Project Documentation: Audio/Video Search System

Project Overview

This project focuses on building a system that allows users to search through audio or video content using input in the form of text or audio. The system identifies speakers, transcribes audio into Hindi, segments the content semantically, and allows users to ask questions about the content. Key technologies used include Whisper for transcription, PyAnnote for speaker diarization, ChromaDB for semantic search and vector storage, and Retrieval-Augmented Generation (RAG) for question answering.

Technologies Used

- Whisper: Hindi speech-to-text transcription.
- PyAnnote: Speaker diarization and segmentation.
- ChromaDB: Vector database for semantic search.
- RAG (Retrieval-Augmented Generation): Question answering over audio-transcript pairs.

Assigned Tasks

Task Title	Task Description
Extract Audio from Video	Extract clear audio from video files. Ensure audio is clean, consistent, and standardized for processing.
Split Audio into Spoken Segments	Detect and segment speech parts in the audio. Output start and end timestamps for each speech segment.
Identify Which Segments Are Spoken by the Speaker	Given a sample of the speaker's voice, compare each segment. Label each segment as spoken by either the teacher or student(s).
Transcribe Spoken Segments to Hindi Text	Transcribe each audio segment into Hindi. Ensure each transcript includes timestamps and speaker label.
Prepare Segments for Semantic Search	Break transcript into searchable units (e.g., sentences). Prepare each unit for embedding or vectorization.
Enable Question-Based Search Over Transcripts	Accept a question in Hindi or English. Match it to relevant transcript units. Return the corresponding timestamp and speaker.
Design Unified Output Format	Design a consistent structure to store transcript + speaker + timestamp. Ensure this format can be used by both the search and UI layers.
Build Simple Search & Playback Interface	Create an interface to search transcripts and jump to specific times. Display

	speaker-labeled transcript alongside media playback.

Author's Role and Contribution

As the lead developer, I designed and implemented a speaker-aware Hindi transcription system that processes classroom or conversational audio. My role included extracting audio from video files, performing speaker diarization using PyAnnote, identifying speakers with Resemblyzer, slicing the audio into speaker-specific segments, and transcribing each segment into Hindi using Whisper. I structured the output into timestamped and speaker-labeled units, saved in JSON format. Additionally, I implemented sear...

Project Workflow

1. Audio is extracted from uploaded video content and standardized (mono, 16kHz).
2. The system uses PyAnnote to perform speaker diarization and generate time-aligned segments per speaker.
3. Resemblyzer is optionally used to match unknown voices to reference speakers (e.g., teacher).
4. Each segment is saved separately and transcribed into Hindi using Whisper.
5. The Hindi transcript is broken into semantic units (e.g., sentences), which are embedded and stored in ChromaDB.
6. Users can ask questions in Hindi or English. RAG retrieves the most relevant transcript segments from ChromaDB and generates an answer.
7. A unified output format stores the transcription, speaker, and timestamp, enabling both backend processing and frontend integration.
8. A user-friendly interface supports searching the transcript and playing back relevant audio segments.

Use Cases

- Educational platforms to generate labeled transcripts for lectures.
- Subtitle generation for regional language content.
- Conversational analytics for meetings, interviews, or group discussions.
- Interactive search across spoken content in Hindi using natural language.
- Real-time feedback and engagement tracking in live classrooms.

Challenges Faced

Some of the key challenges included long processing times for diarization on CPU, handling large files efficiently, aligning speaker segments with transcription output, and ensuring

Whisper's Hindi transcription was contextually accurate. Integrating multiple libraries and maintaining consistent output formatting across the pipeline also required careful planning and validation.

Future Enhancements

Planned improvements include automatic punctuation and grammar enhancement for Hindi text, subtitle file generation (.srt/.vtt), improved diarization speed using lightweight or fine-tuned models, and real-time capabilities for live classrooms. We also aim to enhance the accuracy of question answering and incorporate personalized speaker training using voice fingerprints.