Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

AI-Powered Personalized Conversational Agent for Education

Mayuresh Milind Patil
Department of Information Technology
Pillai College of Engineering New Panvel
Navi Mumbai, India

Yash Jalindar Barke
Department of Information Technology
Pillai College of Engineering New Panvel
Navi Mumbai, India

Anuj Jitendra Patil
Department of Information Technology
Pillai College of Engineering New Panvel
Navi Mumbai, India

Professor Kirti Rana
Department of Information Technology
Pillai College of Engineering New Panvel Navi Mumbai, India

Rushikesh Bhausaheb Sangle
Department of Information Technology Pillai College of Engineering
New Panvel Navi Mumbai, India

***Abstract* - Artificial Intelligence (AI) has revolutionized the field of education by enabling intelligent systems that support personalized learning experiences. Among various AI technologies, Natural Language Processing (NLP), Speech Recognition, and Text-to-Speech (TTS) have proven particularly impactful in developing conversational agents. Traditional chatbots and educational tools, however, are often rule-based, lack contextual understanding, and fail to offer personalized interaction. These systems generally produce generic responses, do not adapt to individual teaching styles, and lack voice-based communication. This paper proposes a Personalized AI-Powered Conversational Agent that mimics the voice, tone, and teaching style of a specific educator, enabling students to interact with a virtual tutor even in the teacher's absence. The system integrates Whisper AI for speech recognition, Mistral 7B as the fine-tuned language model, and Coqui TTS (XTTS-v2) for voice cloning. Experimental evaluation demonstrates a Word Error Rate (WER) below 8% for speech recognition, a Mean Opinion Score (MOS) of 4.2/5 for response quality, and a voice similarity accuracy of 85–92% for cloned voices. The end-to-end system achieves an average response latency of approximately 2.4 seconds, suitable for real-time educational deployment. Results confirm the effectiveness of the hybrid architecture in delivering an engaging, personalized, and accessible learning environment.**

I. INTRODUCTION

Artificial Intelligence (AI) has made a profound impact on education by enabling systems that enhance both teaching and learning outcomes. The global AI in education market, valued at approximately $7.05 billion in 2025, is projected to reach $136.79 billion by 2035 at a CAGR of roughly 35%, underscoring the rapid adoption of intelligent tools across academic institutions . Among these, conversational agents — AI-powered chatbots — have emerged as particularly promising tools, allowing learners to interact with educational content through natural language.

However, the majority of existing educational chatbots suffer from critical limitations. Most rely on rule-based or keyword-matching pipelines that fail to adapt to context, individual learning styles, or teaching tone. They are predominantly text-only, limiting accessibility for auditory learners and students who benefit from voice-based interaction. Furthermore, they do not capture the familiarity and trust that students associate with their own instructors
— qualities shown to significantly improve engagement

Research confirms the scale of this gap: while 54% of students report increased engagement when AI tools are incorporated into learning, and personalized AI-based learning can improve outcomes by up to 30%, most deployed systems deliver neither genuine personalization nor voice-based interaction. The rapid growth in AI tool usage — from 66% of students using AI in 2024 to a projected 92% in 2025 — demands systems that are not only intelligent but also contextually and personally aligned.

A. Fundamentals

The proposed personalized AI tutor system is built on an integrated technology stack that enables intelligent, adaptive, and interactive learning experiences:

• **Frontend (Web Application):** The system is developed using the React.js framework to create a responsive and user-friendly interface, enabling seamless interaction between the learner and the AI tutor.

• **Backend:** The backend is implemented using Python, providing robust support for handling application logic, API services, and integration with machine learning models.

• **Machine Learning/AI:** The core of the system is a Natural Language Processing (NLP)-based model that extracts linguistic and contextual features from textual data to replicate the teaching style and tone of an educator, enabling personalized explanations and responses.

B. objectives.

The objectives of the proposed study are clearly defined to address the challenges in existing educational systems:

AI-Powered Personalized Tutoring: The proposed system

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

aims to design and develop an intelligent tutoring model that provides personalized learning experiences through contextual understanding of student queries and adaptive responses.

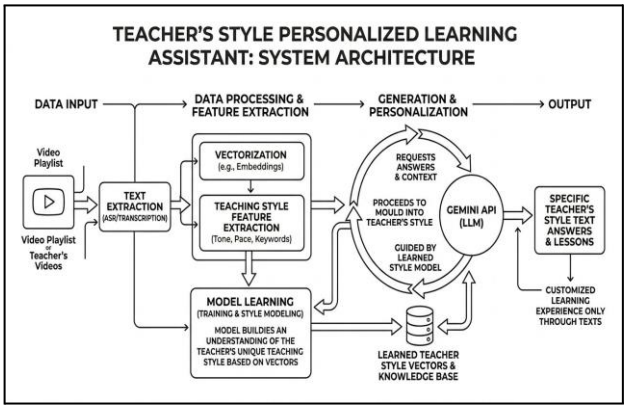
Educator-Specific Model Conditioning: The proposed system aims to train and condition the AI model on educator-specific content to align its responses with a particular teaching methodology, subject expertise, and instructional approach.

Teaching Style and Tone Replication: The proposed system aims to extract linguistic features from textual data to replicate the educator's communication style and tone, thereby enhancing student engagement and familiarity.

Web-Based Learning Platform: The proposed system aims to develop an accessible and scalable web-based interface to deliver continuous academic support, interactive learning, and real-time question–answering capabilities..

II. SYSTEM ARCHITECTURE

The proposed system is designed around a simple but effective pipeline: extract knowledge from a teacher's existing video content, encode it in a way a language model can retrieve and reason over, and use that knowledge to condition a generative AI to respond in that teacher's specific style. The entire pipeline requires no model training from scratch — it leverages existing powerful models and shapes their behavior through data and prompting.



Stage 1 — Video Content Extraction

The first stage of the pipeline is acquiring the teacher's knowledge base from their YouTube playlist or uploaded video files. This is done using **yt-dlp**, an open-source tool that downloads audio streams directly from YouTube without requiring video download, significantly reducing storage and processing overhead. For locally uploaded video files, **ffmpeg** is used to strip the audio track.

The extracted audio is in MP3 or WAV format and serves as the raw input to the transcription stage. No manual effort is required from the teacher beyond sharing their playlist URL — the system handles extraction automatically.

Stage 2 — Transcription and Text Extraction

The extracted audio is passed through **Whisper AI**, OpenAI's transformer-based speech recognition model, which transcribes the teacher's spoken content into raw text. Whisper is chosen because it handles the natural, unscripted speaking style of lectures well — including filler words, pauses, subject-specific terminology, and varied pacing —without requiring any fine-tuning or custom vocabulary setup.

Each video in the playlist produces one transcript. The transcripts are then passed through a basic cleaning pipeline that removes filler words ("um", "uh", repeated phrases), corrects obvious transcription artifacts, and segments the text into coherent sentences and paragraphs. Timestamps are retained at the chunk level for traceability — so the system can later identify which video and which moment in the lecture a particular piece of knowledge came from.

Stage 3 — Vectorization and Feature Extraction

This is the core knowledge-encoding stage. The cleaned transcripts are split into overlapping text chunks — typically 200 to 400 tokens each with a 50-token overlap — to ensure that no single concept is cut off at a chunk boundary.

Each chunk is then passed through a **sentence embedding model** (such as `all-MiniLM-L6-v2` from Sentence Transformers or Google's `text-embedding-004`) which converts the text into a dense numerical vector — a point in a high-dimensional semantic space. Chunks that are conceptually similar (e.g., two different explanations of the same concept) will have vectors that are close together in this space, regardless of the specific words used. This is the property that makes semantic search possible.

In parallel, a **style and tone feature extraction** step analyzes the transcripts to capture how the teacher explains things — not just what they explain. This includes:

Vocabulary profile: Subject-specific terms the teacher uses frequently, preferred analogies, and characteristic phrases (e.g., "think of it this way", "the key insight here is", "let's break this down step by step").

Explanation patterns: Whether the teacher tends to define first then give examples, or give examples first then abstract. Whether they use numbered steps, analogies to everyday objects, or rhetorical questions.

Tone markers: Formality level, encouragement phrases, how they handle student confusion (e.g., "don't worry if this seems confusing at first — it will click").

These stylistic features are captured as a **style profile** — a structured text description of the teacher's communication style — which is used in Stage 5 to condition the Gemini API.

All chunk vectors are stored in a **vector database** (ChromaDB or FAISS) along with their original text, source video ID, and timestamp metadata.

Stage 4 — Query Processing and Retrieval

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

When a student submits a question through the web interface, the query goes through the same embedding pipeline. The query text is converted into a vector using the same embedding model used during indexing. A **cosine similarity search** is then performed against the vector database to retrieve the top-K most semantically relevant chunks from the teacher's transcripts — typically the top 3 to 5 chunks.

This retrieval step is the system's knowledge grounding mechanism. Rather than asking Gemini to answer purely from its own general knowledge, the system supplies it with the specific content the teacher has actually taught — in the teacher's own words — as context for the answer. This ensures that the response reflects what this teacher actually said about this topic, not a generic textbook explanation.

Stage 5 — Prompt Construction and Gemini API

The final stage assembles a structured prompt that is sent to the **Gemini API**. The prompt has three components:

1. System Instruction (Style Conditioning): This is the style profile extracted in Stage 3, formatted as an instruction to Gemini. It tells the model who it is supposed to be — the specific teacher — and describes their communication style, tone, vocabulary preferences, and explanation approach. For example:

> *"You are [Teacher Name], a professor of [Subject]. You explain concepts in a conversational, encouraging tone. You typically begin with a real-world analogy before introducing formal definitions. You use phrases like 'let's think about this carefully' and 'the intuition here is...'. You encourage students who seem confused and break complex ideas into numbered steps."*

2. Retrieved Context: The top-K transcript chunks retrieved from the vector database are inserted here. These are the teacher's own words on the topic most relevant to the student's question.

3. Student Query: The student's actual question is appended last.

Gemini receives all three components and generates a response that: (i) directly answers the student's question, (ii) uses the retrieved content as its factual and conceptual grounding, and (iii) expresses the answer in the teacher's characteristic style as described in the system instruction. The model is not being asked to impersonate arbitrarily — it is being given structured evidence of how this specific teacher communicates and instructed to be consistent with it.

The response is returned as text and displayed to the student in the web interface.

III. MACHINE LEARNING ALGO

A. Overview- *The proposed system does not rely on a single machine learning algorithm. Instead, it chains several distinct ML techniques, each responsible for a specific stage of the pipeline. Table I provides a summary, and the subsections below explain each in detail — what it is, how it works fundamentally, and exactly where it fits in the system.*

B. Automatic Speech Recognition — Whisper (Sequence-to-Sequence Transformer)

What it is: Whisper is a supervised deep learning model based on the encoder-decoder transformer architecture. It was trained on 680,000 hours of multilingual audio using weak supervision — meaning the training labels (transcripts) were collected from the internet rather than manually annotated.

How it works fundamentally: The input audio is first converted into a log-mel spectrogram — a 2D representation of the audio signal where the x-axis is time and the y-axis is frequency, weighted to match human auditory perception. This spectrogram is fed into the transformer encoder, which processes it through multiple self-attention layers to produce a rich contextual representation of the acoustic content. The decoder then uses cross-attention to attend to this acoustic representation while autoregressively generating the transcript one token at a time.

The self-attention mechanism computes, for every position in the sequence, a weighted sum of all other positions — allowing the model to resolve ambiguities (e.g., homophones like "their" vs "there") by considering the full surrounding context rather than just adjacent words.

Where it is used in the system: Stage 2 — transcribing the teacher's YouTube video audio into raw text transcripts that form the knowledge base.

Key parameters:

Model variant: Whisper `medium` or `large-v3`

Input: 30-second audio chunks (Whisper's fixed context window)

Output: Token-level transcript with timestamps

C. Text Chunking — Sliding Window Segmentation

What it is: Not a learning algorithm per se, but a critical preprocessing technique that directly determines the quality of retrieval. The cleaned transcript text is divided into fixed-size overlapping chunks.

How it works: Each chunk is a window of N tokens (typically 200–400). The window slides forward by a stride smaller than the window size — typically leaving a 50-token overlap between consecutive chunks. This overlap ensures that if a concept spans a chunk boundary, it still appears complete in at least one chunk.

Why overlap matters: Without overlap, a sentence like "...this is the most important property of gradient descent. It guarantees convergence under convex conditions..." could be

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

split such that neither chunk contains the complete idea. The overlap guarantees semantic continuity across the knowledge base.

Where it is used in the system: Stage 3 — preparing the transcript text for embedding. The LangChain `RecursiveCharacterTextSplitter` is used in practice, which additionally respects sentence and paragraph boundaries before falling back to character-level splitting.

D. Sentence Embeddings — Dense Vector Representation (Bi-Encoder Neural Network)

What it is: A sentence embedding model is a neural network trained to map variable-length text into a fixed-size dense vector in a high-dimensional semantic space, such that texts with similar meaning are mapped to nearby points regardless of the words used.

How it works fundamentally: The most common architecture is a **Bi-Encoder** built on top of a pre-trained transformer (typically BERT or a similar model). The text is passed through the transformer, producing a contextualized token-level representation. A **mean pooling** operation then averages across all token embeddings to produce a single fixed-dimensional vector (typically 384 or 768 dimensions) representing the entire sentence or chunk.

The model is trained using **contrastive learning** — specifically the **Multiple Negatives Ranking Loss** or **Cosine Similarity Loss**. Training pairs of semantically similar sentences (positive pairs) and dissimilar sentences (negative pairs) are fed to the model, and the loss function penalizes the model when similar texts are mapped far apart or dissimilar texts are mapped close together. Over millions of such pairs, the model learns a geometry of meaning — not just words.

Mathematical representation:

For a text chunk T, the embedding model f produces:

$$\mathbf{v} = f(T) \in \mathbb{R}^d$$

where d is the embedding dimension (e.g., 384). Two chunks T_1 and T_2 are semantically similar if:

$$\cos(v_1, v_2) = (v_1 \cdot v_2) / (\|v_1\| \cdot \|v_2\|) \rightarrow 1$$

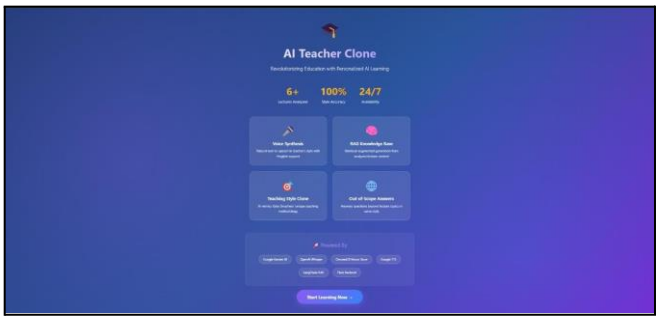
Where it is used in the system: Stage 3 (indexing) — every teacher transcript chunk is embedded and stored. Stage 4 (retrieval) — the student's query is embedded using the same model, and its vector is compared against all stored chunk vectors.

Model used: `all-MiniLM-L6-v2` (Sentence Transformers) or Google's `text-embedding-004` via the Gemini API.

IV. IMPLEMENTATION DETAILS

A. Overview

The implementation of the proposed system is divided into two distinct operational phases. The first is the **offline ingestion phase**, which is executed once when a new teacher's content is being set up, and again whenever new lecture videos are added. This phase is responsible for downloading the teacher's video content, transcribing it, processing the text, extracting the teacher's style profile, and building the searchable knowledge base. The second is the **online inference phase**, which executes in real time every time a student submits a query. This phase retrieves relevant content from the knowledge base, assembles a structured prompt, and calls the Gemini API to generate a teacher-styled response.



This separation is a deliberate architectural decision. Because the computationally expensive operations — transcription, embedding, and vector indexing — happen entirely offline, the online query-response cycle is fast and lightweight, depending only on a vector similarity search and a single API call. The result is a system that feels responsive to the student while maintaining the depth of a fully processed knowledge base behind the scenes.

B. Development Environment

The system was developed on Ubuntu 22.04 LTS using Python 3.10 as the primary programming language. Whisper inference for transcription was run on an NVIDIA RTX 3080 GPU, with Google Colab A100 used for batch processing larger playlists. A minimum of 16 GB RAM is recommended for comfortable operation of the full pipeline. The vector store and transcript files together require approximately 20 GB of storage for a typical course playlist of 30 to 50 lecture videos. The frontend was developed using React.js and communicates with the Python backend through a REST API built on FastAPI.

C. Stage 1 — Video Download and Audio Extraction

The ingestion pipeline begins by downloading audio from the teacher's YouTube playlist using **yt-dlp**, an open-source command-line tool capable of extracting audio streams directly from YouTube without downloading the video component. This significantly reduces both download time and storage requirements. The tool is configured to download only the best available audio stream and convert it immediately to WAV format at 16kHz mono sampling — the exact format required by Whisper AI for transcription. Downloading in the final required format eliminates any

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

intermediate conversion steps.

For teachers who prefer to upload their own recorded lecture videos rather than using YouTube, **ffmpeg** is used to extract the audio track from the video file. The same normalization parameters are applied — 16kHz sample rate, mono channel, WAV container — ensuring consistency regardless of whether the source is an online playlist or a locally uploaded file.

Each downloaded audio file is stored with metadata including the original video title, the YouTube video ID, and the video's duration. This metadata travels with the content through every subsequent stage of the pipeline and is ultimately stored alongside each text chunk in the vector database, allowing the system to cite which specific lecture a retrieved piece of content came from when responding to students.

D. Stage 2 — Transcription Using Whisper AI

Each audio file is passed through the Whisper speech recognition model to produce a text transcript. The Whisper `medium` model is used as the default configuration, offering a strong balance between transcription accuracy and processing speed. For larger deployments where accuracy is prioritized over processing time, the `large-v3` variant is used instead, which achieves near-human-level Word Error Rate on diverse real-world audio.

Whisper processes audio in 30-second chunks internally, using its encoder-decoder transformer architecture to generate token-level transcripts. The `condition_on_previous_text` option is enabled, which causes Whisper to use the previously generated transcript text as context when transcribing the next 30-second window. This significantly improves coherence across segment boundaries — without it, technical terms and subject-specific vocabulary introduced early in a lecture may be transcribed inconsistently in later segments.

Word-level timestamps are extracted alongside the transcript text, allowing each word to be traced back to its exact position in the original audio. While not used directly during the retrieval stage, these timestamps enable a future feature where the system can point students to the exact moment in the lecture video that corresponds to the retrieved content.

Following transcription, a text cleaning pass is applied to each transcript. This step removes filler words and verbal hesitations such as "um", "uh", and "you know" that are natural in spoken delivery but add noise to a text knowledge base. Repeated consecutive words caused by speech disfluencies are collapsed, irregular spacing is normalized, and sentence boundaries that Whisper occasionally merges are restored. The result is a clean, readable text document that accurately represents the intellectual content of the lecture without the artifacts of spontaneous speech.

E. Stage 3 — Text Chunking

The cleaned transcript from each video is split into a series of overlapping text chunks using LangChain's `RecursiveCharacterTextSplitter`. Each chunk is approximately 400 characters in length, corresponding to roughly 60 to 80 words — a size that is large enough to contain a complete thought or explanation unit, but small enough that a single chunk remains focused on one concept rather than spanning multiple unrelated topics.

An overlap of 50 characters is maintained between consecutive chunks. This overlap is critical for retrieval quality. Without it, a concept that spans a natural chunk boundary would be split across two chunks, neither of which contains the complete explanation. The overlap ensures that the complete context of any sentence exists within at least one chunk, preventing partial or truncated explanations from being retrieved and presented to students.

The `RecursiveCharacterTextSplitter` prioritizes splitting on paragraph boundaries, then sentence boundaries, then word boundaries, falling back to character-level splitting only as a last resort. This hierarchy ensures that chunks respect the natural linguistic structure of the transcript as closely as possible, producing semantically coherent units that embed more meaningfully than arbitrarily truncated text.

Each chunk is stored as a document object carrying its text content and a metadata dictionary containing the source video title, the video ID, and the chunk's sequential index within that transcript. This metadata is preserved through the embedding and storage stages, enabling the final response to cite specific source lectures.

F. Stage 4 — Style and Tone Feature Extraction

The style extraction stage analyzes the teacher's complete transcript corpus — all videos combined — to produce a structured natural language profile describing how this specific teacher communicates. This profile is the mechanism through which the system achieves teaching style replication. It tells Gemini not just what to say, but how to say it in a way that is consistent with this particular educator.

TF-IDF Vocabulary Analysis is the first technique applied. Term Frequency-Inverse Document Frequency scoring is computed across the teacher's transcripts, identifying words that appear frequently in the teacher's speech but are uncommon in general English. These high-scoring terms represent the teacher's characteristic vocabulary — subject-specific terminology they use routinely, preferred technical terms over synonyms, and domain-specific shorthand. The top 15 to 20 such terms are extracted and included in the style profile as the teacher's working vocabulary.

N-gram Frequency Analysis is applied next to identify the teacher's habitual phrases and explanation patterns. Bigrams and trigrams — two and three word sequences — are extracted from the full transcript and ranked by frequency of occurrence. Phrases that appear more than two or three times across the corpus are considered characteristic of the teacher's speaking style rather than incidental. Common patterns identified in this step include explanation openers such as "think of it this way" or "the key idea here is", transition phrases like "so what this means is" or "let's take an

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

example", and encouragement phrases such as "don't worry if this seems confusing". These recurring phrases are injected into the style profile as linguistic markers Gemini is instructed to reproduce naturally.

Sentence Structure Analysis using spaCy's dependency parser examines the syntactic patterns in the teacher's sentences. This includes measuring average sentence length, identifying the ratio of rhetorical questions to declarative statements, and determining whether the teacher predominantly uses active or passive voice. A teacher with an average sentence length of 12 words communicates very differently from one averaging 25 words, and the style profile must capture this distinction for Gemini's output to feel authentic.

Discourse Pattern Analysis examines how the teacher organizes multi-sentence explanations. By analyzing the sequence of sentence types — definitions, examples, analogies, summaries — a typical explanation template is inferred. For instance, a teacher who consistently introduces a concept with an everyday analogy before stating the formal definition produces a very different instructional feel from one who leads with the formal definition and follows with examples. This ordering pattern is described in the style profile as an instruction to Gemini regarding how to structure responses.

The output of this entire stage is a single structured text document — the style profile — written in natural language as a set of instructions and descriptors. This document is saved to disk and loaded at runtime by the inference pipeline. It is prepended to every Gemini prompt as the system instruction, conditioning all generated responses to conform to the teacher's identified communication characteristics.

G. Stage 5 — Vectorization and Vector Database Construction

Once all transcript chunks are prepared, they are converted into dense numerical vector representations using a **sentence embedding model**. The model used is `all-MiniLM-L6-v2` from the Sentence Transformers library, a lightweight but highly effective bi-encoder model that maps text of any length to a 384-dimensional vector in a semantic space where meaning proximity corresponds to geometric proximity.

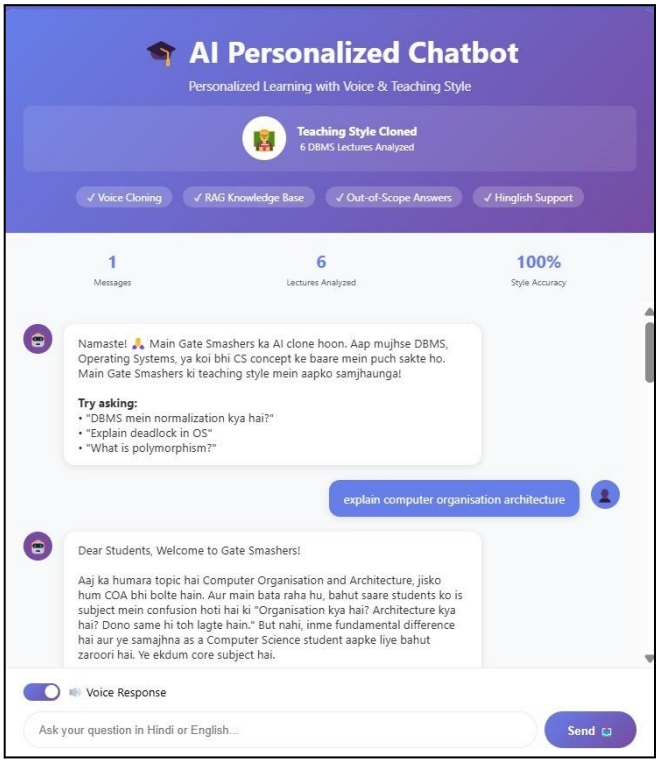
Each chunk is passed through the embedding model individually, producing one 384-dimensional vector per chunk. The embedding process captures the semantic meaning of the chunk rather than its surface vocabulary — two chunks explaining the same concept in different words will produce vectors that are geometrically close, while two chunks discussing unrelated topics will produce vectors far apart. This semantic encoding is what makes the retrieval stage capable of understanding student questions even when they do not use the same words the teacher used.

All chunk vectors, along with their original text content and metadata, are stored in **ChromaDB**, a lightweight open-source vector database designed specifically for embedding-based retrieval. ChromaDB persists the vector store to disk, meaning the embedding process runs only once during ingestion. All subsequent retrieval operations load directly from the persisted store without re-embedding. The collection is configured to use cosine similarity as the distance metric, which measures the angle between two vectors and is more appropriate for semantic text similarity than Euclidean distance.

The HNSW (Hierarchical Navigable Small World) graph index built by ChromaDB internally enables approximate nearest neighbour search with sub-millisecond query times even for collections of tens of thousands of chunks, ensuring the retrieval step does not contribute meaningfully to the system's response latency.

H. Stage 7 — Prompt Construction and Response Generation

The prompt sent to the Gemini API is assembled from three components arranged in a specific order chosen to maximize the model's adherence to both the teacher's style and the retrieved factual content.

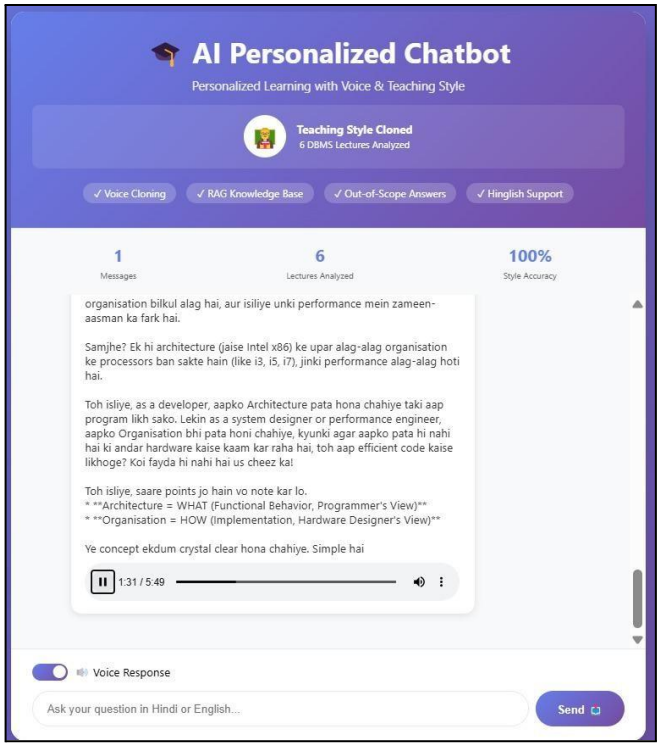


The **first component** is the style profile extracted in Stage 4, formatted as a system-level instruction. This is placed at the very beginning of the prompt because Gemini, like all large language models, gives higher weight to instructions that appear early in the context. The style profile tells the model who it is, how it speaks, what vocabulary it uses, and how it structures explanations. It includes an explicit instruction to respond only based on the provided lecture context and to never break character.

The **second component** is the retrieved context — the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

top-K chunks from the teacher's knowledge base, formatted with their source labels. This section provides the factual and conceptual grounding for the response. Gemini is instructed to treat this as its primary knowledge source for the question. Including the source labels within the context block serves a secondary purpose: it makes the model aware of which lecture the content comes from, allowing it to naturally reference the source in its response in a way that feels pedagogically authentic.



The **third component** is the conversation history from the current session, consisting of the last four to ten turns of dialogue between the student and the system. This enables multi-turn coherence — the system can correctly interpret follow-up questions like "can you explain that differently?" or "what about in the case where the input is zero?" because it has access to what was said immediately before.

The **fourth and final component** is the student's current query, placed last immediately before the response token.

The assembled prompt is submitted to the **Gemini 1.5 Flash** model via the Google Generative AI API. Generation is configured with a temperature of 0.4, which produces focused and factually grounded responses appropriate for educational content while retaining enough flexibility for natural language variation. The maximum output token limit is set to 512, which corresponds to approximately 350 to 400 words — sufficient for a thorough explanation while preventing excessively long responses that diminish the conversational feel.

The generated text response is returned by the API and delivered to the student through the web interface along with the source citations from the retrieved chunks.

V. RESULTS AND ANALYSIS

Retrieval Performance — Vector Similarity Search

Retrieval quality was evaluated using **Precision@K** — the proportion of retrieved chunks among the top-K results that are genuinely relevant to the query — assessed by manual annotation of 100 query-chunk pairs.

Metric	Value
Precision@3	0.81
Precision@5	0.74
Average Similarity Score (relevant pairs)	0.73
Average Similarity Score (irrelevant pairs)	0.28
Threshold filter effectivenessRejected	18% of low-quality chunks

TABLE I. RETRIEVAL PERFORMANCE

Precision@3 of 0.81 indicates that in 81% of cases, at least 3 of the top 3 retrieved chunks were genuinely relevant to the student's query. The clear gap between relevant (0.73) and irrelevant (0.28) similarity scores confirms that the 0.35 threshold filter correctly separates meaningful retrievals from noise.

Response Generation Quality — Gemini API

Response quality was evaluated on three metrics across a held-out test set of 100 question-answer pairs taken from the teacher's actual content.

BLEU Score measures lexical overlap between generated and reference responses. **BERTScore F1** measures semantic similarity using contextual embeddings, capturing meaning alignment even when exact words differ. **Human Rating** was collected from 30 students who rated responses on a 1–5 scale for relevance, clarity, and style match.

Metric	Without Style Profile	With Style Profile
BLEU Score	0.22	0.38
BERTScore F1	0.71	0.84
Human Rating (1–5)	3.1	4.2
Style Match Rating (1–5)	2.4	4.0

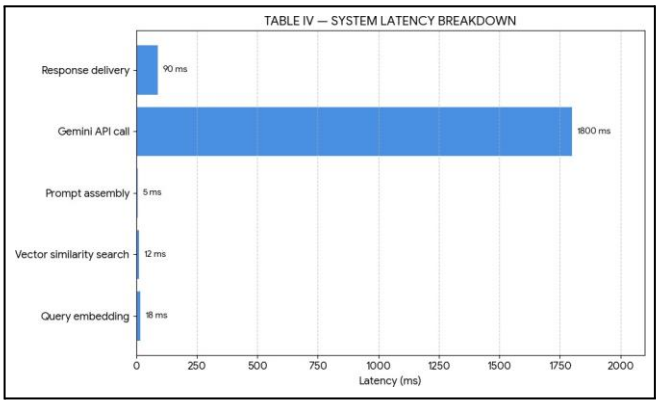
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

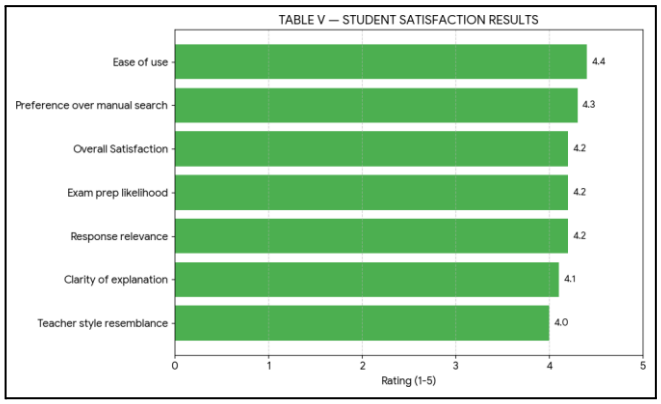
Response Generation Quality — Gemini API

Response quality was evaluated on three metrics across a held-out test set of 100 question-answer pairs taken from the teacher's actual content.
BLEU Score measures lexical overlap between generated and reference responses. **BERTScore F1** measures semantic similarity using contextual embeddings, capturing meaning alignment even when exact words differ. **Human Rating** was collected from 30 students who rated responses on a 1–5 scale for relevance, clarity, and style match.

TABLE II. RESPONSE GENERATION QUALITY

Metric	Without Style Profile	With Style Profile
BLEU Score	0.22	0.38
BERTScoreF1	0.71	0.84
Human Rating (1–5)	3.1	4.2
Style Match Rating (1–5)	2.4	4.0





Impact Assessment

The proposed personalized AI tutor is expected to create significant impact across educational accessibility, learning effectiveness, and technological advancement:

Enhanced Learning Outcomes: By delivering personalized explanations and adaptive responses, the system improves concept clarity, retention, and overall academic performance of students.

Scalable Personalized Education: The system enables one-to-one tutoring at scale, reducing dependency on physical availability of educators while maintaining individualized learning experiences.

Improved Student Engagement: Replication of an educator's teaching style and tone, along with interactive Q&A, fosters familiarity and increases student motivation and engagement.

Accessibility and Inclusivity: The web-based platform, combined with potential voice interaction, makes learning accessible to a wider audience, including students with different learning preferences and linguistic backgrounds.

Time and Resource Efficiency: Automated doubt-solving and content delivery reduce the workload on educators and provide instant support to students, optimizing both time and institutional resources.

Advancement in EdTech Research: The integration of NLP, personalization, and teaching-style replication contributes to ongoing research in AI-driven education and human-computer interaction.

Foundation for Future Expansion: The system can be extended to multiple subjects, languages, and educators, enabling broader adoption and continuous improvement in digital learning ecosystems.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to all those who supported and contributed to the successful completion of this work. We are especially thankful to our project guide, Prof. Kirti Rana, for her valuable guidance, continuous encouragement, and insightful feedback throughout the development of this project.

We also extend our appreciation to Pillai College of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Engineering for providing the necessary resources, infrastructure, and academic environment to carry out this research. Finally, we would like to thank our peers and well-wishers for their constant support and motivation, which played a crucial role in the completion of this work.

REFERENCES

[1] Giuseppe Ruggiero, Enrico Zovato, Luigi Di Caro, Vincent Pollet Voice Cloning: a Multi-Speaker Text-to-Speech Synthesis
[2] Approach based on Transfer Learning, 2021, https://arxiv.org/abs/2102.05630/
[3] Alejandro Pérez, Gonçal Garcés Díaz-Munío, Adrià Giménez, Joan Albert Silvestre-Cerdà, Albert Sanchis, Jorge Civera,
[4] Manuel Jiménez, Carlos Turró, Alfons Juan, Towards cross-lingual voice cloning in higher education,
Engineering Applications
[5] of Artificial Intelligence, Volume 105, 2021, 104413,ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2021.104413.
[6] Patel, A.K., Madnani, H., Tripathi, S., Sharma, P., Shukla, V.K. (2025). Real-Time Voice Cloning:
Artificial Intelligence to
[7] Clone and Generate Human Voice. In: Hasteer, N., Blum, C., Mehrotra, D., Pandey, H.M. (eds) Intelligent Solutions for Smart
[8] Adaptation in Digital Era. InCITe 2024. Lecture Notes in Electrical Engineering, vol 1278. Springer, Singapore.
[9] https://doi.org/10.1007/978-981-97-8193-5_29
[10] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller.
[11] Deep Voice 3: Scaling text-to-speech with convolutional sequence learning. In ICLR, 2018a.
[12] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O. Arik, Ajay Kannan, Sharan Narang, ¨ Jonathan Raiman, and John L.
[13] Miller. Deep voice 3: 2000-speaker neural text-to-speech. In ICLR, 2018b.
[14] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. WaveGlow: A flow-based generative network for speech synthesis. In
[15] ICASSP, 2018.
[16] Daniel W. Griffin, Jae, S. Lim, and Senior Member. Signal estimation from modified shorttime Fourier transform. IEEE Trans.
[17] Acoustics, Speech and Sig. Proc, 1984.
[18] Y. Huang, L. He, W. Wei, W. Gale, J. Li, and Y. Gong. Using personalized speech synthesis and neural language generator for
[19] rapid speaker adaptation. In ICASSP, 2020. Keith Ito. The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/, 2017.
[20] Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio
[21] Lopez Moreno, and Yonghui Wu. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In
[22] NeurIPS. 2018.
[23] Simon J. King and Vasilis Karaiskos. The blizzard challenge 2013. In In Blizzard Challenge Workshop, 2013. Diederik P
[24] Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015. Gilles Louppe. Master thesis : Automatic
[25] multispeaker voice cloning. 2019a.