Final Project Report

COMS 402 – Toast

Client: Student Project

Instructor: Dr. Simanta Mitra

Teaching Assistant: Mohammed Rahman

Team Members:

Backend

Manas Mathur

Maku Forchu

Mazin Bashier

Frontend

Luke Munn

Ezra Odole

Client Letter

Chapter 1: Requirements

Project Overview

Our team built a complete web application that uses artificial intelligence to provide automated feedback for public speaking. Users can upload or record speech videos through the system which analyzes delivery quality and emotional tone and vocal confidence and structural clarity. The platform uses Hume AI to analyze vocal emotion and intonation and OpenAI GPT to create personalized feedback and a pose estimation model to evaluate nonverbal communication.

The project aims to assist students in public speaking development through immediate accessible feedback. The platform provides an easy-to-use interface for instructors to handle classes and assess student work and monitor student development.

We also aim to provide teachers with support for grading student speeches

Functional Requirements

- Students can register for classes by entering the class code which their instructor provides
- Users can either record speech videos directly on the platform or upload existing recordings
- The backend system analyzes speech content through audio and visual feature extraction (e.g., intonation, pacing, gestures)
- The system uses Hume AI to analyze emotional and vocal signals while GPT produces delivery feedback
- The system divides speech content into Introduction, Body and Conclusion sections to provide detailed analysis for each part
- The system provides instant feedback about WPM speed and filler words and pauses and emotional delivery and structural organization to users
- The platform enables professors to establish classes while monitoring all student submissions and adding comments to specific video recordings

- Users can access their previous submissions and review all feedback they received in the past

Non-Functional Requirements

- The system needs to deliver feedback to users after they submit their videos within a time frame of 1–2 minutes.
- All personal and video data must be securely stored and protected with token-based authentication
- The application needs to be responsive and work properly on modern desktop browsers.
- The system should implement backend endpoints that handle tasks in the background to prevent blocking user interactions.
- APIs need to deliver responses that are both structured and consistent while being able to handle errors properly.
- The system needs to scale up its capabilities to handle numerous users who upload videos at the same time.
- The system needs to operate continuously during peak usage periods such as classroom assignments.

Chapter 2: Design

System Architecture

The application includes a React frontend and a FastAPI backend, with MySQL for database storage. AI services such as Hume and OpenAI GPT are used for emotion and delivery feedback. Video files are stored locally or on AWS S3. Processing occurs in the background after upload.

Frontend:

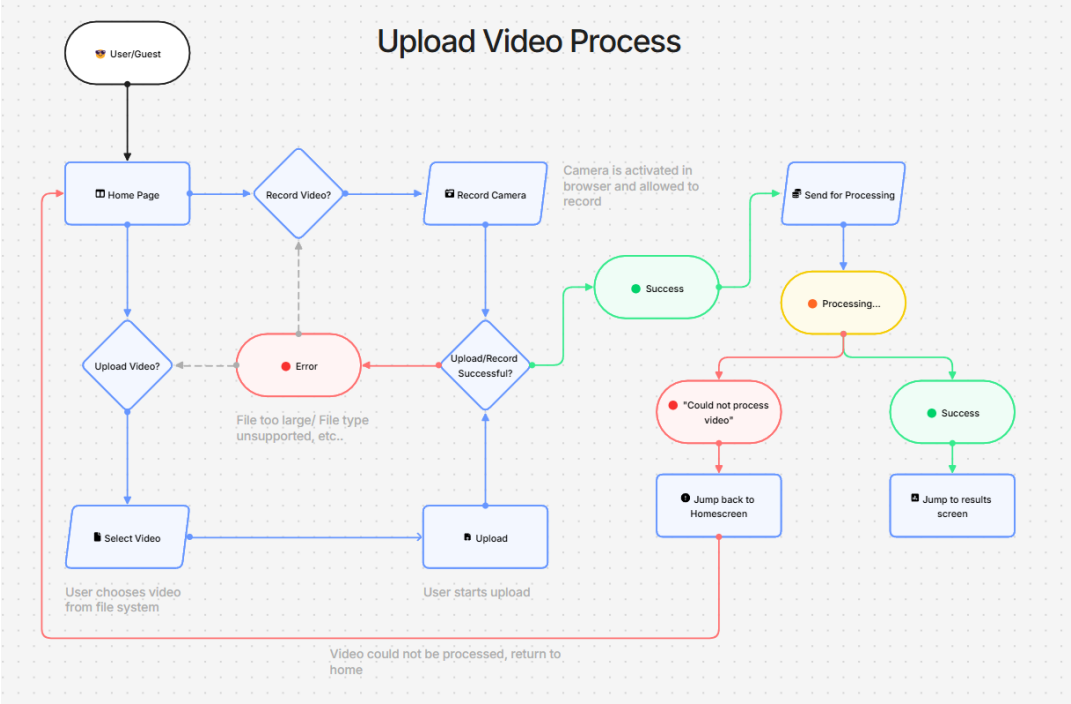

Figure 1: Upload Video Process: This flowchart illustrates the user journey for uploading or recording a video. Users can either record directly in the browser or upload a file from their device. If the upload is successful, the video is processed; otherwise, an error message appears. After processing, users are redirected to the results or back to the homepage if processing fails.

Backend:

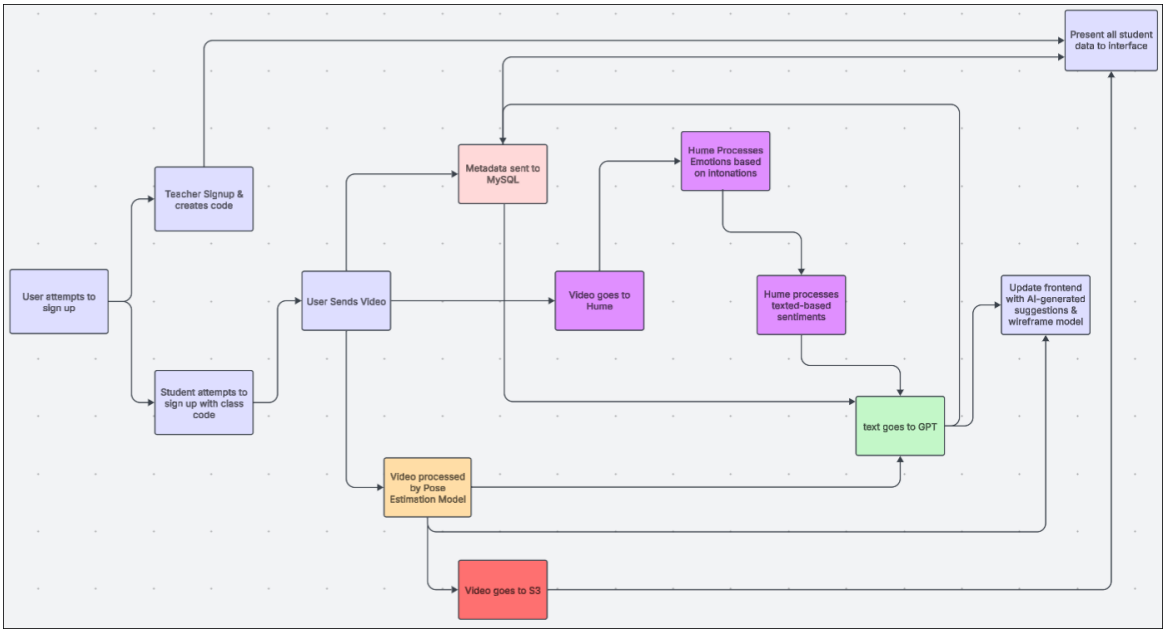

Figure 2: This diagram shows the backend pipeline from user signup to feedback generation. Uploaded videos are analyzed by Hume for emotions and by a pose model for body language. Transcripts are sent to GPT for delivery feedback. Results are stored and displayed in the frontend for students and instructors.

API Design

Teachers Endpoints:

Endpoint	Method	Description
/create_class	POST	Creates a new class
/add_comment	POST	Add comments to a video
/get_comments/{video_id}	GET	View all comments for a video

Students Endpoints:

Endpoint	Method	Description
/signup	POST	Register a new user
/login	POST	Log in and receive an access token
/submit-context	POST	Submit context (speech type, audience, message)
/submit-audio/	POST	Upload audio for analysis
/uploadVideo	POST	Upload a speech video
/uploadVideo/{class_id}	POST	Upload a video to a specific class
/join_class	POST	Join a class using a class code
/get-overview/{job_id}	GET	Get combined overview feedback
/get-hume/{job_id}	GET	Get emotion analysis from Hume
/get-gpt-feedback/{job_id}	GET	Get delivery feedback from GPT
/results/{job_id}	GET	Get full results
/results/{job_id}/video	GET	Get processed video with analysis overlay
/get-speechfiltered/{job_id}	GET	Get structured speech feedback
/view_Videos/	GET	View available videos in the class
/video_comments/{video_id}	GET	View comments on a video
/viewVideos/{class_id}	GET	View videos uploaded to a specific class

Technologies Used

- **Frontend**: React.js
- **Backend**: FastAPI (Python)
- **Database**: MySQL
- **AI**: Hume AI API, OpenAI GPT-4
- **Storage:** Local disk or AWS S3

Chapter 3: Work Done

Manas Mathur – Backend

For the project I focused on pose processing, pause estimation, and the logic between students and teachers. The computer vision model utilizes mediapipe to determine where users' are located in a video, detecting when their hands are in their pockets, when their arms are crossed, when they're picking their hair, and when they have an unfocused gaze. Each frame of the video is monitored for possible improper poses via image processing. The data is then outputted down the processing pipeline for display by the frontend. We determine when users pause & their lengths using Google's VAD, allowing us to determine when there's voice activity. This data is incorporated into Mazin's prompt to determine when the pauses are appropriate. As for the functionality between teachers and students, I created the ability for teachers to create classes and viewing the data within the classes. Added the functionality for students to join classes and upload videos for each class, where the video enters our process pipeline.

Maku Forchu – Backend

The development of our platform's core backend infrastructure was my main focus. I designed the MySQL database schema to establish table relationships which supported our user, video, class and feedback models. The backend system connected to AWS S3 through which I established the video file and cover photo storage logic in the bucket while maintaining reliable content upload and retrieval functionality for user-generated content. The system used JWT token generation to handle user authentication while I built complete sign-up and login features that included password hashing and validation and account creation functionality.

I took a central role in the video upload pipeline to ensure correct file saving and database linking of user-submitted files. I participated in debugging initial API endpoints while reviewing pull requests and worked together with the GPT and pose estimation teams to achieve seamless integration with our feedback pipeline. The backend system became stable and scalable through my work while providing support to other project components when needed.

Mazin Bashier – Backend

The development of backend systems that enable our AI feedback engine was my main focus. The application received connection to Hume AI API for emotional tone and voice intonation analysis which enabled speaker confidence and delivery trait evaluations.

The GPT prompt structure received my design input to produce customized feedback that incorporates emotional analysis and delivery quality assessment and speech content evaluation. I added functionality to determine words per minute (WPM) and detect filler word usage as part of delivery analysis improvements.

I established the logic to divide each speech into Introduction, Body and Conclusion segments which enabled us to deliver targeted feedback according to public speaking rubrics.

I assisted in developing the /results/{job_id} endpoint which merges feedback from GPT, Hume and our pose estimation model into one unified response. I collaborated with the frontend team to achieve seamless integration and uniform data presentation and participated in designing the system architecture and result presentation and backend scalability plans.

Luke Munn – Frontend

Focused mostly on the functionality of the web page, including recording/uploading video, implementing the video seekbar, setting up the skeleton formatting for the web pages, creating layout for React hierarchy, and testing and implementing the APIs to connect to backend functionality.

Specifically for API implementation, I tested and implemented the teacher and student functionality which includes packaging and sending videos as well as context information to the backend for processing, teachers adding classes, students joining classes, teachers receiving all uploaded videos from a class, students receiving all of their previous videos in their class, teachers and students reviewing each videos results, as well as teachers commenting on the results of students. Parsing all data for videos and results completed myself.

I also implemented the entirety of the teacher page functionality including displaying classes and videos from students, as well as the results page for teachers including commenting logic. Most of the results page was built by myself and has been adapted for the teacher page fully by myself and Ezra adapted it for the student page.

Ezra Odole – Frontend

The development of our platform's frontend experience fell under my responsibility. I designed the website layout to create a clean modern design which represents our brand while enhancing usability. We chose a brown color scheme for our ‘Toast’ app and I designed the entire landing page and student dashboard while creating responsive interfaces that provide easy navigation for users across different devices. We wanted to make sure that, unline workday, the UX was

intuitive and easy to use, with no excessive pages and links to click through to get to what you need. I developed the frontend authentication system which handled user registration and login functions and managed sessions through JWT tokens, as well as keeping track of the react states in the program with user data flowing in from the backend.

I established the frontend connection to backend authentication routes to create secure and uninterrupted communication between the client and server. And I provided assistance during debugging sessions while offering UI feedback across the platform and maintained consistent styling and user experience throughout all pages.

Chapter 4: Results Achieved

The project ended with the successful deployment of a fully operational AI-based public speaking feedback platform. The system enables users to complete a full roundtrip process which includes registration and speech upload followed by detailed personalized feedback generated through audio and text and visual analysis.

The following section presents the entire process along with its essential endpoints which enabled this achievement:

1. User Onboarding

- The API provides two endpoints for student and teacher authentication through /signup (POST) and /login (POST) which enable secure dashboard access. JWT-based authentication ensures that all subsequent API calls are secure.
- Teachers can establish class groups through the /create_class (POST) endpoint while students can join these groups by using their unique class codes for data organization purposes.

2. Uploading and Submitting Context

- /uploadVideo/{class_id} (POST): Students upload a recorded or pre-existing video file linked to a specific class.
- /submit-context (POST): Users submit additional speech metadata — including speech type, audience, and message — to guide GPT-based feedback.

3. Processing and AI Analysis

- The backend system starts processing videos automatically after each upload:
    - The Hume AI platform receives audio data for vocal emotion and intonation analysis to determine confidence levels and energy and mood characteristics.
    - The generated transcripts activate OpenAI GPT to produce feedback about delivery quality and structure and engagement level.
    - The pose estimation model performs video frame analysis to detect gestures and gaze direction and hand position and complete body language.
    - The VAD model determines at what sections in the video the user is speaking versus when they're taking a pause.

4. Feedback Aggregation and Retrieval

- /results/{job_id} (GET): This is the core endpoint that returns the final combined result — integrating GPT feedback, Hume emotion analysis, and pose estimation in a single structured response.
- /get-overview/{job_id}, /get-gpt-feedback/{job_id}, and /get-hume/{job_id} (GET): These endpoints allow partial result retrieval for debugging or presentation purposes.
- /get-speechfiltered/{job_id} (GET): Provides section-specific speech feedback (Introduction, Body, Conclusion) based on rubrics.

5. Result Display and Interaction

- /viewVideos/{class_id} and /view_Videos/ (GET): Allow instructors and students to browse previously submitted videos.
- /video_comments/{video_id} (GET) and /add_comment (POST): Enable teachers to view and leave written feedback on individual student videos.

End-to-End Round Trip Summary

- A student registers, joins a class, and uploads a speech video.
- The system requires contextual information to deliver personalized feedback.
- The system operates in the background to analyze the video by evaluating emotional content and delivery methods and nonverbal signals.
- The system will notify the student when the results are ready and they can access their feedback through the /results/{job_id} endpoint.
- The system allows professors to view all submissions while enabling them to add comments and monitor student performance throughout the class.

Appendix: Demo Slides

1. DEMO 1

2. DEMO 2

3. DEMO 3