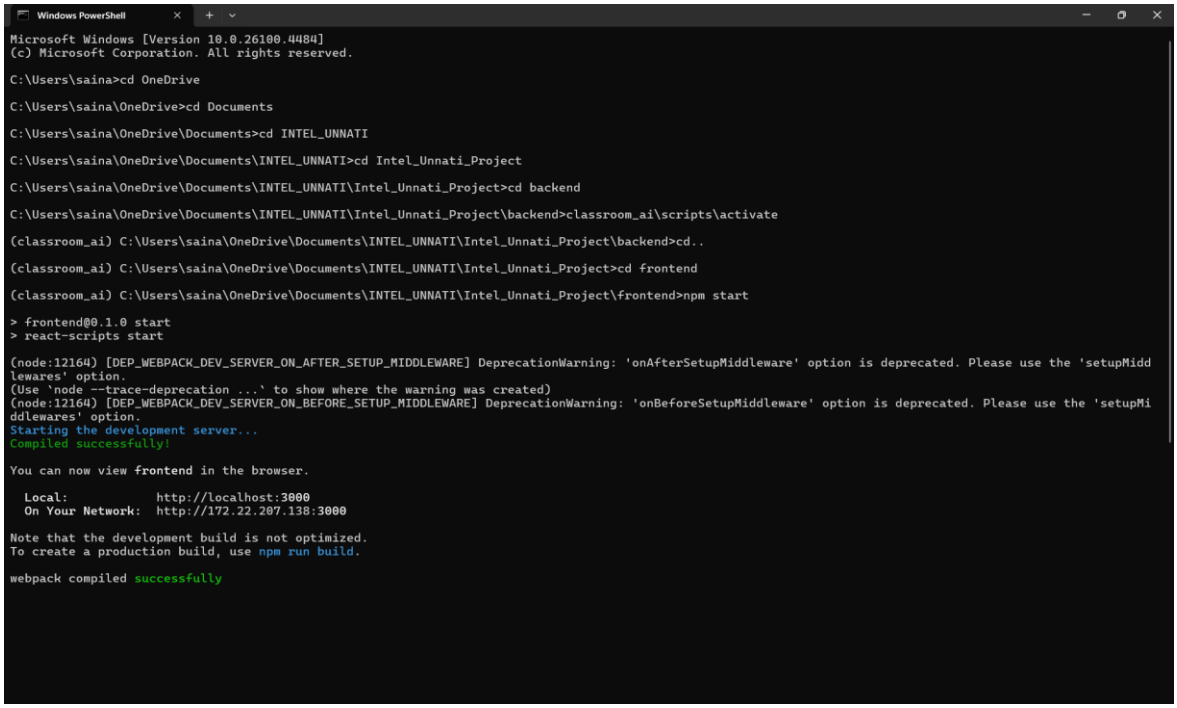
├── package-lock.json

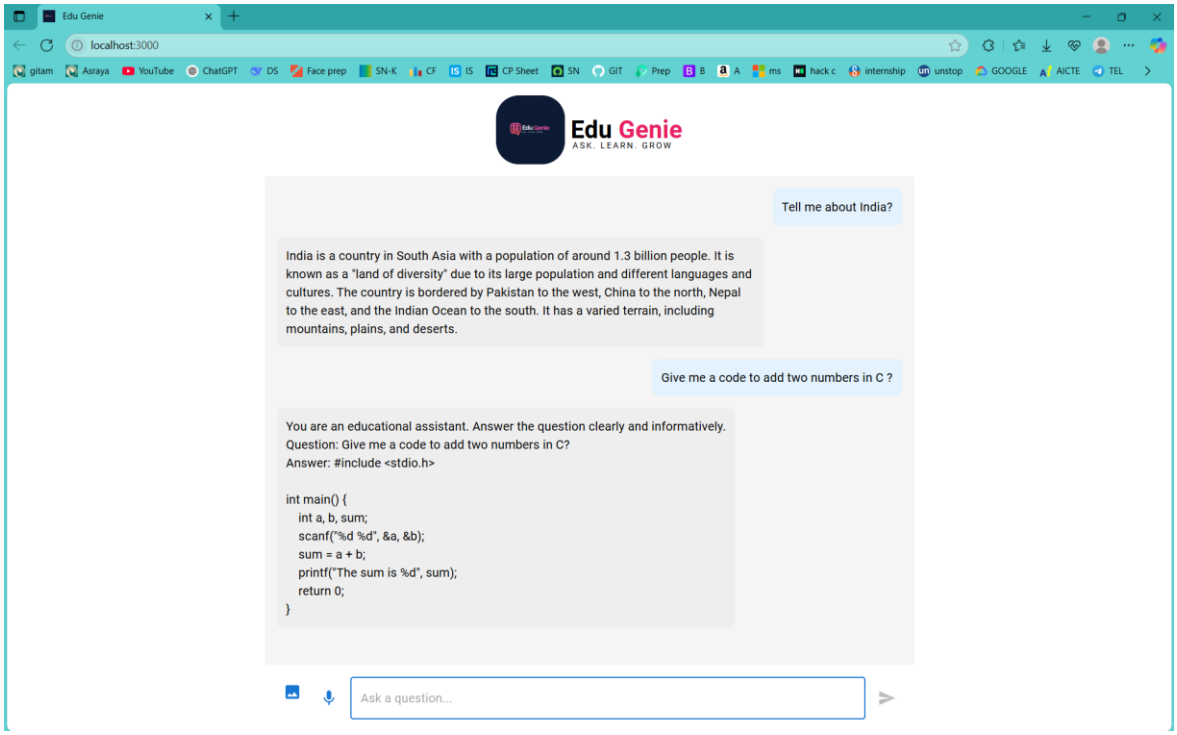
├── package.json

└── README.md

Sample output Images:







Conclusion

To build and run the Educational AI Assistant, you need:

Python backend dependencies like Flask, Transformers, OpenVINO, and OCR tools.

- System-level tools such as Tesseract OCR and an active internet connection for Google Speech-to-Text.

A clear and organized folder structure to manage code, models, and user inputs like images or voice files.

- Proper installation of the Phi-2 language model in OpenVINO format for fast and efficient responses.

Once the setup is done:

The assistant will:

- Answer text or voice questions using the Phi-2 AI model
- Understand scanned notes or textbook images through OCR

Respond quickly, even on standard Intel CPUs (thanks to OpenVINO optimization)

This system makes learning more interactive, accessible, and personalized for students. It also helps teachers by offering real-time support, reducing manual efforts, and making education more AI-powered and modern.