**Project Report**

**Authors –**

* Suchit Gupta, Manipal Institute of Technology, Manipal
* J. Karthikeyan, Manipal Institute of Technology, Manipal
* Arnav Jagia, Manipal Institute of Technology, Manipal

**GitHub link of the project repository** - [Suchit027/Intel-Unnati-Program](https://github.com/Suchit027/Intel-Unnati-Program/tree/main)

**Project Title: AI-Powered Interactive Learning Assistant for Classrooms**

**1. Introduction**

Modern education demands inclusive, intelligent systems that support diverse classroom needs. This project presents an AI-powered smart classroom assistant that integrates multiple AI capabilities to automate and assist various classroom activities. The system features automated facial recognition-based attendance, speech-to-text transcription for deaf students, text-to-speech support for non-verbal communication, and automatic summarization of academic materials. Built using Django (backend) and Next.js (frontend), this application combines accessibility, automation, and AI to improve classroom interaction.

**2. Objectives**

* Automate the attendance process through facial recognition.
* Convert live speech to text for deaf or hard-of-hearing students.
* Convert written text into speech for non-verbal or speech-impaired teachers.
* Summarize academic PDF materials for both students and teachers.
* Build a full-stack web application with Django (backend) and Next.js (frontend).

**3. Key Features**

**3.1 Facial Recognition for Attendance**

Core Components Used:

* MTCNN for face detection
* InceptionResnetV1 (pretrained on VGGFace2) for face embedding
* Cosine similarity for matching

Workflow:

* Input image is processed using MTCNN to detect and crop the face.
* The face is then passed to the InceptionResnetV1 model to extract a 512-dimensional embedding vector.
* Matching is performed using cosine similarity between the input face embedding and the saved embeddings of registered students.
* If similarity exceeds a threshold (default 0.7), the student is marked present.

**3.2 Speech-to-Text for Deaf Students**

Model Used:

* openai/whisper-large-v3 – State-of-the-art automatic speech recognition model.
* Hugging Face pipeline "automatic-speech-recognition" used for ease of integration.

Workflow:

* Users upload either audio or video files via a web form.
* If a video is uploaded, ffmpeg is used to extract mono-channel, 16kHz audio.
* The Whisper pipeline processes the file and returns the transcribed text.
* Transcription is shown on the webpage for hearing-impaired students.

Features:

* Accepts multiple audio formats (.wav, .mp3, etc.) and even video files (.mp4, .mkv).
* Automatically converts and normalizes audio to Whisper-compatible format.
* High accuracy even with background noise, various accents, or mixed languages.
* Leverages the transformers library for high-level access to Whisper.

**3.3 Text-to-Speech for Non-Verbal Teachers**

Model Used:

* microsoft/speecht5\_tts – Transformer-based neural TTS model
* Speaker embedding sourced from: Matthijs/cmu-arctic-xvectors

Workflow:

* Teachers input text via the web interface.
* The Django backend uses SpeechT5Processor to tokenize input text.
* SpeechT5 generates waveform audio conditioned on a learned speaker embedding, simulating natural voice output.
* Output is saved as a .wav file and played in-browser via a static route.

Speaker Identity:

* The speaker voice is simulated using precomputed x-vectors from the CMU Arctic dataset (dataset[7306]["xvector"]), ensuring consistency and natural tone.

**3.4 PDF Summarization**

Model Used:

* facebook/bart-large-cnn: A transformer-based abstractive summarization model optimized for long documents.

Workflow:

* Students or teachers upload PDF files via a web form.
* The server extracts text content from each page using pdfplumber, preserving paragraph structure.
* Paragraphs longer than 10 words are selected for summarization.
* Each paragraph (truncated to 1024 words) is summarized using the BART model.
* Output includes both the **original paragraph** and the **generated summary**, rendered on the frontend.

Advantages:

* Produces coherent and concise summaries.
* BART handles academic language well and retains key technical points

**4. Technology Stack**

| **Component** | **Technology Used** |
| --- | --- |
| Backend Framework | Django (Python) |
| Frontend Framework | Next.js (React-based) |
| Database | SQLite (development) |

**5. Architecture Overview**

**A screenshot of a computer

AI-generated content may be incorrect.**

**6. Screenshots**

A screenshot of a video player

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a video stream

AI-generated content may be incorrect.

A screenshot of a register

AI-generated content may be incorrect.

A screenshot of a person's profile

AI-generated content may be incorrect.

**7. Challenges Faced**

* Integrating multiple AI services while keeping latency low.
* Handling noise and accents in speech recognition.
* Extracting meaningful content from scanned/poor-quality PDFs.
* Real-time processing in browser-compatible formats.

**8. Future Improvements**

* Support for multiple classrooms and user roles (Admin, Teacher, Student).
* Adding support for blind users with complete screen readers.
* Improve face recognition accuracy in low-light environments.
* Integration with school management systems (LMS APIs).

**9. Team Contributions –**

* Suchit Gupta was responsible for developing the facial recognition and text-to-speech modules. He also led the backend integration of all core components including facial recognition, text-to-speech, speech-to-text, and PDF summarization within the Django framework.
* J. Karthikeyan developed the speech-to-text and PDF summarization modules. He also handled the frontend integration of all features (facial recognition, text-to-speech, speech-to-text, and PDF summarization), connecting the Django backend with the Next.js-based user interface.
* Arnav Jagia provided critical support to both Suchit Gupta and J. Karthikeyan by assisting in model development and helping debug and optimize the Django backend and Next.js frontend. He also tested the accuracy of facial recognition model with a sample celebrity face dataset.
* The team collaboratively designed the project architecture, brainstorming and evaluating multiple ideas to finalize an effective and inclusive workflow for the application.

**10. Conclusion**

This project demonstrates how AI can be used to make classrooms more inclusive, automated, and efficient. With facial recognition for attendance, real-time transcription, speech synthesis, and content summarization, it empowers both students and teachers. Built with modern web technologies and deployed as a full-stack app, it can be scaled and integrated into educational institutions.