**Intel Unnati® Industrial Training Program 2025**

**1. Team Details**

**Team Name:** CoreMentis

**Team Members:**

1. Priyanshi Gupta
2. Aditya Popli
3. Nishchay Gupta

**2. Theme Details**

* **Theme Number : 4**
* **Theme Name :** Build a Multimodal AI assistant for classrooms to dynamically answer queries using text, voice, and visuals while improving student engagement with personalized responses.
* **Theme Benefits:** The project is changing how classrooms will be designed and deployed, with real-time multimodal in-the-moment interactions that support their learning preferences. It provides an engaging and interactive learning experience with multimodal visuals and voice interactions while tracking student attentive engagement and providing interventions in a timely manner. The analytics gained also enables teachers to identify trends in a student's performance and confusions, allowing for a better differentiated personal pedagogical practice. With this project, it creates smarter, more personalized and inclusive learning opportunities.

**3. Idea and Approach Details**

**Solution Overview:** The solution consists of a multimodal AI teaching assistant that can process text, voice and visual queries in real-time, and creates contextual responses by using NLP, computer vision and speech processing capabilities. By using facial expression analysis, it was able to determine engagement levels and change its teaching style if engagement changed during the teaching episode. Visual-aids, gamified quizzes and personalized feedback ensures that he/she has an interactive, accessible and effective learning experience.

1. **Technical Stack:**

|  |  |
| --- | --- |
| **Component** | **Technology / Model** |
| Chatbot | Groq Llama 3 (via API) |
| Speech-to-Text | OpenAI Whisper |
| Text-to-Speech | Mozilla TTS (FastPitch), pyttsx3 |
| Image-to-Text (OCR) | EasyOCR |
| Engagement Analyzer | OpenVINO models (face, emotion, etc.), sklearn Linear Regression |
| Frontend | React, TailwindCSS |
| Backend | Flask, Python |

1. **How OpenVINO is Utilized in CoreMentis:**

* Speech-to-Text (Whisper):
  + The Whisper model runs significantly faster with OpenVINO, allowing for real time voice-to-text transcription on CPUs.
* Image-to-Text (EasyOCR):
  + OCR models run with OpenVINO to very efficiently extract text from notes and textbooks from images.
* Engagement Analyzer:
  + Models for detecting face, emotion, gaze, and posture run using OpenVINO for lag-free real-time analysis via a web cam.
* Model Optimization Scripts:
  + Custom scripts allow model conversion, quantization, and memory tuning to improve efficiency for new models.
* Unified Backend API:
  + In the backend endpoints display model status data and hardware statistics making it easy to debug, analyze performance, and ease troubleshooting.

1. **Decision Rationale:**

Assumptions:

* Classrooms will have access to basic hardware (camera, mic, and Intel AI-enabled PCs) for real-time input processing.
* Students and teachers are open to interacting with AI tools via voice, text, and visual interfaces.
* Reliable internet and software environment are available for deploying cloud-based or hybrid models
* Visual and audio data collected can be ethically used for engagement tracking and personalization.

Constraints:

* Multimodal processing (especially real-time video and speech) must operate under low-latency to avoid lag.
* Facial expression analysis may face accuracy issues due to lighting, occlusions, or cultural variations.
* System must ensure data privacy and security, especially with minors in educational environments.
* Integration with existing LMS platforms may require adherence to specific APIs and data formats.

1. **Innovation Highlights:**

* Multimodal Inquiry Processing:
  + This innovation has the capability to respond to text input, auditory input, and visual input (in real time) to create, maintain, and foster connections with students in ways essentially unheard of in traditional technology utilising human-machine interface or student engagement.
* Engagement Based Modifications:
  + The assistant uses facial expressions and tone of speech to notice when a user is disengaged, and then makes changes to its teaching method quickly, using visual representations, analogies, or gamification techniques.
* Customizable Learning Engine:
  + The assistant is able to track user behaviour over time and log a dialogue memory of all past inquiry sessions, and as a result generates individual learning trajectories, identifies zones of confusion, and adapts responses depending of previously established trajectories of success or failure.
* Visual Content Creation:
  + The assistant has the ability to create diagrams, flowcharts, or charts in an instance, using a variety of data produced by LLMs plus tools like Graphviz or OpenVINO, which help to make abstract concepts more intuitive for learners to engage.
* Plug-and-Play LMS Capabilities:
  + The assistant is easy to integrate with an LMS like Google Classroom or Moodle, providing an easy means for teachers to manage assignments, and assignments feedback, as well as data analytics on their students’ performance.

1. **Feasibility and User-Friendliness:**

The AI assistant is technically viable and utilizes reliable open-source frameworks, including Hugging Face transformers, OpenCV, Whisper and DeepFace, which will connect to Intel AI PCs built for the purpose of multi-modal processing in real-time. A modular development pathway that can grow and scale fits a phased development approach that starts with a minimally viable product (MVP) that includes text and visual inputs, eventually integrating audio and vision capabilities.

The AI assistant is easy to deploy, able to be integrated with Google Classroom, Moodle, or Canvas, and will run on-location or in cloud settings. It will require minimum teacher training due to built-in user-friendliness and intuitive hardware (webcam and mic which students are used to using). The user must be a priority and can be designed from the users' perspectives, with multiple modes of input, modes of communication, and output, as well as inclusivity in access to, multilingual capabilities, and wrap-around access; it is intended as student-facing. Teachers will have access to a dashboard including engagement analytics, and AI-supported suggestions for individual and collective student interventions.

* Technology wise feasible with trusted open-source tools and Intel AI hardware.
* Modular development and phase-based feature rollout (i.e., text gradually to visuals gradually to audio).
* Easy to implement and deploy on existing LMS platforms such as Google Classroom, Moodle etc.
* Minimal hardware requirements, just a webcam and microphone.
* Multi-modal student inputs and personalized responses to students.
* Teacher dashboard to display analytics, AI-enabled suggestions and real-time updates.
* Inclusivity & accessibility, using multiple languages and assistive tech.
* Gamified learning and experience to motivate learners with badges, leaderboards, quizzes, etc.

**Success Metrics:**

|  |  |  |
| --- | --- | --- |
| **Success Criteria** | **Metric/Indicator** | **Target Outcome** |
| Student Engagement | Avg. session time, interaction frequency | At least 25% increase in class interaction |
| Query Resolution Accuracy | AI response accuracy | ≥ 90% accurate answers to student queries |
| Personalization Effectiveness | Feedback on relevance | Average rating of 4.5/5 or higher |
| Teacher Adoption Rate | Weekly active users | At least 70% of teachers using it regularly |
| Learning Improvement | Pre- and post-assessment comparison | ≥ 20% improvement in understanding scores |
| Engagement Detection Accuracy | Accuracy in detecting confusion/disengagement | ≥ 85% detection accuracy |
| Accessibility Reach | Languages supported, assistive feature availability | Support for 3+ languages, accessibility tools |
| System Usability | System Usability Scale (SUS) score | ≥ 80 (Grade A usability rating) |
| Latency & Performance | Response time under load | Max 2 seconds latency, 99% uptime |

## Methodology/Architecture Diagram

## Comprehensive Methodology for CoreMentis Content Viewer

## System Overview: The CoreMentis Content Viewer provides users (primarily students) with an intuitive interface to search for, view, and interact with educational content (articles, images, videos) relevant to their queries. The system leverages both traditional web technologies and AI-powered backend services for content retrieval, processing, and presentation.

## User Interaction Flow

## Query Submission

## The user accesses the Content Viewer web page.

## A search bar is prominently displayed, allowing the user to enter a topic, question, or keyword.

## The user submits the query, typically by pressing Enter or clicking a search button.

## UI Feedback

## The interface provides immediate feedback (e.g., loading spinner) indicating that the request is being processed.

1. **Frontend Processing**
   * **Request Construction**
     + The frontend (built with HTML/CSS/JavaScript or React) captures the user’s query.
     + It constructs an HTTP request (usually a POST or GET) containing the query and any additional parameters (e.g., filters, user ID).
   * **API Communication**
     + The request is sent asynchronously (AJAX/fetch/XHR) to a designated backend API endpoint (e.g., /api/content/search).

1. **Backend Workflow**
   * **Request Reception & Validation**
     + The backend (Flask-based Python server) receives the request.
     + Input validation is performed to ensure the query is well-formed and safe.
   * **Content Retrieval**
     + The backend invokes a content scraper or retriever module.
     + This module may:
       1. Scrape trusted educational sources (e.g., Bing, Wikipedia, YouTube, internal databases).
       2. Use APIs to fetch structured data.
       3. Apply keyword-based or semantic search to maximize relevance.
   * Content Processing & Enrichment
     + Retrieved content is parsed and filtered:
     + Remove duplicates, irrelevant, or low-quality results.
     + Extract metadata: title, description, URL, source, media type, etc.
     + (Optional) AI/NLP modules may:
       1. Summarize lengthy articles.
       2. Extract key topics or highlights.
       3. Rank results based on relevance, recency, or user profile.
   * Response Formatting
     + The backend packages the processed content into a structured JSON response.
     + Each result includes necessary metadata for frontend rendering.
2. **Frontend Rendering**
   * **Data Reception**
     + The frontend receives the JSON response from the backend.
   * **Dynamic UI Update**
     + Results are displayed in a visually appealing, accessible format (e.g., cards, lists, grids).
     + Each item shows a title, snippet, image/video thumbnail, and a link to the source.
     + Users can interact with the results (expand details, open links, preview images/videos).
   * **Error Handling & Feedback**
     + If no results are found or an error occurs, the UI displays a helpful message.
     + Loading indicators and error states ensure a smooth user experience.
3. **Iterative Interaction**
   * Users can refine their query, apply filters, or perform new searches.
   * The system supports multiple searches in a single session, maintaining responsiveness.

1. **Advanced Features (Optional/Extension)**
   * Personalization: Tailor results based on user history or preferences.
   * Accessibility: Integrate text-to-speech for visually impaired users.
   * Analytics: Track user queries and engagement for continuous improvement.
   * Security: Sanitize inputs and outputs to prevent XSS, injection, or data leaks.
   * Caching: Implement caching for popular queries to reduce latency
   * Quality Assurance
   * Testing: Unit and integration tests for both frontend and backend.
   * Monitoring: Logging and error tracking for backend failures.
   * User Feedback: Mechanisms for users to report issues or suggest improvements.

* + 1. **Diagram**

A screenshot of a computer

AI-generated content may be incorrect.

**Team Contributions – CoreMentis**

**Priyanshi Gupta**

* **Engagement Analysis**: Led the development of the real-time engagement analyzer using advanced computer vision techniques to assess attention levels, facial cues, and user focus dynamically.
* **Frontend UI/UX**: Designed and implemented core user interface components with a strong emphasis on accessibility, intuitive navigation, and cross-platform consistency for students and teachers alike.
* **System Integration**: Orchestrated backend integration between AI modules and input-output flows, enabling cohesive and stable cross-module communication.

**Aditya Popli**

* **Multimodal Output & Generation**: Developed output modules to generate chatbot responses in multiple formats text, synthesized speech, and context-matched images.
* **Image Generation & Web Scraping:** Implemented dynamic image generation and advanced web scraping pipelines to enrich chatbot interactions with relevant visuals and contextual content.
* **Chatbot Development:** Contributed to the design and refinement of the core chatbot logic, enhancing accuracy, personalization, and contextual understanding.

**Nishchay Gupta**

* **Multimodal Input Handling**: Built robust pipelines to process and sync text, voice, and image inputs, ensuring seamless interaction across input types.
* **Context Management**: Developed mechanisms for maintaining dialogue state and interaction history, allowing personalized, context-aware conversations across sessions.
* **API Development**: Designed and implemented backend APIs to support communication between frontend interfaces, NLP engines, and visual processing modules.