EduMorph: AI-Powered Multimodal Learning Assistant

# 1. Problem Statement

Modern educational tools often rely on unimodal input (mainly text or video), which limits accessibility and engagement, especially for:  
- Students with different learning styles (visual, auditory, kinesthetic).  
- Learners with disabilities (e.g., speech or vision impairments).  
- Environments lacking consistent internet or hardware support.  
There’s a need for an adaptive, multimodal, AI-driven educational assistant that personalizes the learning experience through seamless integration of multiple input/output forms.

# 2. Target Users

- K-12 students  
- Students with disabilities  
- Rural/under-resourced learners

# 3. AI Models Used

- Whisper (speech recognition)  
- CLIP + MediaPipe Hands + Vision Transformer (gesture/image recognition)  
- GPT-4o (text understanding and generation)  
- Text-to-Speech (TTS): ElevenLabs or Azure TTS  
- Reinforcement Learning for personalization

# 4. Interaction Flow

User Input (e.g., speech + gesture)  
 ↓  
Speech → Text (Whisper)  
Gesture → Intent (MediaPipe + classifier)  
 ↓  
Text + Intent → LLM (GPT-4)  
 ↓  
LLM → Response generation + visuals + voice output  
 ↓  
Personalized recommendation → Next activity/module

# 5. Architecture Overview

User (Voice + Gesture)  
 ↓  
[1] Speech Recognition (Whisper) → text  
[2] Gesture Recognition (MediaPipe + CNN) → intent  
 ↓  
[3] NLP (LLM): Combine (text + intent) → contextual query  
 ↓  
[4] AI Response: Generate explanation → output as:  
 a) Text summary  
 b) Diagrams (using DALL·E or Matplotlib)  
 c) Audio (TTS engine)  
 ↓  
[5] Feedback Tracker: Store user interaction → update learning profile

# 6. Pseudocode

def multimodal\_learning\_assistant():  
 speech\_text = speech\_to\_text("audio\_input.wav") # Whisper  
 gesture\_intent = detect\_gesture("camera\_stream") # MediaPipe + classifier  
  
 prompt = f"The user said: {speech\_text} and gestured: {gesture\_intent}."  
   
 explanation = query\_llm(prompt) # GPT-4  
  
 if "diagram" in explanation:  
 diagram = generate\_diagram(explanation)  
  
 speech\_output = text\_to\_speech(explanation)  
  
 store\_user\_data(user\_id, prompt, explanation)  
  
 return {  
 "text": explanation,  
 "audio": speech\_output,  
 "visual": diagram  
 }