INTERNSHIP REPORT

AI-POWERED INTERACTIVE LEARNING ASSISTANT FOR CLASSROOMS

BY

TEAM TRISHUL

INTEL UNNATI INDUSTRIAL TRAINING

DURATION: 15/05/2025 TO 30/06/2025

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(DEEMED TO BE UNIVERSITY)

BENGALURU

BATCH: 2022-2026

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to **Debdyut Hajra**, our esteemed instructor from Intel, for their invaluable guidance, support, and constructive feedback throughout the duration of this project. Their insights were instrumental in shaping our understanding and overcoming various challenges.

We express our gratitude to our college mentor **Dr. Karthik S, Associate Professor**, Department of Electronics and Communication Engineering, Gandhi Institute of Technology and Management, Bengaluru, for their support, guidance, and suggestions throughout the project work.

We also extend our thanks to the developers of the open-source libraries and frameworks, including Streamlit, Hugging Face Transformers, OpenVINO, Whisper, and pyttsx3, which provided the foundational tools for this project. Their contributions to the AI and software development communities are truly appreciated.

Finally, we acknowledge the collaborative environment that fostered our learning and allowed us to bring the Smart Classroom Assistant to fruition.

ABSTRACT

The Smart Classroom Assistant is an innovative AI-Powered application designed to revolutionize the student learning experience. This project addresses common educational challenges such as information overload, the need for personalized study tools, and efficient knowledge retrieval. By integrating a hybrid architecture that leverages both local, optimized AI models (Flan-T5 for contextual Q&A, Emotion Classifier for user sentiment, both accelerated by OpenVino) and robust cloud-based services (Google Gemini API for general knowledge), the assistant provides a versatile and responsive platform. Key functionalities include intelligent Q&A from uploaded documents (TXT, PDF, DOCX), dynamic content summarization, interactive quiz generation, and a console-based voice interface for hands-free operation. This report details the systems architecture, features, implementation challenges, individual contributions, and outlines future enhancements, demonstrating a practical application of advanced AI techniques in an educational context.

TABLE OF CONTENTS

INTRODUCTION

AI in education is transforming traditional learning by providing personalized and interactive experiences. This project, **"AI-Powered Interactive Learning Assistant for Classrooms,"** aims to support students in clarifying academic doubts, revising topics, and learning effectively through intelligent text and voice-based interactions.

The Smart Classroom Assistant is an AI-Powered application designed to enhance the learning experience for students by providing intelligent assistance with study notes, general knowledge queries, and educational tools. Leveraging a hybrid approach of local and cloud-based AI models, the assistant offers features such as contextual question-answering from uploaded documents, real-time general knowledge Q&A with web search capabilities, dynamic quiz generation, and content summarization. The project aims to create an accessible and efficient study companion for modern learners.

PROJECT FEATURES

The Smart Classroom Assistant provides the following key functionalities:

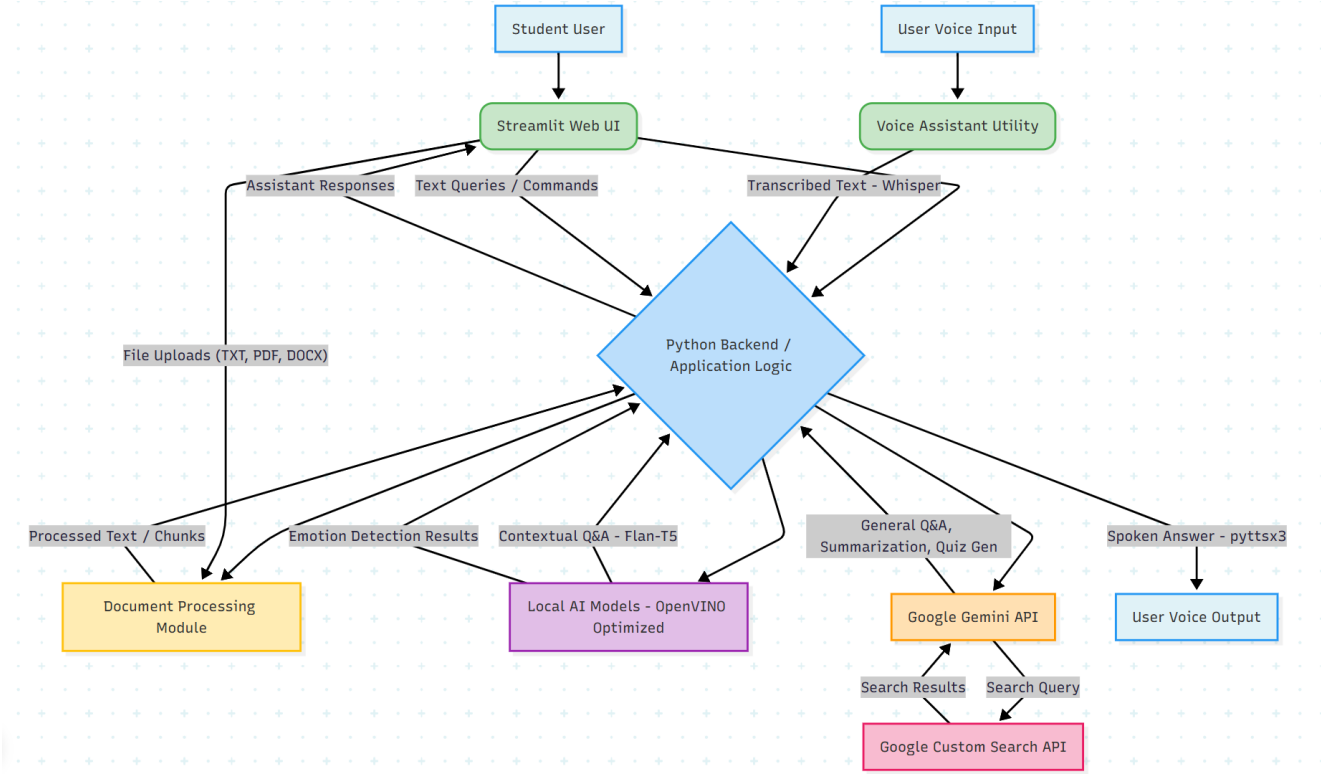
- **Contextual Question Answering (from uploaded documents):**
  - Users can upload notes in *.txt*, *.pdf,* or *.docx* formats.
  - The assistant processes these documents by chunking them into manageable sections.
  - It uses a locally optimized Flan-T5 model (via OpenVino) to answer questions directly related to the content of the uploaded notes.
  - Intelligent chunk retrieval ensures only relevant sections of the document are provided as context to the local model.
- **General Question Answering (Real-time & Conversational):**
  - For questions outside the scope of uploaded documents, or when no document is loaded, the assistant routes queries to the Gemini 2.0 Flash API.
  - The Gemini API incorporates a real-time web search tool for up-to-date information, ensuring accurate answers to current events or dynamic facts.
- **Emotion Detection:**
  - Analyses user input to detect underlying emotions, providing a layer of empathy and context to interactions. (Optimized with OpenVino).
- **Notes Summarization:**
  - Generates concise, numbered summaries of uploaded notes using the Gemini API, helping students quickly grasp key concepts.
- **Quiz Generation:**
  - Creates multiple-choice quizzes (5 to 7 questions) based on the content of uploaded notes, aiding in self-assessment and revision.
- **Important Questions/Points Extraction:**
  - Identifies and extracts most important questions from a generated quiz or crucial points from a generated summary, helping students focus their study.
  - Can also generate important practice questions directly from the full notes content.
- **Voice Input (Console-based, separate utility):**
  - A separate Python script demonstrates voice recording (using *sounddevice*), transcription (using local Whisper model), and text-to-speech output (using *pyttsx3*) for a hands-free interaction experience.
- **Optimized Performance:**
  - Local AI models (Flan-T5 for QA and emotion classifier) are optimized using OpenVINO for faster inference on Intel hardware, enhancing responsiveness.

SYSTEM ARCHITECTURE

The Smart Classroom Assistant employs a hybrid architecture combining local processing with cloud-based AI services:

- **Frontend (Streamlit UI):**
  - Built with Streamlit for a user-friendly web interface.
  - Handles file uploads, text inputs, and displays chat history and feature outputs.
- **Backend Logic (Python):**
  - Manages session state, routing logic, and orchestrates interactions between different components.
- **Local AI Models:**
  - **Flan-T5-medium:** Used for contextual Q&A from uploaded documents.
    - **Optimization:** Integrated with OpenVino for accelerated inference.
  - **Emotion Classifier (j-hartmann/emotion-english-distilroberta-base):** Used for emotion detection.
    - **Optimization:** Integrated with OpenVino for accelerated inference.
  - **Whisper-base (for voice utility):** Used for speech-to-text transcription in the separate console-based voice assistant.
- **Cloud AI Services (Gemini API):**
  - **Gemini 2.0 Flash:** Used for general knowledge Q&A, quiz generation, summarization, and extracting important points/questions.
  - **Google Custom Search JSON API:** Integrated as a tool within Gemini for real-time web search capabilities.
- **Document Processing:**
  - *pypdf* for PDF parsing.
  - *python-docx* for DOCX parsing.
  - Custom chunking logic for efficient context retrieval for local models.

FIG: Hybrid AI Architecture of the Smart Classroom Assistant



TECHNOLOGIES USED

- **Programming Language:** Python
- **Web Framework:** Streamlit
- **AI/ML Libraries:**
  - Hugging Face *transformers* (for Flan-T5 and emotion classifier)
  - *optimum-intel* & *openvino* (for model optimization)
  - *openai-whisper* (for speech-to-text in voice utility)
  - Google Gemini API (via requests)
- **Audio Processing:**
  - sounddevice (audio recording)
  - numpy (array and signal processing)
  - scipy.io.wavfile (audio file handling and saving)
- **Text-to-speech:** *pyttsx3* (for voice utility)
- **Document Parsing:**
  - *pypdf*
  - *python-docx*
  - os, json (file handling and data management)
- **Version Control:** Git, GitHub
- **Development Tools:**
  - Visual Studio Code (code editing and project management)
  - GitBash / Command Prompt / Terminal (command-line operations)

INDIVIDUAL CONTRIBUTIONS

This project was a collaborative effort, with each team member playing a crucial role in its successful development and delivery.

- **Harshit Ramesh Hundia- Lead Developer, AI Integrator & Project Lead**
  - **Conceptualization & Design:** Spearheaded the initial project concept, defined the core functionalities, and designed the overall hybrid AI architecture, including the intelligent routing between local and cloud models.
  - **Hybrid AI Architecture:** Developed the intelligent routing logic to seamlessly switch between local (OpenVINO-optimized Flan-T5) and cloud (Gemini API) models based on query type and document context availability.
  - **OpenVino Optimizarion:** Researched, integrated, and debugged OpenVINO for accelerating local model inference (Flan-T5 for QA and emotion classifier), significantly improving application responsiveness.
  - **API Integration:** Implemented robust integration with the Google Gemini API for general Q&A, quiz generation, summarization, and the Google Custom Search API for real-time information.
  - **Document Processing:** Developed the complete pipeline for handling diverse document formats (TXT, PDF, DOCX), including the crucial text extraction, intelligent chunking, and relevant chunk retrieval mechanisms for effective contextual Q&A.
  - **Debugging & Refinement:** Led all major debugging efforts, identifying and resolving complex issues such as API key handling, Streamlit session state inconsistencies, model inference errors, and merge conflicts, ensuring the application's stability and functionality.
  - **Project Management & Oversight:** Took primary responsibility for the project's technical direction, ensuring all components integrated seamlessly and met the defined objectives.

- **M Vishnu Teja- Documentation & Research Contributor**
  - **Technical Documentation Support:** Assisted in compiling and organizing sections of the project report, focusing on outlining the features and technologies used.
  - **Initial Research & Requirements:** Conducted preliminary research on potential features for a classroom assistant and helped in outlining initial project requirements.
  - **Resource Gathering:** Contributed to gathering and organizing external resources and sample data for testing the application's features.

- **Darshan H- Quality Assurance & Presentation Support**
    - **Feature Testing:** Performed testing of various application features, including document upload, general Q&A, and basic command execution, providing feedback on observed behavior.
    - **Usability Feedback:** Offered insights on the user interface and user experience, suggesting minor adjustments for clarity.
    - **Presentation Material Organization:** Assisted in organizing visual elements and content for the project presentation, contributing to slide layout and content arrangement.

CHALLENGES FACED AND SOLUTIONS

- **Model Selection and Hybrid Routing Strategy:**
    - **Challenge:** Initially, attempting to use a single local model (Flan-T5) for both contextual document-based Q&A and general knowledge questions resulted in suboptimal and often inaccurate answers for the latter, as its training was not geared for broad factual recall or real-time information.
    - **Solution:** A strategic decision was made to implement a hybrid model architecture. The local Flan-T5 model was specifically optimized and constrained to excel at contextual Q&A from uploaded documents, leveraging its ability to extract information from provided text. For general knowledge queries, real-time information, and complex generative tasks (like quiz/summary generation), the more powerful and broadly trained Google Gemini API was integrated, often with a web search tool. This dual-model approach significantly improved the accuracy and utility of the assistant for diverse query types.
- **Streamlit Session State Management:**
    - **Challenge:** Ensuring consistent behavior across Streamlit reruns and correctly persisting data (such as chat history, loaded notes, and application state) proved challenging, often leading to unexpected resets or data loss.
    - **Solution:** Leveraged *st.session_state* extensively and meticulously managed its updates. Careful attention was paid to how and when session state variables were initialized, modified, and cleared (e.g., when loading/removing documents or clearing chat history) to maintain a stable and predictable user experience.
- **Hybrid Model Routing Complexity:**
    - **Challenge:** Designing a reliable system to intelligently decide whether to route a user's query to a local model (for speed and privacy with document specific questions) or a a cloud model like Gemini (for general knowledge, complex generation, and web search) required iterative refinement.

    - **Solution:** Developed clear conditional logic based on several factors: the presence of a loaded document, the relevance of document chunks to the user's query, and the quality of responses from the local model. Gemini was established as the ultimate fallback for queries that the local model could not confidently address or for general knowledge questions.
- **Git and GitHub Workflow Management:**
    - **Challenge:** Initial unfamiliarity with Git commands and GitHub workflows led to common issues such as *Author identity unknown, ! [rejected] main -> main (fetch first)*, and merge conflicts *(main|MERGING state)*.
    - **Solution:** Systematically addressed each Git error. *git config --global user.email* and *git config --global user.name* were used to establish author identity. For rejected pushes, the *git pull origin main* command was used to fetch and merge remote changes before pushing. Merge conflicts were resolved by manually editing conflicting files (e.g., README.md) and then using *git add* and *git commit* to finalize the merge. This iterative process solidified understanding of version control best practices.
- **Python Dependency Management (*requirements.txt*):**
    - **Challenge:** Ensuring all necessary Python libraries were correctly listed in *requirements.txt* for easy project setup, especially when working across different environments (local vs. Canvas), and preventing the file from being empty.
    - **Solution:** Emphasized the critical importance of activating the correct Python virtual environment (e.g., *openvino_env* using source) before running *pip freeze > requirements.txt.* This ensured that all installed dependencies were accurately captured. Regular review and cleanup of the generated *requirements.txt* file were also performed to remove unnecessary packages.
- **Flan-T5 Context Window Limitations:**
    - **Challenge:** Even when used for contextual Q&A, initial attempts to use the local Flan-T5 model with large document chunks led to *Token indices sequence length* warnings and model hanging, indicating the model's limitations in processing extensive input contexts.
    - **Solution:** Implemented robust text chunking strategies to break down large documents into smaller, manageable segments. This was coupled with refined prompt engineering for the local model to ensure it focused on relevant information within its context window. A fallback mechanism to the more capable Google Gemini API was introduced for instances where the local model struggled or provided insufficient answers due to context limitations.
- **API Key Management and Security:**
    - **Challenge:** Securely handling sensitive API keys (for Gemini and Google Custom Search) for both local development and deployment within the Canvas environment required careful configuration to prevent exposure.

- **Solution:** For local development, environment variables (*os.getenv*) were used to load API keys, keeping them out of the codebase. For the Canvas environment, the system's provided global *__api_key__* variable was utilized, abstracting the key management. Warnings were also implemented to alert developers if placeholder keys were still present.

FUTURE ENHANCEMENTS

- **Enhanced Voice Integration:** Develop a custom Streamlit component for direct, live voice input within the web UI, replacing the separate console utility.
- **Personalized Learning Paths:** Integrate features to track student progress, identify weak areas based on quiz performance, and suggest personalized study plans.
- **Interactive Quiz Features:** Add functionality for users to take quizzes directly in the UI, receive instant feedback, and track scores.
- **Multi-modal Input:** Explore accepting image inputs (e.g., diagrams, handwritten notes) for analysis and Q&A.
- **Mobile Responsiveness:** Further optimize the Streamlit UI for seamless use on mobile devices.
- **Multilingual support:** Expand to multilingual support for regional languages.
- **User Authentication and Profiles:** Implement user login/registration to allow for personalized settings, save chat histories, and manage multiple sets of notes securely.
- **Integration with Learning Management Systems (LMS):** Explore APIs to connect with platforms like Moodle, Canvas, or Google Classroom to import/export notes, assignments, and potentially sync quiz results.
- **Automated Attendance Tracking:** Implement a feature for automated attendance tracking, potentially using voice recognition or facial recognition (with appropriate privacy considerations and consent) to mark student presence in virtual or physical classrooms.

CONCLUSION

The Smart Classroom Assistant successfully demonstrates the power of combining local and cloud-based AI models to create an intelligent and efficient learning tool. By providing contextual Q&A, general knowledge access, and educational utilities like quiz generation and summarization, it aims to significantly aid students in their studies. The project highlights practical applications of LLMs, speech processing, and performance optimization techniques, laying a strong foundation for future advancements in AI-powered education.

REFERENCES

1. https://console.cloud.google.com/
2. https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/download.html?PACKAGE=OPENVINO_BASE&VERSION=v_2025_2_0&OP_SYSTEM=WINDOWS&DISTRIBUTION=PIP
3. https://www.cloudskillsboost.google/
4. https://marketplace.visualstudio.com/items?itemName=GoogleCloudTools.cloudcode
5. https://git-scm.com/doc & https://docs.github.com/en/get-started/quickstart/set-up-git
6. https://docs.python.org/3/installing/index.html