INTEL-UNNATI AI&ML JOURNAL
Number 2 (2025), pp. 1–13

Code:IUIT-2025

MACHINE LEARNING

AI-Powered Interactive Learning Assistant for Classrooms

Elizabeth Benny, Navya R, and Parvathy S Nair

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: In the era of personalized education, artificial intelligence can profoundly transform classroom learning experiences.EduBotics Assistant is an AI-powered interactive learning system designed to operate fully offline, making it ideally suited for schools and educational institutions with limited or no internet connectivity. The system integrates a Streamlit-based frontend with a Flask backend to deliver a seamless user experience across multiple input modalities text, voice, and images. Leveraging an optimized Gemma-2B large language model deployed via Intel's OpenVINO toolkit, the assistant processes natural language queries in real time while ensuring low-latency, efficient CPU-based inference. For voice interactions, the system incorporates the Vosk speech recognition toolkit to transcribe spoken queries, and Tesseract OCR is employed to extract text from images. An interactive quiz module further enhances the learning process by dynamically generating multiple-choice questions based on user-defined topics, thereby supporting self-assessment and active learning. The fully offline design not only ensures data privacy and faster response times but also makes advanced educational tools accessible in remote or underserved regions.

Keywords: Offline AI Assistant, EduBotics, Gemma-2B, OpenVINO, Vosk, Tesseract OCR, Streamlit, Flask, Quiz Generation, Multimodal Learning, Educational Technology, Voice Recognition, Image-Based QA, Self-Assessment.

1 Introduction

Artificial intelligence (AI) is changing how students interact with course materials, obtain knowledge, and improve their comprehension. Despite these developments, a lack of internet connectivity still poses a problem for many educational institutions, particularly those in underserved or rural areas. Due to inadequate or nonexistent internet connectivity, served regions still encounter difficulties. In order to bridge this gap, we introduce

EduBotics Assistant, an interactive learning platform that is entirely offline and powered by AI. It is made to operate independently of cloud services and internet APIs.With the help of a locally deployed language model, this system allows students to engage using text, speech, and graphics as well as receive precise, context-aware responses. With a Flask backend managing modular API-based interactions across all features and a Streamlit frontend offering a straightforward, responsive, and user-friendly interface, the assistant is constructed with a Python-based architecture.

The Gemma-2B language model powers the backend, which has been tuned with Intel's OpenVINO toolset to guarantee quick and effective inference on CPU-based systems.Tesseract OCR supports image-based inquiries by extracting legible material from submitted pictures for interpretation, while Vosk speech recognition toolset processes voice input to enable accurate offline transcription. Furthermore, students can create and take topic-specific multiple-choice tests using a dynamic quiz generation module, which encourages interactive self-learning and knowledge evaluation.

The EduBotics Assistant retains full offline functionality because all models and dependencies are housed locally, guaranteeing data privacy, minimal latency, and system dependability.It provides intelligent and multimodal learning support to schools without requiring internet connectivity, making it a scalable and accessible option for educational institutions with limited resources.

2 Libraries Used

In the project for various tasks, following packages are used.

```
flask
flask_cors
transformers
optimum.intel
PIL (Pillow)
pytesseract
torch
wave
os
subprocess
json
numpy
vosk
requests
sounddevice
soundfile
tempfile
base64
streamlit
```

3 Methodology

To develop a fully offline, intelligent educational assistant tailored for classroom environments, we adopted a modular, multi-modal architecture. The system supports various input types text, voice, and images and delivers accurate educational responses using offline

natural language processing, speech recognition, and image analysis. The methodologies are designed for real-time performance on local hardware without relying on internet access.

Offline AI Inference with OpenVINO:

- **Model Deployment:** The Gemma-2B language model is optimized using Intel OpenVINO toolkit for efficient CPU-based inference.
- **Latency Optimization:** Model execution time is significantly reduced, allowing faster and more interactive learning experiences even without GPU or cloud support.

Natural Language Processing (NLP):

- **Prompt Engineering:** User questions are embedded within structured prompts to guide the model in generating focused, relevant answers.
- **Tokenization:** Hugging Face's AutoTokenizer is used to prepare text for the model.
- **Response Extraction:** Raw model output is filtered to extract only the relevant answer portion using string parsing.

Voice Input Processing:

- **Audio Capture:** The Streamlit frontend records voice input using the system microphone.
- **Conversion and Transcription:** The audio is converted to `.wav` using FFmpeg and transcribed using the offline Vosk model.
- **NLP Integration:** Transcribed text is then processed through the AI model for response generation.

Image-Based Question Processing:

- **OCR Extraction:** Tesseract OCR processes uploaded images to extract readable text.
- **AI Querying:** The extracted text is embedded in a prompt and passed to the AI model for interpretation and response.

Dynamic Quiz Generation:

- **MCQ Creation:** The user inputs a topic, and the AI model returns a JSON array containing five MCQs.
- **Frontend Parsing:** Streamlit dynamically displays the questions and captures user responses via radio buttons.
- **Evaluation:** On submission, user answers are validated against the correct ones, and the score is displayed immediately.

Flask-Based Backend API:

- **Modular Endpoints:** Four REST endpoints chat, voice, image, and quiz handle different input types.
- **Model Integration:** Each endpoint processes user input, formats it, and performs inference using the Gemma-2B model or Vosk for transcription.

Streamlit Frontend Interface:

- **Unified UI:** Built entirely in Python using Streamlit, offering a clean, intuitive browser-based interface.

- **Mode Selection:** Sidebar options let users switch between Chat, Image Q&A, Voice Q&A, and Quiz modes.
- **User Personalization:** A welcome screen captures the user's name and maintains conversational continuity.

Error Handling and Resilience:

- **Frontend Validation:** Prevents invalid inputs and provides user-friendly alerts.
- **Robust Backend Parsing:** Handles malformed JSON during quiz generation and logs raw outputs for debugging.
- **File Cleanup:** Temporary audio files are deleted after use to conserve system resources.

4 Implementation

The implementation of this offline AI-powered educational assistant is divided into three integrated modules: the Streamlit-based frontend interface, the Flask backend server, and the offline AI model inference pipeline. The complete architecture is optimized to work entirely offline on local machines without GPU or internet access.

Frontend Interface: The user-facing application is built entirely using Streamlit, allowing dynamic user interaction in a browser environment while running purely on Python. Upon startup, a welcome screen prompts the user to input their name and transitions to the main interface, which supports four core modes:

- **Text Q&A:** Allows users to type questions and receive AI-generated responses.
- **Voice Q&A:** Enables voice interaction by capturing microphone input (WebM), recording 5-second clips, and forwarding them for transcription.
- **Image Q&A:** Lets users upload an image (e.g., textbook snippet), which is then processed via OCR and sent to the AI for answers.
- **Quiz Generator:** Accepts a topic as input, generates 5 MCQs using the model, and dynamically displays them with interactive answer selection and scoring.

Each mode is accessed via a sidebar and switches seamlessly using Streamlit's session state management. All UI components including image uploads, radio buttons, and dynamic results are rendered using Python.

Backend Server: The backend is developed using `Flask` with CORS enabled for secure cross-origin communication with the frontend. The backend exposes multiple endpoints that process user queries based on modality:

- `chat`: Accepts a text prompt, tokenizes it, and returns a response from the AI model.
- `voice`: Accepts voice recordings, converts them to WAV using FFmpeg, transcribes using the Vosk STT engine, and sends text to the model.
- `image`: Uses PIL and Tesseract OCR to extract text from uploaded images and queries the model.
- `quiz`: Accepts a user-defined topic and returns five structured MCQs as a JSON array using prompt-based generation.

Each request is tokenized via Hugging Face's AutoTokenizer and processed using an OpenVINO-optimized Gemma-2B model running locally on CPU.

Offline AI Model Inference: The AI inference stack is entirely offline and includes:

- **Gemma-2B (OpenVINO):** Deployed for generating answers to user queries and MCQ generation. Optimized for CPU execution.
- **Vosk:** Used for offline voice-to-text transcription, integrated with FFmpeg to preprocess audio.
- **Tesseract OCR:** Extracts text from uploaded images and supports image-based Q&A functionality.

All models are pre-downloaded and loaded at runtime. No internet or external APIs are required.

Error Handling and Robustness: The application includes built-in error validation across both frontend and backend. The frontend prevents empty input submissions and alerts users when recordings are missing. On the backend, try-except blocks capture errors such as malformed audio, failed model outputs and JSON decoding failures (e.g., during quiz generation). Temporary files like audio clips are automatically removed after use.

Offline-Centric Design: The entire system is designed for disconnected environments. All models and libraries run offline and the system is optimized for low-resource devices using CPU-based inference. This makes it suitable for classrooms, rural labs or edge devices where cloud access is unavailable.

5 Results & Discussion

The EduBotics Assistant was successfully developed and evaluated as a fully offline, AI-powered educational support system. Built using a Streamlit frontend and a Flask backend, the system integrates four key functionalities **Text Q&A, Voice Q&A, Image Q&A, and Quiz Generation** all powered by the **Gemma-2B language model optimized with OpenVINO**. Additional modules include **Vosk** (for offline speech-to-text), **Tesseract OCR** (for image processing) and **FFmpeg** (for audio conversion).

The performance summary of all major modules in the EduBotics Assistant is shown in Table 1, highlighting average response times, models or tools used, accuracy levels and user experience. This helps in evaluating the system's overall efficiency and responsiveness across different input modes.

Table 1: Performance Summary of EduBotics Assistant Modules

No.	Module	Avg. Time	Model/Tool Used	Accuracy Level	User Experience
1	Text Q&A	6–12 sec	Gemma-2B (OpenVINO)	High (Contextual)	Informative and accurate
2	Voice Q&A	8–15 sec	Vosk + Gemma-2B	High (for clear speech)	Hands-free interaction
3	Image Q&A	7–12 sec	Tesseract + Gemma-2B	Moderate to High	Good with printed text
4	Quiz Generator	10–18 sec	Gemma-2B (OpenVINO)	High Relevance	Real-time evaluation

An overview of the backend technologies integrated into the system can be found in Table 2. It lists the frameworks and tools used for each feature, demonstrating how different components such as Flask, OpenVINO Vosk, and Tesseract are orchestrated to enable offline functionality.

Table 2: Backend Technologies Used in EduBotics Assistant

No.	Feature	Frameworks / Tools Used
1	Text Input	Flask + Gemma-2B (OpenVINO)
2	Voice Input	Flask + Vosk + FFmpeg
3	Image Q&A	Flask + Tesseract + Gemma-2B
4	Quiz Generation	Flask + Prompt Engineering + Gemma-2B

The system 's features are further assessed in terms of usability and offline support in Table 3. This evaluation matrix outlines the input types accepted by each module, the level of offline support, and the relative ease of use, validating the assistant's practicality in classroom settings.

Table 3: Feature Evaluation Matrix

No.	Feature	Input Type	Offline Support	Ease of Use
1	Text Q&A	Text (Keyboard)	Fully Offline	High
2	Voice Q&A	Audio (Mic)	Fully Offline	Medium–High
3	Image Q&A	Image Upload	Fully Offline	Medium
4	Quiz Generator	Text (Topic)	Fully Offline	High

Text Q&A: Delivered highly contextual answers in 6–12 seconds using Gemma-2B. Ideal for fact-based academic queries.
Voice Q&A: Combined Vosk STT (3–6 sec) with AI inference (5–9 sec), giving a hands-free, responsive interaction.
Image Q&A: Tesseract OCR extracted printed text accurately. Output quality was moderately affected for handwritten or low-quality images.
Quiz Generation: Automatically generated 5 MCQs from a topic prompt with accurate scoring and UI integration. Response time: 10–18 seconds.

The results validate that the EduBotics Assistant meets the goal of providing intelligent and responsive educational support completely offline.

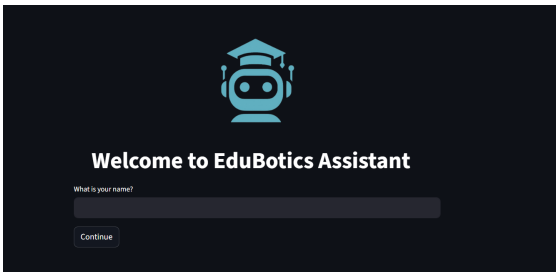
6 Conclusions

The creation of the AI-powered Interactive Learning Assistant shows how sophisticated language models, speech recognition, and picture processing can be combined to produce

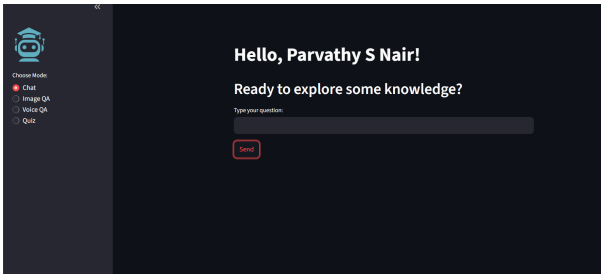
a full instructional tool that operates completely offline. Through the use of technologies like Tesseract OCR, Vosk for speech recognition, and the OpenVINO-optimized Gemma-2B model, the system offers a smooth and engaging learning experience across a variety of input types, including text, audio, and images.

This project's capacity to operate without internet connectivity is one of its main advantages, making it extremely accessible for educational institutions and schools in rural or underdeveloped locations. Through a dynamic quiz creation module, the system encourages active learning in addition to providing efficient solutions to academic questions.

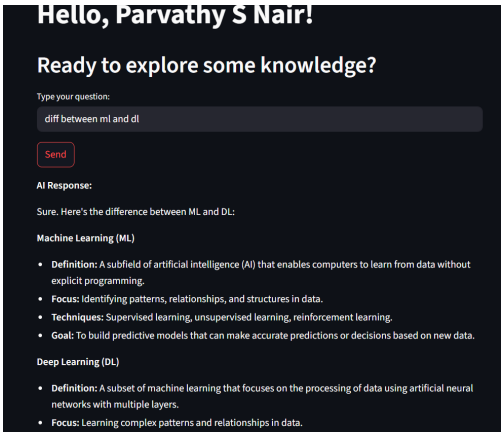
Both educators and students can interact with the assistant in an intuitive manner because of its user-friendly design. All things considered, the project's objective of developing an intelligent, offline teaching aid for schools is accomplished. It creates opportunities for additional improvements, including support for many languages, performance statistics, or content personalization, which can increase its usefulness and influence in the educational sector.



Welcome screen prompting for user name



Main interface showing all features



Text Q&A: Chat input and response

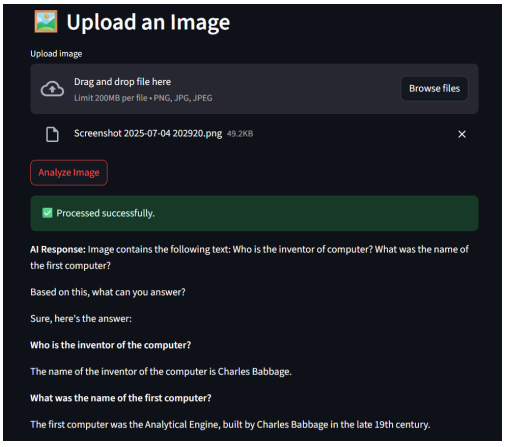
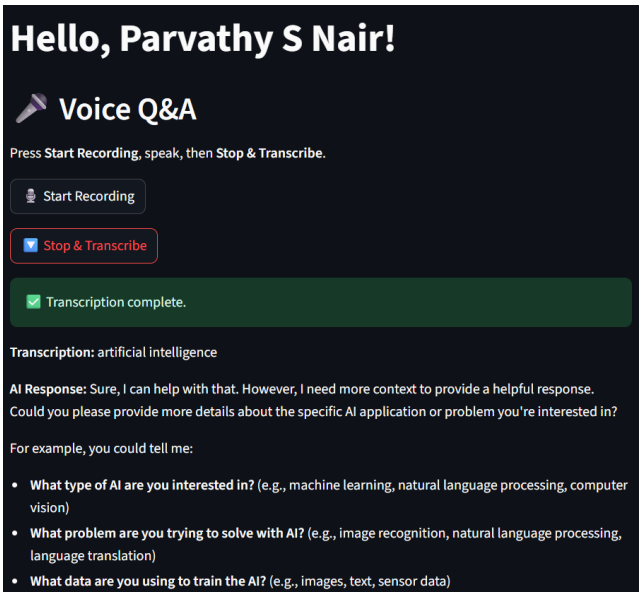
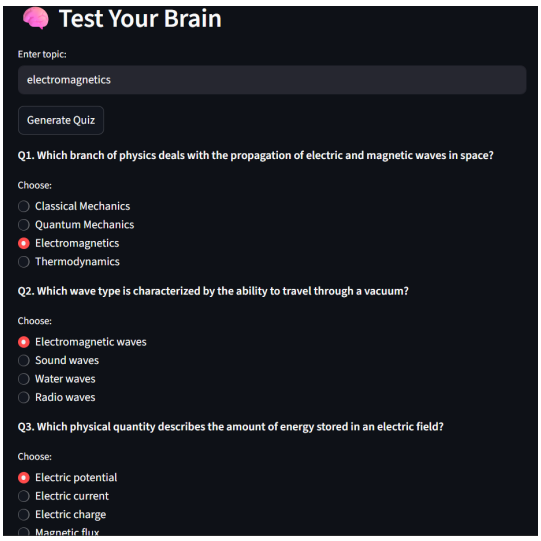


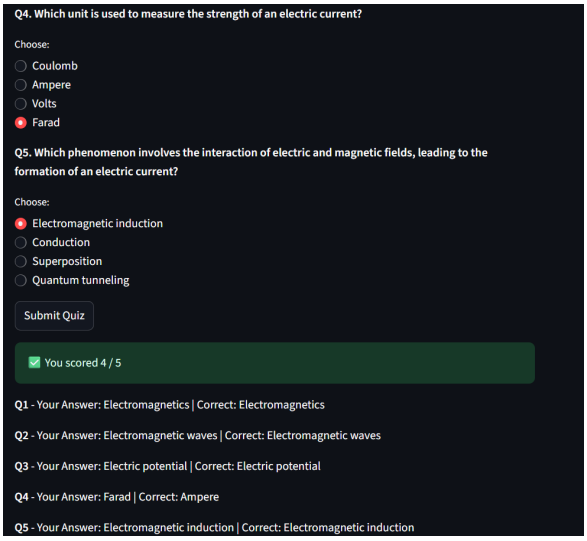
Image Q&A: Upload and analysis



Voice Q&A: Transcription and response



(a) Quiz-based Interaction



(b) Quiz generation and evaluation

Acknowledgments

We take this opportunity to express our sincere gratitude to all those who supported and guided us throughout the successful completion of our project titled "AI-Powered Interactive Learning Assistant for Classrooms", developed under the platform EduBotics. We

would like to express our heartfelt thanks to our project guide, Dr Starlet Ben Alex, for their continuous encouragement, expert advice, and invaluable support during the various phases of the project. We are grateful to the Head of the Department and all the faculty members of the Electronics and Communication Engineering Department for providing us with the facilities and academic environment necessary for the successful execution of this work. We also gratefully acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under the Intel© - Unnati Programme, whose expertise and encouragement have been instrumental in shaping our project and technical skills. We sincerely thank our institution for providing us with the opportunity and resources to pursue this innovative project. Finally, we extend our deep appreciation to our friends and family for their motivation and to all our team members for their teamwork, dedication, and consistent efforts in bringing this project to life.

Individual Contributions

Elizabeth Benny

- Contributed to the development of the text-based chatbot feature, allowing users to input questions and receive AI responses.
- Assisted in setting up the input flow and ensuring a smooth conversational interface.
- Helped in formatting and refining the user interface for consistency.
- Actively participated in the preparation of the project report, especially in documenting the chatbot functionality and layout.

Navya R

- Worked on integrating the image-based question answering module, enabling the assistant to process uploaded images using OCR.
- Helped with the interface layout and flow of the image QA section.
- Supported the refinement of user experience in that module.
- Contributed to writing parts of the final report related to image processing and interface design.

Parvathy S Nair

- Led the complete design and development of the AI-powered web platform, ensuring all modules were integrated into a unified and fully offline-capable system.
- Built and implemented the voice-based interaction system using Vosk for offline speech recognition and AI-based response generation.
- Developed the topic-wise quiz generator with automatic evaluation and scoring features.
- Managed the backend integration, model handling, audio processing, and seamless coordination between all features.
- Played a major role in testing, optimization, and final validation of the platform, while also assisting in structuring and reviewing the overall project report.

References

[1] INTEL CORPORATION. *OpenVINO Toolkit Documentation*, 2024. Last Accessed July 8, 2025.

[2] HUGGING FACE. *Transformers: State-of-the-art Machine Learning for PyTorch, TensorFlow, and JAX*, 2024. Last Accessed July 8, 2025.

[3] HUGGING FACE. *Gemma-2B Model Card*, 2024. Last Accessed July 8, 2025.

[4] STREAMLIT INC. *Streamlit: The Fastest Way to Build Data Apps in Python*, 2024. Last Accessed July 8, 2025.

[5] ALPHA CEPHEI. *Vosk Speech Recognition Toolkit*, 2024. Last Accessed July 8, 2025.

[6] GOOGLE. *Tesseract Open Source OCR Engine*, 2024. Last Accessed July 8, 2025.

[7] PYTHON SOFTWARE FOUNDATION. *Flask: Web Development One Drop at a Time*, 2024. Last Accessed July 8, 2025.

[8] FFMPEG DEVELOPERS. *FFmpeg: A Complete, Cross-Platform Solution to Record, Convert, and Stream Audio and Video*, 2024. Last Accessed July 8, 2025.

[9] ANBALAGAN, S. *Advancing Knowledge from Multidisciplinary Perspective: Engineering, Technology, and Management*, In Proc. of the International Conference, 2024, Last Accessed July 8, 2025.

[10] PIATNITCKAIA, E. *Development of Speech Technologies in Educational Process*, In INTED2024 Proceedings, 2024. Last Accessed July 8, 2025.

[11] KUMAR, S., AND YADAV, B. *Artificial Intelligence-Based Voice Assistant*, In 2022 International Conference on Intelligent Technologies (CONIT), Belagavi, India, 2022. Last Accessed July 8, 2025.

A Main code sections for the solution

A.1 Welcome User Input (Frontend - Streamlit)

Show welcome screen and collect user name.

```
if not st.session_state.entered:
 st.session_state.name = st.text_input(" What is your name?")
if st.button("Continue") and st.session_state.name.strip():
        st.session_state.entered = True
        st.rerun()
```

A.2 Text-based Chatbot (Frontend + Backend)

A.2.1 Frontend

```
question = st.text_input("Type your question:")
if st.button("Send") and question.strip():
    res = requests.post(f"{BACKEND_URL}/chat", json={"question": question})
```

A.2.2 Backend Endpoint

```
@app.route("/chat", methods=["POST"])
def chat():
    data = request.get_json()
    question = data.get("question", "").strip().lower()
    if question in ["hi", "hello", "hey"]:
        return jsonify({"answer": "Hello! How can I assist you today?"})

    prompt = f"User: {question}\nAssistant:"
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=
                                                1024)
    with torch.no_grad():
        output = model.generate(**inputs, max_new_tokens=150, do_sample=False,
                                pad_token_id=tokenizer.eos_token_id)
    reply = tokenizer.decode(output[0], skip_special_tokens=True).split("Assistant
                                                :")[-1].strip()
    return jsonify({"answer": reply})
```

A.3 Image-based Question Answering (OCR + QA)

A.3.1 Frontend

```
uploaded = st.file_uploader("Upload image", type=["png", "jpg", "jpeg"])
res = requests.post(f"{BACKEND_URL}/image", files={"image": uploaded})
```

A.3.2 Backend Endpoint

```
@app.route("/image", methods=["POST"])
def image_qa():
    image = request.files["image"]
    img = Image.open(image.stream).convert("L")
    extracted_text = pytesseract.image_to_string(img)
    prompt = f"Image contains the following text:\n{extracted_text.strip()}"
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=
                                                1024)
    with torch.no_grad():
        output = model.generate(**inputs, max_new_tokens=150, do_sample=False,
                                pad_token_id=tokenizer.eos_token_id)
    reply = tokenizer.decode(output[0], skip_special_tokens=True).split("Assistant
                                                :")[-1].strip()
    return jsonify({"answer": reply})
```

A.4 Voice-based Question Answering (Voice to Text + QA)

A.4.1 Frontend

```
audio_data = sd.rec(...)
sf.write(tmp.name, audio_data, ...)
res = requests.post(f"{BACKEND_URL}/voice", files={"audio": f})
```

A.4.2 Backend Endpoint

```
   @app.route("/voice", methods=["POST"])
def voice():
    audio = request.files['audio']
    wav_path = "input.wav"
    audio.save(wav_path)
    subprocess.run(["ffmpeg", "-y", "-i", wav_path, "-ar", "16000", "-ac", "1", "
                                            converted.wav"])
    wf = wave.open("converted.wav", "rb")
    recognizer = KaldiRecognizer(vosk_model, wf.getframerate())
    result_text = ""
    while True:
        data = wf.readframes(4000)
        if len(data) == 0:
            break
        if recognizer.AcceptWaveform(data):
            result_text += json.loads(recognizer.Result()).get("text", "") + " "
    result_text += json.loads(recognizer.FinalResult()).get("text", "")
    transcription = result_text.strip()
    prompt = f"User: {transcription}\nAssistant:"
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=
                                            1024)
    with torch.no_grad():
        output = model.generate(**inputs, max_new_tokens=150, do_sample=False,
                                pad_token_id=tokenizer.eos_token_id)
    reply = tokenizer.decode(output[0], skip_special_tokens=True).split("Assistant
                                            :")[-1].strip()
    return jsonify({"answer": reply, "transcription": transcription})
```

A.5 Quiz Generator (Topic-Based MCQs)

A.5.1 Frontend

```
res = requests.post(f"{BACKEND_URL}/quiz", json={"topic": topic})
```

A.5.2 Backend Endpoint

```
@app.route("/quiz", methods=["POST"])
def quiz():
    data = request.get_json()
    topic = data.get("topic", "").strip()
    prompt = f"""
Generate exactly 5 multiple-choice questions on the topic "{topic}".
```

```
Each must be a JSON object with "question", "options", and "answer".
Return only a JSON array of 5 such objects.
"""
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=
                                                512)
    with torch.no_grad():
        output = model.generate(**inputs, max_new_tokens=700, pad_token_id=
                                                    tokenizer.eos_token_id)
    decoded = tokenizer.decode(output[0], skip_special_tokens=True)
    json_data = decoded[decoded.find("["):decoded.rfind("]")+1].replace("âĂIJ", '"'
                                                ).replace("âĂİ", '"')
    questions = json.loads(json_data)
    return jsonify({"questions": questions})
```

B Model Initialization (Backend)

B.1 Gemma-2B Model via OpenVINO

```
model_dir = "openvino_model"
tokenizer = AutoTokenizer.from_pretrained(model_dir)
model = OVModelForCausalLM.from_pretrained(model_dir)
```

B.2 Vosk Voice Model

```
vosk_model_path = "vosk-model-en-us-0.22"
vosk_model = VoskModel(vosk_model_path)
```

B.3 Server App Launch

B.3.1 Backend Flask Runner

```
if __name__ == "__main__":
    app.run(port=5000)
```

B.3.2 Frontend Streamlit Config

```
st.set_page_config(page_title="EduBotics Assistant", layout="centered")
```