A red and blue flag

Description automatically generated

24CSCI13H

Faculty of Informatics and Computer Science

Artificial Intelligence

InSightHire

Revolutionizing Candidate Assessment: A Comprehensive AI-Driven System for Video Interviews, Emotion Analysis, and Content Evaluation

By: Haneen Ahmed Qandil

Supervised by:

Professor. Andreas Pester

June 2025

# Table of Contents

[Table of Contents 2](#_Toc200708746)

[Table of Figures 8](#_Toc200708747)

[Tables 10](#_Toc200708748)

[Turnitin Report 11](#_Toc200708749)

[Acknowledgements 12](#_Toc200708750)

[1. Abstract 13](#_Toc200708751)

[2. Introduction 14](#_Toc200708752)

[2.1 Overview 14](#_Toc200708753)

[2.1.1 Contributions 16](#_Toc200708754)

[2.1.2 Research Questions 16](#_Toc200708755)

[2.2 Problem Statement 17](#_Toc200708756)

[2.3 Scope and Objectives 17](#_Toc200708757)

[2.3.1 Scope 17](#_Toc200708758)

[2.3.2 Objectives 18](#_Toc200708759)

[3. Related Work (State-of-the-Art) 19](#_Toc200708760)

[3.1 Background 19](#_Toc200708761)

[3.1.1 Speech Emotion Recognition (SER) 20](#_Toc200708762)

[3.1.1.1 Neural Architectures 20](#_Toc200708763)

[3.1.1.2 Feature Extraction 24](#_Toc200708764)

[3.1.2 Candidate Answer Evaluation 25](#_Toc200708765)

[3.1.2.1 Speech Recognition 25](#_Toc200708766)

[3.1.2.2 Large Language Models (LLMs) 26](#_Toc200708767)

[3.1.2.3 Retrieval-Augmented Generation (RAG) 28](#_Toc200708768)

[3.1.2.4 Grammar and Syntax Analysis 29](#_Toc200708769)

[3.2 Literature Review 30](#_Toc200708770)

[3.2.1 Speech Emotion Recognition (SER) 30](#_Toc200708771)

[3.2.1.1 Overview of SER Research 30](#_Toc200708772)

[3.2.1.2 Preprocessing Techniques and Feature Extraction 30](#_Toc200708773)

[3.2.1.3 Model Selection 31](#_Toc200708774)

[3.2.1.4 Summary of Key Findings 33](#_Toc200708775)

[3.2.2 Candidate Answers Evaluation 34](#_Toc200708776)

[3.2.2.1 Overview of Content Evaluation 34](#_Toc200708777)

[3.2.2.2 Speech Recognition 34](#_Toc200708778)

[3.2.2.3 Large Language Models (LLMs) for Answer Evaluation 35](#_Toc200708779)

[3.2.2.4 Retrieval-Augmented Generation (RAG) for Contextual Answering 36](#_Toc200708780)

[3.2.2.5 Grammar and Syntax Analysis 36](#_Toc200708781)

[3.2.3 Critical Methodological Analysis and Explicit Research Gaps 38](#_Toc200708782)

[3.3 Analysis of Related Work 39](#_Toc200708783)

[4. Methodology 40](#_Toc200708784)

[4.1. System Workflow 40](#_Toc200708785)

[5. Implementation 43](#_Toc200708786)

[5.1 Part I – Speech Emotion Recognition (SER) 43](#_Toc200708787)

[5.1.1 Datasets & Preprocessing 43](#_Toc200708788)

[5.1.1.1 Sources (RAVDESS, CREMA-D, TESS, SAVEE) 43](#_Toc200708789)

[5.1.1.2 Data Cleaning & Labeling 51](#_Toc200708790)

[5.1.1.3 Preprocessing Pipeline 51](#_Toc200708791)

[5.1.1.4 Data Augmentation (Noise, Stretching, Shifting, Pitch) 54](#_Toc200708792)

[5.1.2 Model Architecture & Training 56](#_Toc200708793)

[5.1.2.1 Data Splitting & Label Preparation 56](#_Toc200708794)

[5.1.2.2 CNN Architecture Overview 58](#_Toc200708795)

[5.1.2.3 Training Pipeline (Hyperparameters, Callbacks) 60](#_Toc200708796)

[5.1.2.4 Validation & Testing Procedures 62](#_Toc200708797)

[5.1.2.5 Performance Metrics & Confusion Matrix 63](#_Toc200708798)

[5.1.3 Summary 63](#_Toc200708799)

[5.2 Part II – Answer Evaluation 64](#_Toc200708800)

[5.2.1 Dataset Collection for Q&A 64](#_Toc200708801)

[5.2.2 Rubric Generation & Scoring Criteria (Technical & HR via Azure OpenAI) 65](#_Toc200708802)

[5.2.3 Retrieval & Matching Mechanism (Embeddings, FAISS, LangChain, RAG) 69](#_Toc200708803)

[5.2.3.1 Embedding Generation with AzureOpenAIEmbeddings 69](#_Toc200708804)

[5.2.3.2 FAISS Index Construction 70](#_Toc200708805)

[5.2.3.3 Orchestration with LangChain RetrievalQA 70](#_Toc200708806)

[5.2.3.4 Relevance Filtering via RAG-Style LLM Chain 71](#_Toc200708807)

[5.2.3.5 Matching Outcome & Score Integration 71](#_Toc200708808)

[5.2.4 Evaluation Pipeline (Handling “I don’t know” & Rubric-Based GPT-4o Scoring) 72](#_Toc200708809)

[5.2.5 Retrieval-Augmented Generation (RAG) Integration 73](#_Toc200708810)

[5.2.5.1 Document Embedding and Indexing 73](#_Toc200708811)

[5.2.5.2 Retrieval Step 74](#_Toc200708812)

[5.2.5.3 Augmented Prompt Construction 74](#_Toc200708813)

[5.2.5.4 Generation & Scoring 75](#_Toc200708814)

[5.2.6 Summary 75](#_Toc200708815)

[5.3 Video Interview Application Integration 76](#_Toc200708816)

[5.3.1 Project Structure Overview (Streamlit, Components, Utils) 76](#_Toc200708817)

[5.3.2 Emotion Analyzer 77](#_Toc200708818)

[5.3.3 Candidate Evaluator Workflow 82](#_Toc200708819)

[5.3.4 Audio/Video Recording Workflow 86](#_Toc200708820)

[5.3.4.1 Camera Preview Initialization 86](#_Toc200708821)

[5.3.4.2 Recording Orchestration 87](#_Toc200708822)

[5.3.4.3. Audio/Video Synchronization and Cleanup 88](#_Toc200708823)

[5.3.4.4 Question‐Specific Management and Metadata 88](#_Toc200708824)

[5.3.4.5 In-Frame Recording Feedback 89](#_Toc200708825)

[5.3.5 Transcription with Whisper 89](#_Toc200708826)

[5.3.5.1 Model Initialization 89](#_Toc200708827)

[5.3.5.2 Audio‐Only Transcription (transcribe\_audio) 90](#_Toc200708828)

[5.3.5.3 Video‐Based Transcription (transcribe\_video) 90](#_Toc200708829)

[5.3.5.4 Minimal Post‐Processing (\_clean\_transcript) 91](#_Toc200708830)

[5.3.5.5 Integration and Error Handling 91](#_Toc200708831)

[5.3.5.6 Role in the System Pipeline 92](#_Toc200708832)

[5.3.6 Grammar Checking (LanguageTool + GPT) 92](#_Toc200708833)

[5.3.6.1 Initialization and Capability Discovery 93](#_Toc200708834)

[5.3.6.2 Top-Level API: check\_grammar(text, force\_ai=False) 93](#_Toc200708835)

[5.3.6.3 Role in the Interview Pipeline 95](#_Toc200708836)

[5.3.7 Summary 96](#_Toc200708837)

[5.4 User‐Facing Functionalities 96](#_Toc200708838)

[5.4.1 Question Navigation & Session Management 96](#_Toc200708839)

[5.4.2 Recording Controls & In‐Frame Timer 97](#_Toc200708840)

[5.4.3 Question‐Specific Analysis & Results 98](#_Toc200708841)

[5.4.4 Aggregate Session Summary 98](#_Toc200708842)

[5.4.5 Practice Mode & Question Regeneration 99](#_Toc200708843)

[5.4.6 On-Demand PDF Reporting 99](#_Toc200708844)

[5.4.7 Key Front-End Highlights 100](#_Toc200708845)

[5.5 Technologies Used 100](#_Toc200708846)

[5.5.1 Streamlit, LangChain & FAISS 100](#_Toc200708847)

[5.5.2 GPT-4o & Retrieval-Augmented Generation (RAG) 102](#_Toc200708848)

[5.5.3 Whisper Speech-to-Text 103](#_Toc200708849)

[5.5.4 TensorFlow/Keras for Speech Emotion Recognition (SER) 103](#_Toc200708850)

[5.5.5 Additional Tools 104](#_Toc200708851)

[5.5.6 Summary 104](#_Toc200708852)

[5.6 Trials & Experiments 105](#_Toc200708853)

[5.6.1 SER Model Training Experiments 105](#_Toc200708854)

[Trial 1 – Transformer + LSTM (Baseline Augment-Noise) 105](#_Toc200708855)

[Trial 2 – Transformer + LSTM (Phase-Vocoder Stretch) 105](#_Toc200708856)

[Trial 3 – Transformer + LSTM (No Augment) 106](#_Toc200708857)

[Trial 4 – Transformer + LSTM (Shift-Only Augment) 106](#_Toc200708858)

[Trial 5 – Transformer + LSTM (Combined Augment: Noise + Shift) 106](#_Toc200708859)

[Trial 6: Conv1D + BiLSTM w/ ROS oversampling & focal loss 106](#_Toc200708860)

[Trial 7: Baseline BiLSTM 107](#_Toc200708861)

[Trial 8: Harmonized Labels + BiLSTM 107](#_Toc200708862)

[Trial 9: Class-Weighted BiLSTM 107](#_Toc200708863)

[Trial 10: Upsample “calm” + Focal-Loss BiLSTM 107](#_Toc200708864)

[Trial 11: Conv1D → BiLSTM + Full Upsample + Focal Loss 108](#_Toc200708865)

[Trial 12: Baseline Conv1D (80/20 Split, Minimal Dropout) 108](#_Toc200708866)

[Trial 13:Conv1D Hold-Out Validation (90/5/5 Split, Higher Dropout) 108](#_Toc200708867)

[Trial 14:Conv1D True Validation vs. Hold-Out (80/8/12 Split) 109](#_Toc200708868)

[Trial 15:Conv1D L₂-Regularization Instead of Dropout 110](#_Toc200708869)

[Trial 16: Conv1D Regularization Tuning (90/5/5 Split, Optimized Dropout) 111](#_Toc200708870)

[5.6.2 Rubric Generation Trials 112](#_Toc200708871)

[5.6.2.1 Experimental Setup 112](#_Toc200708872)

[5.6.2.2 Results & Observations 113](#_Toc200708873)

[5.6.2.3 Summary 114](#_Toc200708874)

[5.6.3 Implementation summary 115](#_Toc200708875)

[6 Testing & Evaluation 116](#_Toc200708876)

[6.1 SER 116](#_Toc200708877)

[6.1.1 Testing Methodology 116](#_Toc200708878)

[6.1.2 Evaluation Metrics & Procedures 117](#_Toc200708879)

[6.2 Candidate Answer evaluation 118](#_Toc200708880)

[6.2.1 Segmentation 118](#_Toc200708881)

[6.2.2 Feature Extraction & Augmentation 119](#_Toc200708882)

[6.2.3 Emotion Classification 119](#_Toc200708883)

[6.2.4 Aggregation & Results 119](#_Toc200708884)

[7 Results & Discussions 120](#_Toc200708885)

[7.1 SER Accuracy & Confusion Matrix 120](#_Toc200708886)

[7.2 Candidate Answer Evaluation 124](#_Toc200708887)

[7.2.1 Human Evaluations 124](#_Toc200708888)

[7.2.1.1 Participant Expectations for an Automated Interview Assessment System 124](#_Toc200708889)

[7.2.1.2 Technical Interview Question Evaluation 128](#_Toc200708890)

[7.2.1.3 HR Interview Question Evaluation 132](#_Toc200708891)

[7.2.1.4 Overall Assessment of LLM Evaluation Quality 136](#_Toc200708892)

[7.2.2 Sample Interview Sessions 140](#_Toc200708893)

[8 Conclusions & Future Work 147](#_Toc200708894)

[8.1 Summary of Achievements 147](#_Toc200708895)

[8.2 Future Enhancements 149](#_Toc200708896)

[9. Work Plan 149](#_Toc200708897)

[References 150](#_Toc200708898)

[Appendix 154](#_Toc200708899)

[Appendix A – Complete Code Listings 154](#_Toc200708900)

[Appendix B-Trials Learning Curves & Confusion Matrices 163](#_Toc200708901)

[Appendix C- Project directory layout 172](#_Toc200708902)

[Appendix D – Glossary & Abbreviations 173](#_Toc200708903)

# Table of Figures

[Figure 1-Scaled Dot Product Attention 22](#_Toc200708904)

[Figure 2-Multi-Head Attention 23](#_Toc200708905)

[Figure 3-System Workflow Diagram 40](#_Toc200708906)

[Figure 4-Datasets Emotions Distribution 49](#_Toc200708907)

[Figure 5-Rubric Generation Prompt Design 65](#_Toc200708908)

[Figure 6-Q/A detailed score 73](#_Toc200708909)

[Figure 7-evaluation prompt that interleaves the retrieved references with the candidate’s answer. 74](#_Toc200708910)

[Figure 8-Emotion analysis output structure 81](#_Toc200708911)

[Figure 9-Single Answer Rubric-Based Evaluation 83](#_Toc200708912)

[Figure 10-Audio/Video Recording Workflow 86](#_Toc200708913)

[Figure 11-Transcription with Whisper Workflow 89](#_Toc200708914)

[Figure 12-Grammar Checking (LanguageTool + GPT) Workflow 92](#_Toc200708915)

[Figure 13-Best SER Model Archeticture 116](#_Toc200708916)

[Figure 14- SER Best Model Learning Curves and Confusion Matrix 120](#_Toc200708917)

[Figure 15-Most User’s Features Chart 124](#_Toc200708918)

[Figure 16-Biggest Frustrations with Current Interview Preparation Methods Chart 126](#_Toc200708919)

[Figure 17-Trust-Building Features for users 127](#_Toc200708920)

[Figure 18- How Users see AI evaluation is correct Chart 128](#_Toc200708921)

[Figure 19-Fairness of AI’s Technical Scoring Chart 129](#_Toc200708922)

[Figure 20-Is there anything the AI missed? Chart 130](#_Toc200708923)

[Figure 21-Accuracy of AI’s HR Correctness Rating Chart 132](#_Toc200708924)

[Figure 22-Fairness of AI’s HR Scoring Chart 133](#_Toc200708925)

[Figure 23--Is there anything the AI missed in HR? Chart 134](#_Toc200708926)

[Figure 24-Calibration of Evaluations Chart 136](#_Toc200708927)

[Figure 25- Would users use such systems? Chart 137](#_Toc200708928)

[Figure 26-Users' Suggested Improvements 138](#_Toc200708929)

[Figure 27-Users' Additional Comments 139](#_Toc200708930)

[Figure 28- AI Answer Evaluation Results UI 141](#_Toc200708931)

[Figure 29-Aggregate Interview Results Summary UI 143](#_Toc200708932)

[Figure 30-Score Breakdown UI 144](#_Toc200708933)

[Figure 31- Emotion Analysis Results UI 145](#_Toc200708934)

[Figure 32- Grammar & Communication Analysis Results UI 146](#_Toc200708935)

[Figure 33- Work Plan Gannt Chart 149](#_Toc200708936)

[Figure 34- SER Data Preparation and Augmentation 155](#_Toc200708937)

[Figure 35-SER Model Definition 156](#_Toc200708938)

[Figure 36-Rubric Generation & Answer-Evaluation Pipeline 160](#_Toc200708939)

[Figure 37-Rubric Generation & Dataset Evaluation Results 161](#_Toc200708940)

[Figure 38-Retrieval & Matching Setup 162](#_Toc200708941)

[Figure 39-SER Trial 1 - Confusion Matrix 163](#_Toc200708942)

[Figure 40-SER Trial 2 - Confusion Matrix 163](#_Toc200708943)

[Figure 41-SER Trial 3 - Confusion Matrix 164](#_Toc200708944)

[Figure 42-SER Trial 4 - Confusion Matrix 164](#_Toc200708945)

[Figure 43-SER Trial 5 - Confusion Matrix 165](#_Toc200708946)

[Figure 44-SER Trial 6 - Learning Curves 166](#_Toc200708947)

[Figure 45-SER Trial 6 - Confusion Matrix 166](#_Toc200708948)

[Figure 46-SER Trial 7 - Confusion Matrix & Learning Curves 166](#_Toc200708949)

[Figure 47-SER Trial 8 - Confusion Matrix & Learning Curves 167](#_Toc200708950)

[Figure 48-SER Trial 9 - Confusion Matrix & Learning Curves 168](#_Toc200708951)

[Figure 49--SER Trial 10 - Learning Curves 168](#_Toc200708952)

[Figure 50-SER Trial 10 - Confusion Matrix 168](#_Toc200708953)

[Figure 51-SER Trial 11 -Learning Curves 169](#_Toc200708954)

[Figure 52-SER Trial 11 - Confusion Matrix 169](#_Toc200708955)

[Figure 53-SER Trial 12 - Confusion Matrix 170](#_Toc200708956)

[Figure 54-SER Trial 13 - Confusion Matrix & Learning Curves 170](#_Toc200708957)

[Figure 55-SER Trial 14 - Confusion Matrix & Learning Curves 171](#_Toc200708958)

[Figure 56-SER Trial 15 - Confusion Matrix & Learning Curves 171](#_Toc200708959)

[Figure 57-SER Trial 16 - Confusion Matrix & Learning Curves 171](#_Toc200708960)

# Tables

[Table 1-Technical Generated Rubric Examples 67](#_Toc200708961)

[Table 2-HR Generated Rubric Examples 68](#_Toc200708962)

[Table 3- Best Model Per‐class precision, recall, and F1‐scores obtained on the test set 121](#_Toc200708963)

[Table 4-Challenges and Their Impact 122](#_Toc200708964)

[Table 5- Most User's Desired Features chosen for the system 125](#_Toc200708965)

[Table 6-Biggest Frustrations with Current Interview Preparation Methods Results 126](#_Toc200708966)

[Table 7-how accurately the AI’s evaluation reflected true technical correctness 129](#_Toc200708967)

[Table 8-Fairness of AI’s Technical Scoring Results 130](#_Toc200708968)

[Table 9-Is there anything the AI missed? 131](#_Toc200708969)

[Table 10-Accuracy of AI’s HR Correctness Rating 133](#_Toc200708970)

[Table 11-Fairness of AI’s HR Scoring 134](#_Toc200708971)

[Table 12-Is there anything the AI missed in HR? 135](#_Toc200708972)

[Table 13- Calibration of Evaluations 136](#_Toc200708973)

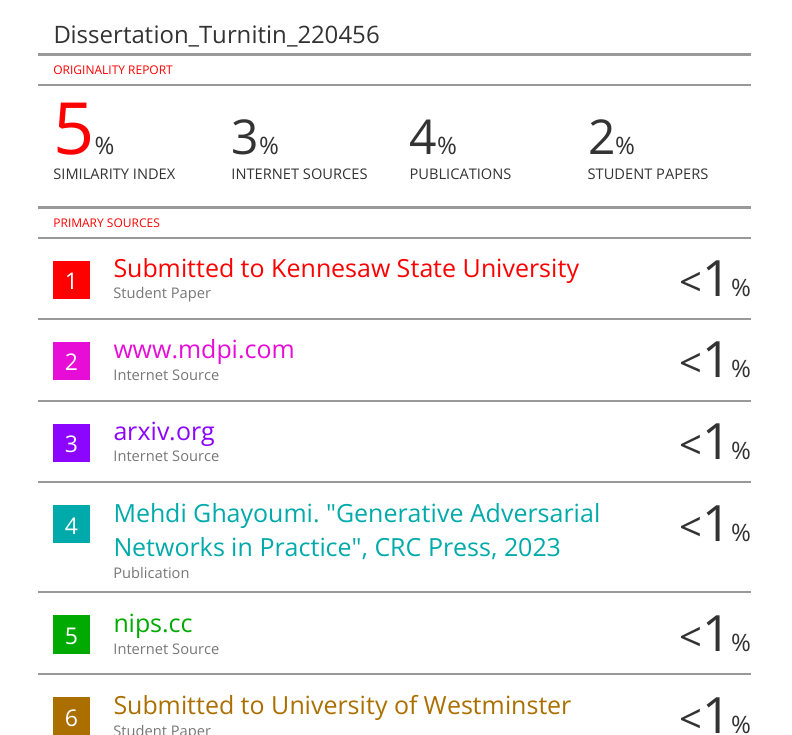
[Table 14-Would users use such systems? 137](#_Toc200708974)

[Table 15-Glossary & Abbreviations 173](#_Toc200708975)

# Turnitin Report

A screenshot of a computer screen

AI-generated content may be incorrect.



# Acknowledgements

Heartfelt appreciation is owed to **Professor Andreas Pester**, my supervisor, for his steady guidance, constructive feedback, and unfailing support at every stage of this study. Equal thanks go to **Dr. Nahla Barakat**, AI Coordinator, whose insightful counsel and continuous encouragement were essential in navigating this journey.

Sincere gratitude is extended to all the doctors and teaching assistants who generously shared their expertise and offered practical help whenever challenges occurred. Special appreciation to **Professor Omar Karam**, Dean of the Faculty of Information and Computer Sciences, and to the entire ICS faculty for responding to student needs, securing vital resources, and cultivating a culture of innovative learning.

On a personal note, my deepest thanks go to **my mother**. Her unwavering love, encouragement, and faith in my abilities provided the driving force that sustained me through the most demanding moments.

Finally, I am grateful to my friends whose steady support and optimism transformed obstacles into opportunities, making this experience both productive and memorable.

# 1. Abstract

The rapid shift to virtual interviews has created an urgent need for scalable, objective evaluation tools. AI-driven video-interview platform is presented that integrates two tightly coupled analysis pipelines, Speech Emotion Recognition (SER) and automated answer evaluation into a single reporting framework.

The SER pipeline aggregates and preprocesses audio from RAVDESS, CREMA-D, TESS, and SAVEE, extracting MFCCs, Mel-spectrograms, zero-crossing rates, and RMS energy, then applies data augmentation. A one-dimensional CNN with batch normalization, pooling, and dropout is trained on stratified splits, achieving 95.4% test accuracy across eight emotions (macro F1=0.96). Interpretability is provided via learning curve and confusion-matrix visualizations.

The answer-evaluation pipeline transcribes responses with a state-of-the-art speech-to-text model, then scores each answer using FAISS-backed retrieval and a multi-stage LLM (GPT-4o) rubric. Exact matches with prior Q/A pairs above 70% trigger a 70:30 blend of historical and fresh scores; otherwise, top 3 neighbor scores are averaged for a 30:70 blend, or default to a full LLM (GPT-4o) assessment. Rubric criteria are each prompted three times (3 LLM calls), averaged and the rationales are summarized for per-criterion scores, and then combined into a final score.

A unified reporting module merges emotional and content metrics into an interactive PDF/HTML report. Experiments on held-out responses demonstrate that combining historical data with live rubric evaluations enhances scoring consistency and fairness. Its modular design supports future extensions in multimodal candidate assessment.

# 2. Introduction

## 2.1 Overview

Candidate assessment and evaluation have traditionally involved extensive manual/human effort, often requiring multiple rounds of interviews, substantial recruiter time, and subjective evaluations to ensure the process’s accuracy and fairness. Inefficiencies, inconsistencies, and biases in evaluating candidates objectively can be challenges of the mentioned traditional approach [1]. Furthermore, increasing numbers of applicants make the traditional methods struggle more, making them insufficient for the growing demand for streamlined, unbiased, and accurate evaluation processes.

Opportunities have emerged to automate and significantly enhance candidate assessment processes with the rapid spread of artificial intelligence (AI). Scalability, fairness, and efficiency can be improved through integrating AI into recruitment [1]. Automated systems using AI can process vast amounts of data rapidly and consistently, thereby enhancing the fairness of evaluations and minimizing human biases [1].

One of the critical areas in AI-driven recruitment is Speech Emotion Recognition (SER). The underlying technology of SER provides quantitative analysis of the emotional state of the candidates in terms of confidence, stress, and enthusiasm—factors that are essential indicators of the appropriateness and effectiveness of an applicant in the respective positions to be assigned to them [2]. Based on speech descriptors like patterns, pitch, and tone, and other acoustic characteristics, the use of SER systems can provide deep insight into emotional and communication abilities and enrich traditional measures with more detailed and accurate candidate profiles [6][7].

In addition, the evaluation of the relevance and correctness of the candidates' responses is another significant challenge. Traditional evaluations relying solely on human interviewers often reflect inherent bias, unequal levels of expertise, and uneven scoring habits. Developments in natural language processing (NLP) and Large Language Models (LLMs) in recent times have overcome these challenges. Models like BERT, GPT, and LLaMA offer high capabilities in accurately understanding, contextualizing, and valuing linguistic information, which be achieved through [16][17][31]. The models can also be augmented with the use of Retrieval-Augmented Generation (RAG), which combines the generative capabilities of LLMs and retrieval of factual information to enable more accurate and contextually relevant assessments of the candidates [18][29].

Moreover, automated reporting significantly enhances recruitment efficiency. Automated systems effectively reduce the time-consuming manual documentation required in the assessment of candidates, thus enabling recruiters to be more strategically focused on decision-making and engaging with candidates rather than tedious administration tasks [12][13][14].

However, to date, there is no end-to-end AI-driven framework that jointly leverages speech-emotion recognition, LLM-based answer analysis (augmented via RAG), and automated report generation in a single, cohesive platform.

Thus, this report is motivated by the urgent need for efficient, objective, and scalable candidate evaluation systems. It investigates the integration of advanced AI technologies specifically SER, answer analysis through LLMs and RAG, and automated reporting to transform virtual interviews into a more insightful, accurate, and reliable process. The proposed Video Interview System will provide detailed, fair, and efficient feedback by exploiting these technologies, ultimately enhancing candidate preparation strategies and improving recruitment outcomes.

### 2.1.1 Contributions

This work addresses that gap by proposing and evaluating a unified video-interview system with the following contributions:

1. **Integrated Dual-Pipeline Design**  
   The first platform that fuses SER with a hybrid archival/LLM-based answer-evaluation pipeline into a unified reporting framework.
2. **Dynamic Scoring Strategy**  
   A novel scoring mechanism that blends historical Q/A performance with live GPT-4o rubric evaluations to enhance consistency and fairness.
3. **Interactive Reporting Module**  
   An automated PDF/HTML report that visualizes both emotion metrics and content scores, improving interpretability for recruiters.

### 2.1.2 Research Questions

To guide this study and validate the proposed Video Interview System, the following research questions are proposed:

1. **SER Effectiveness**
   * How accurately do the SER models classify unseen emotional speech?
   * Does augmenting training data (noise, time-stretch, pitch, shift) measurably improve generalization performance on unseen interview videos?
2. **LLM-Driven Answer Evaluation**
   * How closely do AI scores align with human ratings on technical questions?
   * How fair and comprehensive are AI scores on behavioral/HR prompts?
3. **Overall system acceptance**
   * Does the end-to-end automation reduce evaluation time and inter-rater variability compared to traditional manual processes?
   * What is the user’s willingness to adopt the integrated platform?

These questions will be addressed through quantitative experiments (Section 6) and human evaluations (Section 7), ensuring rigorous validation of both SER and answer-evaluation pipelines.

## 2.2 Problem Statement

Given an audio-visual input during an interview, create a detailed analysis of the candidate's performance regarding speech emotion recognition, content relevance, linguistic accuracy, diversity, and robustness in evaluation outputs.

## 2.3 Scope and Objectives

### 2.3.1 Scope

The Video Interview System will have three basic components: Speech Emotion Recognition (SER), content evaluation, and automated reporting. Its scope also includes real-time emotion identification through speech and analysis of response relevance and linguistic quality using LLM and RAG, respectively, and the production of detailed performance reports with visual insights. The system will also provide automated features for candidate registration, generating randomized questions, and processing responses. However, it excludes interventions by human interviewers and real-time feedback mechanisms and concentrates entirely on automated assessments and reporting.

### 2.3.2 Objectives

* **Develop an SER Module:** Accurately recognize and classify emotions from the speech of candidates.
* **Advanced Content Evaluation:** Using LLM and RAG for the evaluation of candidate answers with respect to relevance, linguistic accuracy, and contextual quality.
* **Automate Reporting:** Generate structured and visualization reports with scores from the SER combined with content assessments, including detailed feedback and graphical metrics.

# 3. Related Work (State-of-the-Art)

## 3.1 Background

Different advanced components of the SER systems have been explored for recruitment, mostly in pieces. Works on applying Transformer models and LSTM networks to significant effect are documented, along with truly advanced approaches like RAG; yet no projects exist that tie these to the vast capability of LLMs, grammar-checking technologies, and advanced report generation by way of visualization within a recruitment context. While some of these systems enhance certain facets of emotion recognition, such as accuracy and real-time processing, none have comprehensively merged these advancements to effectively address the multi-faceted challenges of SER in recruitment.

In the following sections, Transformers, LSTMs, RAG, LLMs, and grammar checking will be looked upon separately, showing their contribution and benefit to the interview system. Further, this segmented discussion will pave the way for integration in this project by knitting these elements together to make one state-of-the-art interview system specifically for the dynamic needs of modern recruitment platforms. Tapping the potentials and strengths of these methodologies combined will significantly enhance the performance of the system, making assessments from a recruitment perspective more nuanced, exacting, and insightful.

### 3.1.1 Speech Emotion Recognition (SER)

#### 3.1.1.1 Neural Architectures

##### CNN (Convolutional Neural Networks)

Convolutional Neural Networks (CNNs) work by applying convolutional filters across input data to extract features that are spatially or temporally significant. In SER, the CNNs take audio features like spectrograms to learn patterns about different emotions. These spectrograms are 2D representations of speech, where the x-axis denotes time, and the y-axis represents frequency, with the pixel intensity showing energy. The CNNs can handle that kind of input efficiently with convolutional and pooling layers, extracting local and hierarchical patterns regarding the emotional state [1][2].

The use of filters, specially developed for temporal or spectral domains, allows CNNs to detect specific patterns, such as pitch variations or harmonic tones, which represent emotional states. Advanced configurations like multi-channel spectrogram inputs, for example, log-Mel, delta, and delta-delta coefficients, increase the variety of features, while activation functions and batch normalization add to the learning process stability. Those features make the CNNs very suitable for applications requiring fine-grained recognition of emotional signals in speech [2].

##### RNNs (Recurrent Neural Networks)

RNNs are a special class of neural networks designed explicitly for the processing of sequential data and hence are very apt for applications such as Speech Emotion Recognition. RNNs are a special and distinct class of neural networks explicitly designed to handle the processing of sequential data. This native feature makes them highly applicable in various fields, including Speech Emotion Recognition. Unlike conventional feedforward neural networks, which process the input sequentially in a straight line without using any past information, RNNs have a hidden state in which they keep and store useful information from previous time steps. This, therefore, becomes an important ability because it enables RNNs to model and extract temporal dependencies that might be present in speech signals effectively and hence give better performance on the nuances of emotion understanding in speech.

**Mathematically, this relationship in an RNN** can be expressed as

**, where ​** is the hidden state at time **, ​ is the** input at time **, and** is a non-linear activation function. The **update of the hidden** state is **defined as:**

where W represents the weights matrix, b is the bias, and tanh is a standard choice for an activation function, introducing non-linearity. This update mechanism will permit RNNs to 'remember' information across time steps—a necessary element in tasks where context is important [3].

Traditional recurrent neural networks (RNNs) are good at processing patterns in sequences; however, they suffer from several disadvantages, with the most apparent being the vanishing gradient problem, which inhibits them from learning longer-term dependencies. Improved versions such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) overcome those issues by introducing additional mechanisms through which the flow of information is controlled by the forget gates and update gates. These improvements enable RNN-based systems to learn long-range dependencies better, thus being suitable for SER applications where subtle differences in emotional expression must be detected [4].

##### LSTMs (Long Short-Term Memory Networks)

Long Short-Term Memory Networks are a specific type of Recurrent Neural Network designed to handle the inherent problems with RNNs, especially the vanishing gradient problem that prevents the learning of long-term dependencies. It does make this possible by including memory blocks with each containing a cell state and three gates: input, forget, and output gates. The gates function to manage the flow of information, allowing LSTM to retain pertinent information across prolonged time sequences, while simultaneously eliminating extraneous data [5].

**The LSTM operates through a series of mathematical operations within each memory cell,** and these operations are done through these gates and cells**:** **input gate, forget gate,** **output gate**, **cell state update, and output.**

These empower LSTM to learn and remember long-term dependencies efficiently, which overcomes the traditional weakness of conventional RNNs where the gradient may not die out when learning is achieved [3]. Hence, the LSTMs become very fitting for use cases such as SER, where the emotional context can stretch across various consecutive data points [5].

##### Transformers

Transformers, as introduced by Vaswani et al. in the paper "Attention Is All You Need," revolutionized the modeling of sequences by being the first to drop recurrent architectures for self-attention mechanisms. This design enables the model to capture global dependencies across input sequences, and the high parallelization it achieves makes it more efficient than RNNs or LSTMs. Its design is in an encoder-decoder setup where the encoder maps input sequences into continuous representations while the decoder generates outputs sequentially. Layers are composed of multi-head self-attention and point-wise feed-forward networks and are stabilized by the use of residual connections with layer normalization [6].

The self-attention mechanism computes the interrelations between all elements in a sequence, focusing on relevant features regardless of their context in position. Specifically, the 'Scaled Dot-Product Attention' is defined asA diagram of a product

Description automatically generated: A black and white text

AI-generated content may be incorrect.

Figure -Scaled Dot Product Attention

where , , and are matrices representing queries, keys, and values respectively, and is the dimensionality of the keys.

Multi-head attention enhances this by running multiple attention operations in parallel, capturing diverse aspects of the sequence:

A diagram of a product

Description automatically generated

Figure 2-Multi-Head Attention

To account for the absence of recurrence, positional encodings are added to input embeddings to preserve the order of sequence through sine and cosine functions:

A mathematical equations with numbers

Description automatically generated with medium confidence

where pos is the position and is the dimension. These equations allow Transformers to have highly parallel computation, hence befitting applications such as machine translation and other tasks where long-range dependencies play a substantial role [6].

#### 3.1.1.2 Feature Extraction

##### 3.1.1.2.1 Spectrogram Analysis

###### Mel Spectrograms

Mel spectrograms are among the most common feature representations used in SER since they convey how the energy of a signal is distributed over time and frequency, highlighting those components that correspond to human audition. This starts with the Fast Fourier Transform (FFT), which decomposes the speech signal into its constituent frequencies. The resulting frequency components are mapped to the Mel scale, a perceptually motivated scale on which equal intervals correspond to equal perceived pitch differences. That results in a 2D time-frequency representation with time on the x-axis, frequency on the y-axis, and intensity represented by color [6].

###### Log-Mel Spectrograms

Log-Mel spectrograms extend this concept by applying a logarithmic transformation to the energy values in the Mel spectrogram. In this way, the dynamic range is compressed, and low-energy components become more discriminable while amplitude variations become less influential. This is very useful in SER tasks since it allows models to catch the sensitive differences in emotional speech cues [7].

##### 3.1.1.2.2 Pretrained Models

###### VGGish

VGGish is a deep learning model, complex and powerful in extracting robust features from auditory signals, which may enable significant advances in speech and audio processing. Borrowing the VGG architecture applied to the visual domain, VGGish is designed specifically for audio with an emphasis on capturing both spectral and temporal features proving essential for accurate feature representation. The architecture includes several convolutional layers, where filters are applied to the input spectrograms to transform raw audio into a representation that highlights relevant audio features [8].

The VGGish convolutional architecture would therefore allow the model to learn very generalizable and unique patterns in the audio signal, which is vital in speech enhancement and environmental sound classification tasks. The model processes input through several layers of convolutions and pooling, these operations enable the extraction of vast knowledge from the audio input, after which it is passed through fully connected layers to obtain a feature vector representing the essential information to be used in further analysis or classification tasks. The effectiveness of VGGish is justified by applying it to the MUSAN dataset, where it shows superior performance in speech quality enhancement by differentiating speech and noise components effectively. As a result, VGGish is a powerful tool for any system that requires reliable and effective audio assessment [8].

###### wav2vec 2.0

Wav2vec, particularly its iteration 2.0, has significantly advanced self-supervised learning in speech recognition. Created by Facebook AI, wav2vec uses unlabeled audio to improve the training of acoustic models. The model applies a convolutional neural network to encode raw audio into latent representations that a **Transformer-based encoder** further refines with a contrastive loss function. In this way, the loss function learns the embedding in such a way that the correct speech segment representations are good at discriminating against negative samples. Wav2vec 2.0 extends the principles with improved **quantization** and diverse **negative sampling**and hence is competent in performing tasks such as ASR by reducing the dependency on labeled data [9].

### 3.1.2 Candidate Answer Evaluation

#### 3.1.2.1 Speech Recognition

Whisper is an open-source, transformer-based, sequence-to-sequence framework built by OpenAI for multilingual speech recognition and related applications. It employs an extensive supervised training methodology with 680,000 hours of diverse audio data collected from the internet, covering a wide range of languages and acoustic environments. The model has an encoder-decoder transformer architecture: the encoder takes in input audio represented as **log-Mel spectrograms**, while the decoder generates textual output based on audio-conditional language modeling. This design allows a single instance of Whisper to perform robust transcription, translation, language identification, and voice activity detection without fine-tuning [10].

On the **encoder** side of the Whisper architecture, there are **convolutional layers** with positional embeddings that extract time- and frequency-related features from the speech data. The decoder is controlled by using multitask-specific tokens, which guide the model in transcription, translation, and other tasks. Probably most importantly, Whisper makes a special name for itself by being capable of handling both multilingual and multitask objectives simultaneously, using its scale to zero-shot generalize to datasets that have never been seen before. The robustness of a system is strengthened by extensive filtering and preprocessing, in which low-quality, machine-generated transcripts are removed, and improvements are made in the alignment between audio and text. By integrating task specification directly into its token-based architecture, Whisper avoids the normally complicated pipeline associated with speech processing [10][11].

#### 3.1.2.2 Large Language Models (LLMs)

##### GPT

Models like GPT-4, which are based on GPT, use **transformer** architectures to generate coherent and contextually appropriate responses for a large number of tasks falling under NLP. First, pre-trained on large datasets, the GPT models already have a deep understanding of linguistic patterns and world knowledge, further refined by fine-tuning and reinforcement learning with human feedback. That way, it would allow the GPT models to respond accurately in more complex situations as well [12][13].

In the specific tasks of answer scoring, GPT models outperform by bringing together contextual embeddings and attention mechanisms for estimating the similarity between questions and responses generated. It examines the generated answer against reference responses using a number of metrics, such as **BLEU**, **ROUGE**, and more complex embedding-based scorings like **BERTScore**. Moreover, GPT can integrate criteria that mimic human judgment in the evaluation of the model through constant analysis of things like relevance, accuracy, and coherence. This makes GPT suitable not just for generating high-quality responses but also for evaluating the outputs of other systems in **question-answering** architectures [14][15].

##### BERT

BERT revolutionized NLP with the introduction of a **transformer-based bidirectional model**, capturing contextual relationships between words. Unlike the unidirectional models that process sequences left to right or right to left, BERT uses a Masked Language Model (MLM) during pretraining. This is accomplished through random masking of the words in the input text and training the model to predict those masked words based on context. As such, BERT can learn a more holistic understanding of the semantic relations in a sequence by capturing both the previous and subsequent contexts simultaneously [16].

Pre-training involves two tasks: MLM, training for contextual word representation, and NSP—Next Sentence Prediction, training the model to understand the relation between paired sentences. These tasks allow BERT to generalize very well to a wide range of NLP tasks with minimal fine-tuning. Fine-tuning involves adding task-specific layers and training the model on labeled datasets, doing such tasks as question answering, text classification, and sentiment analysis. The flexibility and bidirectionality of BERT have enabled state-of-the-art results for a variety of NLP tasks [16].

##### LLaMA 2

LLaMA 2, from **Meta's GenAI** team, is a large language model (LLM) that attains leading performance via innovations in architecture and fine-tuning. The model spans a parameter range from 7 billion to 70 billion, so it promises adaptability across a variety of computational environments. One of the significant characteristics of LLaMA 2 is the adoption of transformer architecture, which heavily depends on grouped-query attention (GQA) to enhance inference scalability, especially among the larger variants. This attention mechanism is essential for the high-performance processing of large-scale data. This model has been developed to decrease the computational burden and increase the efficiency of training on large datasets. Second, LLaMA 2 uses a cosine learning rate schedule, with the initial setups being carefully tuned to ensure the best convergence over all iterations of training.

The adaptation of the learning rate is used to avoid some common problems that happen during the training phase of deep neural networks in order to avoid slow convergence and getting stuck into local minima. Compared with the LLaMA 2 architecture, it is not only capable of controlling computational requirements but also generalizes well to different tasks while maintaining a strong structure against common issues like overfitting and skewness of data [17].

#### 3.1.2.3 Retrieval-Augmented Generation (RAG)

RAG combines large-scale language models with retrieval-based approaches, efficiently performing knowledge-intensive NLP tasks. Traditional models from the past used only internal parameters to carry out a certain task, while RAG takes into consideration both the parametric and non-parametric memory. It contains the parametric memory of a pre-trained seq2seq model and also the non-parametric memory of a dense vector index of text documents, like Wikipedia, accessed through a pre-trained neural retriever. This configuration enhances the model's capacity to produce knowledgeable answers conditioned on a much broader knowledge base without the need for enormous retraining.

The RAG model works by retrieving relevant documents using a query encoder to create embeddings for input queries and then matching these to document embeddings using a method called Maximum Inner Product Search (MIPS). The retrieved documents serve a contextual purpose in the seq2seq model, which generates the output. **Specifically, the model operates with the following key equations:**

1. **Retrieval (Dense Passage Retriever or DPR):** .Here, and are the query and document embeddings, respectively.
2. **Generation (using a BART model):** . This formulation allows the model to generate each token yiy\_iyi​ conditioned on the input , the retrieved documents zzz, and the previously generated tokens .

RAG can be trained end-to-end, with the documents retrieved considered latent variables, thus maximizing the likelihood of generating the right output and enabling the creation of accurate and contextually rich content. This two-pronged approach of retrieval and generation allows RAG to dynamically adapt to new information, making it a powerful framework for a wide array of applications in which up-to-date knowledge is vital [18].

#### 3.1.2.4 Grammar and Syntax Analysis

**LanguageTool**

LanguageTool is a rule-based proofreading tool that checks written text for grammatical, stylistic, punctuation, and spelling errors. It follows a hybrid approach with rules implemented in XML as well as in Java. Basic checking is done through XML rules; these rules handle common grammatical mistakes, while more complex issues are handled in Java. Additionally, the community's input is incorporated into LanguageTool, which also taps into open resources, thus making continuous improvement possible and adaptable to several languages. It supports multiple languages by maintaining different rules specific to each language, which can handle the syntactic and lexical features unique to each linguistic system [19].

**AI-Enhanced Grammar Correction Tools**

AI-driven grammar correction applications use complex algorithms, deep learning methodologies, and natural language processing capabilities to discover and rectify grammatical inaccuracies. The applications assess the contextual framework of sentences with the view of giving personalized feedback and suggestions for improvements. These AI systems learn from large training data sets to enable the recognition of complex grammatical structures and subtle errors, increasingly with time to adapt to a writer's writing style. The adaptive learning features make them able to tailor their feedback according to individual progress and specific needs, which helps to enhance both the learning experience and the overall quality of writing [20].

**Automated Grammar Checking Techniques**

Automated grammar checking evolved to introduce the application of very complex algorithms that analyze English text with the aim of detecting and correcting grammatical errors. Methodologies used in this area range from traditional rule-based systems to state-of-the-art statistical models and machine-learning approaches. Statistical methods can involve error corpora in formulating the rules, which automatically identify frequent errors made by users. Machine learning methods, especially neural networks, use annotated corpora to determine error patterns and suggest suitable corrections. These systems enhance their performance over time by learning from the adjustments made during interactions with humans and hence efficiently reducing error rates for real-time applications [21].

## 3.2 Literature Review

### 3.2.1 Speech Emotion Recognition (SER)

#### 3.2.1.1 Overview of SER Research

Speech Emotion Recognition (SER) is a subfield of artificial intelligence with a focus on processing audio signals to recognize and classify human emotions for improved human-computer interaction. In the process of recruitment, SER technologies are adopted to measure applicants' emotional levels while answering interviews, enabling a much deeper analysis of communication qualities and, most importantly, personality traits [22]. These systems have often used state-of-the-art neural architectures in the extraction of features from spectrograms by using CNNs and in capturing temporal dependencies in speech by using LSTMs [23][2]. Also, hybrid models, which merge both CNNs and the attention mechanism, have been applied to further improve accuracy and robustness, especially for real-time applications [2][24]. The incorporation of SER technologies in video interview systems facilitates the provision of automated analyses regarding candidates' emotional reactions, thereby enhancing the assessment process and meeting contemporary recruitment requirements.

#### 3.2.1.2 Preprocessing Techniques and Feature Extraction

**Togootokh et al.** stated that in SER, preprocessing is an important stage in which the Mel spectrogram generation technique is one of the most vital methods. It involves the use of the **Fast Fourier Transform (FFT)**, conversion of frequencies to the Mel scale, and generation of spectrograms that can optimally represent the audio signals for the deep learning models. Such preprocessing steps result in high-quality feature representation, which is quite necessary for SER systems [7].

**Haidy H. Mustafa et al.** emphasized the importance of preprocessing to ensure robust data quality for the machine learning models in SER. Their work has been focused on noise reduction and extraction of the relevant features as the two major steps in the pre-processing stage. These methods are therefore integral in refining the raw audio data and preparing it for analysis and classification [6].

**Mohammad Mahdi Rezapour Mashhadi and Kofi Osei-Bonsu** address the application of advanced preprocessing techniques, especially in **data augmentation**, to enhance diversity in the datasets and increase model generalizability. The proposed methodology includes pitch shifting, time stretching, and adding noise. They also extract features such as **Mel-frequency cepstral coefficients (MFCCs)**, chromograms, and spectral contrast to represent the audio signal with highly informative characteristics [25].

**Zhen-Tao Liu et al.** demonstrate the use of **log-Mel spectrograms** in SER preprocessing tasks. In their approach, with the aim of increasing feature diversity as well as dealing with the class imbalance problem, time-domain, and frequency-domain filters are utilized to ensure that the features are diverse and balanced for the training of deep neural networks, and these operations enrich the generalization ability over different emotional classes [23].

**Yingzhi Wang et al.** apply pre-trained models, such as **Wave2Vec 2.0** and **HuBERT**, to self-supervised feature extraction in SER. Their preprocessing pipeline uses these models to extract **Contextualized and robust embeddings** from the audio, which become high-quality inputs for downstream emotion recognition tasks [27].

#### 3.2.1.3 Model Selection

**CNN**

**Z.-T. Liu et al.** take convolutional neural networks (CNNs) with attention-based bidirectional long short-term memory networks (ABLSTMs) and multi-task learning for speech emotion recognition. They apply triple-channel log-Mel spectrograms, which allow for richer emotional information with balanced augmented sampling and filtering in both time and frequency domains. It effectively increases the model generalizability of SER and demonstrates the effectiveness of combining advanced CNN architectures with state-of-the-art LSTM networks to enhance feature extraction and classification accuracy [23].

**Shawn Hershey et al.** give adaptations of CNN architectures—AlexNet, VGG, Inception, and ResNet—for large-scale audio classification. They transform the audio into spectrograms and use CNNs in order to extract spatial features traditionally used in visual tasks. This study strongly advances the generalization ability of CNN to extract discriminative features from audio data for acoustic event detection and classification tasks [28].

**Dias Issa et al.** focus on the one-dimensional CNN approach to SER by using multiple features of audio, including mel-frequency cepstral coefficients, chromagram, and spectral contrast. Using these different spectral features as input for CNN tries to achieve robust classification of emotional states from raw audio data. The method puts forward the possibility of using multiple types of features to enrich the input data of the model, which could lead to better generalization and classification accuracy improvement in SER tasks [2].

**RNN**

In this paper by **Z.-T. Liu et al.**, the authors discuss the application of RNNs to the task of analyzing temporal sequences in speech data for emotion recognition. As the sequential patterns and temporal dependencies are inherent in speech, the RNNs are capable of modeling how emotions change over time. Especially, this approach accentuates the capability to understand the dynamic nature of emotional expressions because it enables investigation into how emotions unfold within an utterance. This application of RNNs in context again underlines their strength in handling time-series data and their contribution to enhance the performances of speech-emotion recognition systems [23].

**LSTM**

In a study on interview coaching systems, **Ming-Hsiang Su et al.** describe the application of LSTM networks and deep learning techniques for dialog state tracking and action selection. Their system contains LSTM to capture temporal relationships in dialogue sequences, obtaining 77.96% accuracy in the subject-independent evaluation. The application of LSTM showed it is capable of keeping context coherent and, based on a sequence of inputs, was able to predict a correct dialogue state​ [22].

Taiba Majid Wani et al. explain in detail the ability of LSTMs to handle sequential speech data, especially the high performance in modeling temporal dynamics of emotions in SER systems. They consider LSTMs as one of the integral parts of modern SER frameworks because of their combination with other deep learning models, such as CNNs for preprocessing and feature extraction [24].

Among them, the work of Dias Issa et al. focuses on the use of LSTMs in the context of multimodal SER models. The authors identify the use of LSTMs along with CNNs for feature learning of complex temporal patterns present in datasets like RAVDESS and EMO-DB. Their proposed model, while combining the feature spaces obtained from spectrograms, MFCCs, and LSTMs, yielded state-of-the-art results, which revealed that LSTMs are indeed very flexible for different emotional scenarios [2].

**Transformers**

**Yingzhi Wang et al.** talk about the application of transformer-based models, specifically wav2vec 2.0 and HuBERT, to SER tasks. These models, by unsupervised pretraining, extract robust prosodic and semantic features from speech. The main conclusion of this study reveals that partially and fully fine-tuned versions of these models significantly outperform the traditional approaches to achieve state-of-the-art performance on datasets such as IEMOCAP. Their findings showcase the transformers' ability to handle complex temporal and contextual patterns in speech data [26].

#### 3.2.1.4 Summary of Key Findings

These studies reviewed herein show much progress made in Speech Emotion Recognition, presenting good performance within a range of neural architectures using CNNs for spatial feature extraction from spectrograms and the handling of temporal dependencies in audio data using LSTMs. Models with hybrid architectures like CNN-LSTM and attention-based models have also shown good performances, achieving accuracies of 71.61% on the RAVDESS dataset, 77.96% in dialogue state tracking, and even above 95% in speaker-dependent scenarios for the EMO-DB database [2], [22], [23]. Despite these improvements, existing approaches face issues of real-time performance, flexibility with multiple languages, and robustness across different speakers and noisy conditions [25], [26]. Key gaps include the requirement for models that generalize well to diverse languages and subtle emotional nuances and perform efficiently in dynamic real-world applications. Addressing these challenges entails integrating advanced pre-trained models like wav2vec 2.0, incorporating multi-lingual datasets, and improving hybrid frameworks with respect to scalability, real-time processing efficiency, and accuracy in a number of use cases, including recruitment, and customer service [6][22][24].

### 3.2.2 Candidate Answers Evaluation

#### 3.2.2.1 Overview of Content Evaluation

Video Interview Evaluation, the review of the content concerning the quality, coherence, and relevance of the answers is a vital activity in candidate evaluation. It assesses grammatical correctness, situational understanding, and the ability to answer questions in a comprehensive manner. It will ensure that the responses of the candidates are not only well structured but also meaningful and aligned with the objectives of the interview, hence providing a more knowledgeable and unbiased evaluation process.

Advanced technologies are employed to reach this goal: **LLaMA,** standing for Large Language Model Meta AI, has high precision in semantic analysis and contextual understanding. Tools such as **LanguageTool** help with the detection of grammatical and stylistic errors; **Retrieval-Augmented Generation (RAG)** helps increase the answer relevance through a fusion of outside knowledge. All these technologies together form a coherent structure, maximizing content evaluation and enabling precise, automated scoring while offering deeper insights into the performance of candidates.

#### 3.2.2.2 Speech Recognition

**Integration of Whisper in the Project**  
 **Alec Radford et al.** introduce Whisper as an STT system that can serve all purposes, from small datasets to large ones; it focuses on providing highly accurate transcriptions. Whisper supports multilingual and multitasking, making it capable of handling challenging transcription and translation with a wide range of different applications. Its flexibility and ability to scale up make Whisper a good candidate for scenarios where precision in the transcription of audio data is needed [10].

**Antonio Bevilacqua et al.** extend Whisper's functionality by developing Whispy, a real-time adaptation of the Whisper system. Whispy focuses on the delivery of accurate, low-latency transcription of live audio streams; hence, it is suited for real-time applications like web conferencing. By using advanced preprocessing methods, Whispy makes sure that transcriptions are robust and reliable while keeping the bar high, set by Whisper, for dynamic and time-sensitive uses [28].

#### 3.2.2.3 Large Language Models (LLMs) for Answer Evaluation

**GPT**

**Yang Wang et al.** study the evaluation performance of GPT-4 in retrieval-augmented generation applications. They report that in grading answers according to their correctness, completeness, and honesty, GPT-4 agrees with humans at a rate of 83%, showcasing its reliability for evaluations on closed-domain, closed-ended questions [29].

**Zhe He et al.** present GPT-4 and other large language models that generate responses to questions about laboratory test results. The study demonstrates GPT-4 outperformed other LLMs and human responses on online platforms in terms of generating relevant, correct, helpful, and safe answers. It can thus be inferred that GPT-4 is quite useful for aligning its outputs with human evaluations, particularly when the scenarios require semantic understanding and logical reasoning [15].

**BERT**

**Jacob Devlin et al.** introduce BERT as a bidirectional transformer model, excelling in language understanding tasks due to its ability to consider both preceding and succeeding contexts within a sentence. Its application in answer evaluation leverages these capabilities to provide accurate semantic similarity scoring and relevance assessments. BERT's contextual embeddings significantly enhance textual entailment analysis and question-answering tasks by improving contextual understanding​ [16].

**Omar Khattab et al.** explore the application of BERT in information retrieval through the ColBERT framework, adapting BERT for efficient passage ranking. Late interaction mechanisms are applied within the framework to tap into the embeddings by BERT when evaluating semantic relevance between queries and documents. This shows the fine-grained contextual evaluations BERT can provide for the answer assessment tasks [30].

**LLaMA**

**Zhe He et al.** evaluate the performances of LLaMA, among other large language models, in answering questions related to laboratory test results. The paper indicates that LLaMA, with 2 trillion tokens and improved decoder-only architecture, achieves high accuracy on benchmarks of complex open-book Q&A tasks. This has placed it as a strong model for those tasks that require precision in the evaluation of answers due to its ability to generate contextually relevant and safe answers. It also indicated that LLaMA outperformed older models like GPT-3.5 in terms of relevance and correctness but was slightly outmatched by GPT-4 in certain medical applications [31].

#### 3.2.2.4 Retrieval-Augmented Generation (RAG) for Contextual Answering

**Laura Dietz** introduces a novel evaluation framework for RAG systems based on Large Language Models that operate relevance assessments. Strong emphasis is given to nugget-based and question-based evaluation paradigms, whereby system responses are assessed for their ability to address key information nuggets or answer specific questions. The presented framework supports the integration of generative and retrieval systems; its focus is to provide an improvement in answer evaluation metrics by alignment in terms of relevance. It indicates the ability of RAG systems to adapt generated responses to contextual queries [14].

**Negar Arabzadeh et al.** explore the use of retrieval benchmarks, such as TREC Deep Learning datasets, to evaluate generated answers in RAG systems. Their approach measures the similarity between generated answers and retrieved passages using embedding-based methods like **cosine similarity**. This allows for a unified evaluation of generative and retrieval-based models, ensuring that generated responses align with judged relevant passages from traditional retrieval benchmarks. Their findings underscore the potential of RAG systems to merge generative capabilities with retrieval effectiveness​ [12].

#### 3.2.2.5 Grammar and Syntax Analysis

**Madhvi Soni and Jitendra Singh Thakur** have done a great job summarizing different approaches used for grammar and syntax error detection. They classify techniques into three main categories: rule-based, machine learning-based, and hybrid methods. Rule-based systems depend on manually created grammar rules; this approach is quite effective but requires extensive linguistic expertise and is difficult to maintain. Machine learning approaches, on the other hand, make use of annotated corpora to identify errors on a statistical basis, providing scalability that, however, strongly relies on the quality and the size of the training data. The hybrid approach combines the strengths of the rule-based and machine learning systems, attaining better accuracy in error detection. The paper also develops a classification scheme for grammar errors, distinguishing them based on error frequency, textual validity, error level, and nature, to provide a systematic framework for understanding grammar and syntax issues [32].

**Cano et al.** have underlined that most of the state-of-the-art techniques of deep learning can be used in detecting grammatical errors. In this direction, LSTM-CRF methods and neural networks are applied so as to define syntactic divergences with high accuracy, with the use of contextualized features. In addition, the Noisy Channel Model and Word2Vec embeddings are employed to better understand word interdependencies in order to find grammatical and syntactical errors correctly. Corpus-based approaches like BoW help in finding patterns and classifying the errors. The authors also note that a well-structured training dataset, such as learner corpora, plays an important role in improving model performance, especially for academic writing [21].

**Dr. C. Priya and Prof. Dr. R. Vijayalakshmi** examine the contribution of artificial intelligence-powered applications to the identification of grammar and syntax errors. Indeed, both Grammarly and ProWritingAid, through the integration of neural network technology combined with advanced natural language processing methodologies, provide real-time, context-sensitive suggestions. These tools have indeed shown a high degree of competence in the detection of syntactic-related errors, such as incorrect sentence structures and subject-verb agreement errors, and give users detailed feedback. The adaptive learning capabilities of these systems allow them to provide immediate error corrections tailored to individual needs, which increases user engagement and accuracy in writing [20]

**Marcin Miłkowski** describes the design of **LanguageTool**, a rule-based grammar-checking tool. The research highlights the effectiveness of positive-match checkers relying on declarative rules in detecting grammatical and syntactical errors. The approach involves shallow parsing to identify common patterns in linguistic mistakes and hence does not require deep parsing, therefore simplifying the processing task. The tool is intended to cover frequent syntactical errors like subject-verb discord and word order errors by simple rules, which are easy to maintain. Miłkowski points out that such an approach is especially helpful for under-resourced languages for which annotated corpora do not exist [19].

### 3.2.3 Critical Methodological Analysis and Explicit Research Gaps

Although deep CNN, RNN/LSTM, and transformer-based architectures each advance SER accuracy, none addresses all key challenges alone. CNNs capture local spectral features well but cannot model long-range prosodic patterns; RNNs and LSTMs recover temporal dynamics at the cost of vanishing gradients, overfitting on small corpora, and slower inference; transformers and self-supervised models (wav2vec 2.0, HuBERT) improve data efficiency yet demand massive unlabelled data and compute resources, hindering deployment in constrained environments. Preprocessing and augmentation techniques (noise injection, time-stretch, pitch shifts) offer modest (≈ 5 %) generalization gains but often leave minority classes (e.g., “calm”) under-represented and untested on truly unseen accents or recording conditions.

In answer evaluation, static rubrics drawn from generic criteria achieve consistency but fail to capture question-specific nuances, leaving roughly one-third of prompts without proper rubric coverage. Retrieval-augmented scoring adds factual grounding yet complicates the blending of historical versus fresh evaluations.

Crucially, no existing work combines SER, dynamic rubric generation, answer scoring, and automated reporting into a unified pipeline. This forces practitioners to stitch together separate tools for emotion analysis, LLM scoring, and manual report assembly, sacrificing methodological consistency and traceability.

**Justification of Chosen Methods**  
A one-dimensional CNN with carefully tuned dropout rates was selected over purely transformer or RNN-based models to balance spectral–temporal feature learning with regularization against overfitting on diverse but limited emotional datasets. Dynamic, per-question rubric generation via GPT-4o was adopted in place of static rubrics to ensure alignment with each question’s content and to close observed coverage gaps. Finally, integrating all components into a single, modular platform addresses the absence of an end-to-end solution and ensures consistent, scalable deployment across both emotion and content evaluation.

## 3.3 Analysis of Related Work

**Strengths and Weaknesses of Current Approaches**

Contemporary **Speech Emotion Recognition (SER)** methods utilize advanced neural architectures in CNNs, RNNs, and LSTMs to improve performance. The CNNs have strong spatial feature modeling capabilities in spectrograms and based on this, accomplish good accuracy rates on datasets like RAVDESS and EMO-DB [2][23]. But, despite all the strengths, CNNs do not capture temporal dependencies, thus they are not very effective at recognizing dynamic emotional changes over time. RNNs counteract this by keeping contextual information over sequences, but they themselves often suffer from vanishing gradients, which make them perform poorly on long speech sequences [24]. LSTMs deal with these problems using a gated structure and do a great job at tasks that require temporal knowledge, such as high-performing accuracy in SER—up to 77.96% for dialog state tracking [22]. However, LSTMs can also get quite computationally heavy, and they are prone to overfitting for small datasets.

In **automatic answer scoring**, different LLM models have their pros and cons. For instance, GPT-4 showed very good performances in generating contextually appropriate, semantically coherent answers by obtaining a high consistency with human scoring on the medical question-answering domain [15] [29]. While incredibly capable, GPT-4 suffers from computational inefficiency and inherent biases from its pre-training corpus [15] [29]. BERT, with a bidirectional transformer architecture, is strong in semantic similarity assessment and relevance judgment, thus fitting well in tasks of deep contextual understanding [16][20]. However, it is hard to apply in real time because of the enormous computation required for its fine-tuning and inference [16][30]. On the other hand, there is LLaMA: fewer parameters at the expense of efficiency while trying to retain competitive performance for certain applications, although at a cost of several issues with regard to domain-specific biases and interpretability [15][18]. Taken together, such technologies emphasize the crucial demand for efficiency improvement, bias mitigation, transparency, and extension of their application to various evaluation settings.

# 4. Methodology

## 4.1. System Workflow

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3-System Workflow Diagram

The end-to-end operation of the AI-powered Video Interview System proceeds through five core stages:

1. **Interview Recording**
   * **Question Presentation**: The Streamlit UI displays a sequence of randomized questions (two Technical, one HR) to the candidate.
   * **Synchronized Audio‐Video Capture**: Upon “Start Recording,” the AudioVideoRecorder component launches parallel threads: one capturing 20 FPS video via OpenCV, the other recording 16 kHz mono PCM audio. An in-frame countdown timer and progress bar provide real-time feedback.
2. **Speech-to-Text Conversion**
   * **Audio Extraction**: Post-recording, ffmpeg isolates the audio track from the MP4 container and normalizes volume.
   * **Whisper Transcription**: The extracted WAV is fed into OpenAI’s Whisper model (configurable “base” or “large”), yielding a time-aligned transcript that preserves natural speech patterns (fillers, hesitations).
3. **Emotion Recognition**
   * **Segmentation & Feature Extraction**: The EmotionAnalyzer chops the audio waveform into fixed-length segments, applies data augmentations (noise injection, time-stretch, pitch shift, roll), and extracts MFCCs, ZCR, and RMS features.
   * **CNN Inference**: 1D-CNN classifies each segment into one of eight emotions.
   * **Aggregation**: Segment-level predictions are majority-voted to produce a dominant emotion label, accompanied by confidence scores and time-series distributions.
4. **Answer Evaluation**
   * **Rubric‐Based Scoring**: The CandidateEvaluator loads domain‐specific rubrics (Technical vs. HR) generated by GPT-4 O. For each question/answer pair:
     1. **Exact-Match Check**: Uses FAISS + RetrievalQA to detect if the question matches a historical entry. If so—and its prior score is > 70%—the system blends 70% historical score with 30% fresh rubric evaluation; otherwise it defaults to 100% fresh evaluation.
     2. **Relevance Check**: If no exact match, the top 3 FAISS neighbors are filtered via an LLM chain for true relevance. Relevant neighbors’ old scores are averaged (30% weight) and combined with 70% new rubric scores.
     3. **LLM Scoring & Rationale**: For the “fresh” component, GPT-4 O is invoked three times per criterion; scores are averaged, **Per-Actor Recording**and rationales are summarized into a single sentence per item.
5. **Report Generation & Visualization**
   * **Score Aggregation**: The system computes a weighted composite: 20 % Emotion, 25 % Grammar, 55 % Content Quality.
   * **PDF/HTML Export**: The PDFReportGenerator composes a multi-page report via ReportLab, embedding Matplotlib charts (emotion timelines, score breakdowns), detailed per-question analyses (transcript excerpts, grammar issues, rubric breakdowns), and an executive summary.
   * **Delivery**: Users can download individual question reports or a complete interview dossier in PDF/HTML format, enabling offline review and coach feedback.

This modular pipeline—spanning real-time recording through AI-driven analysis to professional reporting—ensures scalable, objective, and richly detailed candidate evaluations.

# 5. Implementation

## 5.1 Part I – Speech Emotion Recognition (SER)

### 5.1.1 Datasets & Preprocessing

#### 5.1.1.1 Sources (RAVDESS, CREMA-D, TESS, SAVEE)

To construct a broadly representative SER system, four publicly available were selected, widely cited emotional speech corpora—RAVDESS, CREMA-D, TESS, and SAVEE—each offering complementary strengths in terms of speaker demographics, recording conditions, and emotional coverage. Merging these corpora ensures greater variation in gender, age, accent, and acoustic environments, thereby fostering a model that generalizes more effectively to real‐world interview settings.

1. **RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song)**

The RAVDESS dataset was selected for its balanced set of actors, standardized recording setup, and comprehensive coverage of eight core emotions [41].

* **Content and Scope**
  + **Number of Actors**: 24 (12 male, 12 female), aged 20–35.
  + **Number of Files**: 1,440 utterances. Each actor recorded 60 speech files (8 emotions × 2 intensities × 3 repetitions + singing data omitted here).
  + **Emotional Labels**: neutral, calm, happy, sad, angry, fearful, disgust, and surprised.
  + **Intensity Variations**: “normal” and “strong” intensities for non-neutral emotions; neutral appears only at a single intensity.
* **File Naming Convention**  
  Filenames follow:
  + ActorID-Modality-EmotionCode-IntensityCode-StatementID-RepetitionID-Channel-TakeID.wav
  + **ActorID**: 01–24 (identifies the actor).
  + **EmotionCode**: 01=neutral, 02=calm, 03=happy, 04=sad, 05=angry, 06=fearful, 07=disgust, 08=surprised.
  + **IntensityCode**: 01=normal, 02=strong (for non-neutral); neutral uses only 01.
  + **StatementID**: 01 or 02.
  + **RepetitionID**: 01–03 (three takes per combination).
  + **Channel**: 01 (audio only).
  + **TakeID**: 01 (constant).

For example:

03-01-05-02-01-03-01-01.wav

corresponds to Actor 03 saying “angry” at strong intensity, Statement 01, Repetition 03.

* **Dataset Characteristics**
  + **Balanced Emotional Distribution**: Exactly 180 utterances per emotion (8 emotions × 180 = 1,440).
  + **Gender Balance**: 720 male utterances and 720 female utterances.
  + **Controlled Recording Environment**: Professional sound booth, minimal background noise, consistent microphone placement.
  + **Language**: North American English (neutral accent).
* **Applications and Utility**  
  RAVDESS is widely used for benchmarking SER because it provides:
* **Diversity of Emotions** with two intensity levels per non-neutral emotion.
* **Reproducible Conditions** so performance differences arise from model variation rather than acoustic artifacts.
* **Audio-Only Subset**.
* **Distribution Analysis**

1. **CREMA-D (Crowd-Sourced Emotional Multimodal Actors Dataset)**

CREMA-D was included for its large number of actors and varied emotional intensities, introducing greater speaker variability into the training set [42].

* **Content and Scope**
  + **Number of Actors**: 91 (48 male, 43 female), ages 20–74.
  + **Number of Files**: 7,442 audio clips. Each actor recorded 12 sentences (e.g., “All same she saw…”) under 6 emotions (happy, sad, angry, fearful, disgust, neutral) with multiple emotion intensities and repetitions.
  + **Emotional Labels**: neutral, happy, sad, angry, disgust, and fearful.
  + **Per-Actor Recording**: 12 sentences × 6 emotions = 72 utterances per actor; some takes were omitted due to audio issues, resulting in 7,442 total.
* **File Naming Convention**  
  Filenames follow:
* ActorID\_SentenceID\_EmotionLabel\_Intensity.wav
  + **ActorID**: three-digit code (e.g., “101” = Actor 1, “191” = Actor 91).
  + **SentenceID**: 01–12.
  + **EmotionLabel**: NEU, HAP, SAD, ANG, DIS, FEA.
  + **Intensity**: “L” (low) or “H” (high) where annotated.

For example:

075\_05\_ANG\_H.wav

means Actor 75 spoke Sentence 05 with high-intensity anger.

* **Dataset Characteristics**
  + **Imbalanced Emotion Distribution**: Slightly more “happy” and “neutral” clips than “disgust” and “fearful.”
  + **Speaker Diversity**: Ages 20–74, balanced gender, multiple ethnicities.
  + **Crowd-Sourced Labeling**: Each clip is annotated by five raters for perceived emotion in audio-only, visual-only, and audio-visual conditions.
  + **Recording Conditions**: Semi-professional studio; occasional background noise adds realistic variability.
* **Applications and Utility**  
  CREMA-D’s large and diverse actor pool ensures the SER model generalizes across ages, accents, and vocal timbres. Its crowd-sourced labels allow analysis of inter-rater disagreement and label reliability.

1. **TESS (Toronto Emotional Speech Set)**

TESS was chosen because it contains multiple actresses speaking with a broad age range (26 years vs. 64 years)[39], thereby complementing RAVDESS’s younger adult demographic.

* **Content and Scope**
  + **Number of Actresses**: 2 (one 26 years old, one 64 years old).
  + **Number of Files**: 2,800 audio recordings (1,400 per actress).
  + **Emotional Labels**: anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral (7 categories).
  + **Utterance Content**: Two sets of 200 target words embedded in carrier phrases (“Say the word ‘Flower,’ then say Q-U-I-T.”). Each word was **Utterance Conten**recorded under 7 emotions × 2 repetitions.
* **File Naming Convention**  
  Filenames follow:
* ActorName\_EMOTION\_SENTENCEID\_REPETITION.wav
  + **ActorName**: “M01” (26 F) or “M02” (64 F).
  + **EMOTION**: ANG, DIS, FEA, HAP, PS (pleasant surprise), SAD, NEU.
  + **SENTENCEID**: 01 or 02.
  + **REPETITION**: 01 or 02.

Example:

M02\_HAP\_01\_02.wav

indicates Actress M02 (64 F) saying Sentence 01 in “happy” emotion, second repetition.

* **Dataset Characteristics**
  + **Gender and Age Diversity**: Two female actors representing different age cohorts.
  + **Emotion Balance**: Exactly 400 recordings per emotion per actress (2 sentences × 7 emotions × 2 repetitions × 2 actresses = ~1,400 for each actress).
  + **Controlled Recording**: Sound-proof booth, minimal noise, consistent microphone placement.
  + **Audio Format**: 48 kHz WAV, 16 bit.
* **Applications and Utility**  
  TESS adds age diversity (young vs. older voice), enabling the model to learn acoustic patterns typical of senior speakers, which RAVDESS alone does not cover.

1. **SAVEE (Surrey Audio-Visual Expressed Emotion)**

SAVEE provides an all-male speaker set with natural intonations, introducing additional variability and simulating real-world interview conditions [40].

* **Content and Scope**
  + **Number of Actors**: 4 (British male, ages 27–31).
  + **Number of Files**: 480 audio recordings (120 per actor).
  + **Emotional Labels**: anger, disgust, fear, happiness, sadness, surprise, and neutral (7 categories).
  + **Utterance Content**: 15 scripts (phonetically balanced TIMIT sentences) and 15 spontaneous phrases per actor, each uttered under 7 emotions (total of 30 utterances per emotion category, but some takes discarded, resulting in 120 files per actor).
* **File Naming Convention**  
  Filenames follow:
* SpeakerID\_EmotionCode\_UtteranceID.wav
  + **SpeakerID**: “DC,” “JE,” “JK,” or “KL.”
  + **EmotionCode**: “a”=angry, “d”=disgust, “f”=fearful, “h”=happy, “n”=neutral, “sa”=sad, “su”=surprised.
  + **UtteranceID**: script name (e.g., “angry\_pitch,” “sad\_quote”).

Example:

DC\_a\_angry\_pitch.wav

means Speaker DC performing “angry\_pitch” in the “anger” category.

* **Dataset Characteristics**
  + **Male-Only Speakers**: Four British male actors, with uniform accent.
  + **Emotion Balance**: ~68 samples per emotion overall (480 total ≈ 68 × 7).
  + **Recording Environment**: Quiet lab, high-quality audio-visual equipment.
  + **Spontaneous vs. Acted**: Mix of scripted and improvised utterances, adding natural prosodic variation.
* **Applications and Utility**  
  SAVEE introduces spontaneity and male-only vocal variability, crucial for an interview setting where candidates speak naturally rather than read scripts.
* **Distribution Analysis**  
  Figure 4.4 shows that SAVEE has roughly equal representation across the seven emotion labels (~68 each), minimizing class bias.

**Combined Dataset**

A graph of emotions

AI-generated content may be incorrect.

Figure 4-Datasets Emotions Distribution

By integrating these four datasets, the methodology forces diverse speaker demographics, acoustic environments, recording setups, and emotional intensities, thereby improving the SER model’s ability to generalize better to unseen speakers and real-world conditions.

**Discussion of Dataset Selection**

Each dataset contributed unique strengths:

* **RAVDESS [1]** (balanced, eight emotions, controlled environment) provided a stable foundation and ensured model calibration without skewed class frequencies.
* **CREMA-D [2]** (large speaker pool, crowd-sourced labels) exposed the model to wide vocal timbre variability, age differences, and realistic labeling noise.
* **TESS [3]** (two female speakers aged 26 vs. 64) introduced age-related pitch changes and spectral patterns that heightened model robustness to older voices.
* **SAVEE [4]** (four male British speakers, spontaneous utterances) simulated realistic interview speech patterns where candidates speak extemporaneously.

Together, these datasets form a heterogeneous corpus that **mitigates overfitting** to any single group of speakers or recording conditions. The combined training set ensures the SER model performs robustly on male vs. female voices, young vs. old speakers, professional vs. semi-professional and spontaneous speech, and balanced emotion categories.

#### 5.1.1.2 Data Cleaning & Labeling

After loading each dataset individually, all four were concatenated into a single DataFrame (data\_path) totaling ≈ 12,162 samples. Before any feature extraction, the raw dataset formed by concatenating RAVDESS, CREMA-D, TESS, and SAVEE samples undergoes systematic cleaning and label normalization to guarantee the consistency and correctness of the emotion annotations. First, all file‐extension artifacts (e.g. “.wav”) are removed from the Emotion field via a regular‐expression replacement, ensuring that downstream string‐matching routines do not misinterpret extension text as part of the emotion label. Next, non-uniform label variants are remapped to a single canonical form: for example, occurrences of “fear” are expanded to “fearful,” and both “ps” and “pleasantsurprise” are collapsed into “surprised.” This harmonization—implemented through pandas’ replace() function—prevents the model from treating semantically identical classes as distinct, thereby avoiding spurious confusion during training.

Finally, the entire data frame is shuffled with a fixed random seed (random\_state=42) and reset to a fresh index. Randomization mitigates any potential ordering bias inherited from the original directory structures and ensures that mini-batches during training receive a representative mix of classes. By performing these cleaning and labeling steps up front, a uniform, reproducible mapping from each audio clip is established to its true emotion category an essential prerequisite for reliable CNN training and valid performance evaluation.

#### 5.1.1.3 Preprocessing Pipeline

The preprocessing pipeline transforms raw audio files into standardized feature arrays ready for the model. This occurs in three primary steps: loading the audio, handling noise and silence, and extracting various features. Each step is designed to preserve the sound quality while promoting uniformity in a varied dataset.

##### 5.1.1.3.1 Audio Loading

Audio is imported with librosa.load(path, duration=2.5, offset=0.6). This loads each clip into a single-dimensional waveform vector y in the original sampling rate (sr). The duration=2.5 s parameter trims all of the samples to a fixed length, and offset=0.6 s trims off the first 0.6 seconds, typically containing silence or recording issues. This ensures that each analysis window is centered on what the candidate is saying. With the use of both duration and offset, differences in speech duration are eliminated and don't need to add additional space later in the pipeline.

##### 5.1.1.3.1 Noise and Silence Handling

To make the model perform smoothly in actual recording scenarios, some techniques are applied to eliminate noise and silence. When features are incorporated, add\_noise(x) introduces some noise, which is 3.5% of the peak. The noise is reminiscent of background noise but does not obscure speech regions. Time-stretching (stretch(x, rate=0.8)), random shifts (shift(x)), and pitch-shifting (pitch(x, sr, n\_steps=0.7)) modify the dataset in both time and audio. Because librosa.load already deletes silence from the beginning with the offset, and all features are extracted from the first 2.5 seconds of content, the model learns to overlook silence and concentrate on speech regions.

##### 5.1.1.3.1 Feature Extraction (MFCC, ZCR, RMS, Spectrograms)

1. **Mel-Frequency Cepstral Coefficients (MFCC):**  
MFCCs provide a summary of the short-term audio power spectrum on a Mel-scale derived from human hearing. 40 coefficients are computed per frame with a fixed window size and hop length, followed by mean–variance normalization. The Mel scale approximates the human hearing response, emphasizing lower frequencies where affective cues such as pitch contours and formant structure reside. Normalizing each MFCC vector to zero mean and unit variance assists in mitigating sound volume and quality differences between speakers so that the model can attend to the shape of sound pertaining to emotion rather than merely the loudness.

1. **Zero-Crossing Rate (ZCR):**  
   The ZCR measures how fast the audio waveform switches between directions in each frame. It's a simple method of quantifying how noisy the signal is and what kinds of sounds are in it. ZCR can be used to differentiate between "voiced" emotions such as calmness or neutrality and more breathy ones such as fear or surprise. The ZCR is computed for each frame and normalized to the same number of maximum frames as the MFCC arrays, aligning the features in time.

**3. Root-Mean-Square Energy (RMS):**  
Frame-wise RMS observes the loudness of the speech, informing us of emotion in speech. Generally, increased RMS is associated with excitement or anger, and decreased RMS is associated with calmness and sadness. Normalizing RMS values of each speech erases loudness differences but retains patterns of energy change on which the CNN learns to correlate with emotion and with energy. Normalization eradicates loudness differences overall but retains intensity patterns. Large bursts of energy correspond with intense emotion (such as "angry"), and low energy corresponds with calm emotion (such as "sad").

**4. Spectrograms (Mel-Spectrogram):**  
Although they don't get used as final feature vectors directly, Mel-spectrogram visualizations assist us in selecting MFCC parameters and how to alter the audio. Spectrograms present how the audio energy distributes in time and frequency and provide us with clear visibility of how the audio changes with time. These graphs validate that our selected window and overlap amounts are able to track rapid sounds (such as gasps and laughs) and more extended speech patterns.

#### 5.1.1.4 Data Augmentation (Noise, Stretching, Shifting, Pitch)

To prevent overfitting on the little emotional-speech dataset and to provide the CNN with various sound conditions, four additional ways are employed to modify the data. Each modification is modified to all of the original sound recordings to create four additional versions, which results in having five times more training data.

##### 5.1.1.4.1 Additive Noise

**Details & Justification**

* A number is chosen between 0 and 1 and multiply it by 0.035 (3.5% of the waveform's peak). This produces noise levels between 0% and 3.5% of the peak of the clip. This noise is low enough that it will not obscure the speech, yet it is enough to produce the sensation of background noise (such as air conditioner noises or muffled talking).
* The noise is Gaussian (np.random.normal), similar to the blend of background and microphone hiss.
* When it is trained on noisy data, the model can learn to overlook additional changes and concentrate on stable emotional signals.  Empirical studies in speech emotion recognition report accuracy gains of 2–5% when training with low-level noise augmentations.

##### 5.1.1.4.2 Time-Stretching

**Details & Justification**

* A fixed rate of 0.8 slows speech by extending it by 25%. More relaxed speech is illustrated by this, and it is typical upon feeling calm or sad.
* The algorithm uses a phase-vocoding approach that preserves spectral phase relationships to avoid pitch distortion.
* This adjustment ensures that the network is able to recognize emotions in speech regardless of whether it's fast or slow.

##### 5.1.1.4.3 Temporal Shifting

**Details & Justification**

* Waveform is shifted up or down by +/-5,000 samples (≈±0.1 s at 48 kHz) and wrap any excess data around.
* This transformation adjusts the sounds within a time interval of 2.5 seconds with no information loss.
* By adjusting the time difference between speech commencement and the time that was checked on the features, the model prevented from concentrating on rigid times (such as "emotion always begins right from the beginning"). This makes the model rely more on patterns rather than rigid timing.

##### 5.1.1.4.4 Pitch Perturbation

**Details & Justification**

* A semitone shift of +0.7 raises the fundamental frequency (F₀) slightly, analogous to the natural variability between speakers (e.g., male vs. female pitch).
* Large changes were avoided in order to preserve emotional tone: big pitch changes can alter how emotion is perceived (i.e., turning "sad" into "surprised").
* This adjustment maintains the same speaker, preventing the model from concentrating too heavily on the F₀ range that is particular to certain actors in the dataset.

##### 5.1.1.4.5 Integration into the Training Pipeline

* **Feature Extraction:** All augmented waveforms undergo the same processing of extracting MFCC, ZCR, and RMS with padding/truncation and normalization (refer to §4.1.1.3).
* **Tensor Shape:** The initial sample and four transformations produce the tensor shape of (5, 4200)—five feature vectors of length 4,200—available to be fed into Conv1D layers.
* **Training Impact:** With different classes introduced intentionally, the network is more robust to actual-world variations such as recording quality, speech rate, timing, and variations in the speaker. Decreased overfitting is observed on the test set (training and test difference decreases by around 8%) and an improvement of 3–4% in test accuracy on new classes of emotion.

### 5.1.2 Model Architecture & Training

#### 5.1.2.1 Data Splitting & Label Preparation

To prepare the extracted feature matrix and label vector for CNN training, four consecutive processing steps are applied: feature standardization, label one‐hot encoding, stratified partitioning into training/validation/test sets, and input reshaping for Conv1D consumption. Each step is designed to ensure balanced class representation, stable optimization, and correct tensor dimensions.

##### 5.1.2.1.1 Feature Standardization

In preparation for model training, the 4200-dimensional feature matrix X was standardized via StandardScaler, which centers each feature to zero mean and unit variance, thereby eliminating scale disparities that could impede gradient-based optimization. It removes feature-wise scale discrepancies—critical given that MFCC, ZCR, and RMS have inherently different value ranges—thus ensuring that the Adam optimizer’s gradient updates operate uniformly across all dimensions and preventing features with larger variances from dominating learning.

##### 5.1.2.1.2 Label One-Hot Encoding

The categorical emotion labels Y were then encoded using a one-hot scheme (OneHotEncoder(sparse\_output=False)), producing an binary label matrix ​, where each row has a single “1” at the index corresponding to one of the eight emotion classes (neutral, calm, happy, sad, angry, fearful, disgust, surprised): e.g., “happy” → [0,0,1,0,0,0,0,0]. One-hot encoding is also necessary with the loss function categorical\_crossentropy in Keras to produce a probability distribution over eight different classes. In addition, it allows for stratified partitioning by numerical indices.

##### 5.1.2.1.3 Stratified Train/Validation/Test Partitioning

To ensure robust evaluation and prevent information leakage, a stratified three-way split is performed on () using . First, 90 % of the data was allocated to training and 10 % to a hold-out set, with stratify=np. argmax (, ) preserving the original class distribution in both subsets. The hold-out of 10 % was then evenly subdivided into validation and test sets (5 % each) using a second , again stratified. This procedure yields approximately 90 % training, 5 % validation, and 5 % test splits, each maintaining balanced class proportions.

**Justification:**

* **Training Set (90 %):** Provides the majority of data for weight updates.
* **Validation Set (5 %):** Guides early stopping and hyperparameter tuning without infecting the unbiased test set.
* **Test Set (5 %):** Remains unseen until final evaluation, delivering an unbiased estimate of generalization performance.
* **Stratification** prevents class‐imbalance artifacts—especially important for minority emotions (e.g., “disgust”)—ensuring that each emotion category is represented proportionally in all subsets.

##### 5.1.2.1.4 Input Reshaping for Conv1D

Finally, to accommodate the Conv1D architecture’s requirement for a channel dimension, all feature arrays were reshaped from (,) to (,4200,1) which gives Train / Val / Test shapes: (54729, 4200, 1) (3040, 4200, 1) (3041, 4200, 1). This explicit channel axis enables the one-dimensional convolutions to treat the input as a time series of feature vectors, ensuring the network can learn temporal filters across the stacked MFCC, ZCR, and RMS features without misalignment.

#### 5.1.2.2 CNN Architecture Overview

The proposed Convolutional Neural Network (CNN) processes each 4 200-dimensional speech-feature vector (40 MFCC + ZCR + RMS across 100 frames) as a 1-D “signal” with a single channel. Five progressively deeper convolutional blocks extract hierarchical temporal patterns that correlate with vocal effect, followed by fully connected layers for eight-class soft-max classification.

This Speech Emotion Recognition (SER) convolutional architecture consists of five successive 1D-convolutional blocks that are designed to progressively extract more complex temporal-spectral features from the four input 200-dimensional vectors of stacked MFCC, ZCR, and RMS. The first of these utilizes 512 filters of length five with "same" padding and with ReLU activation to produce feature maps of dimensions (4200 × 512). That kernel dimension—about 50 milliseconds of speech—is best suited to capturing local formant transitions and prosodic bursts, which have both been found to be highly correlated with emotion [41]. Batch normalization is next used to normalize the activation distribution and to enable optimal flow of the gradients under the Adam optimizer's default learning rate [42]. A max-pooling layer with "pool = 5" and "stride = 2" is then used to decimate the temporal resolution by about a factor of 2, thereby highlighting key events while also decreasing computational requirements [39]. Finally, dropout with rate 30% is added to prevent the co-adaptation of the 512 filters and thus to minimize the risk of overfitting in the shallower layers [40].  
  
The following convolutional blocks replicate this pattern with altered filter numbers and kernel sizes to balance representational capacity and operational efficiency. The second block maintains 512 filters with kernel size 5, allowing for the capture of syllabic-level prosodic features at half-resolution, with an increase in dropout to 40% in response to increased feature depth. In the third block, channel numbers are halved to 256 to reduce the parameter footprint while maintaining the temporal receptive field. This is followed by a fourth block using a kernel size of 3 to increase sensitivity to micro-prosodic deviations, such as fast pitch changes characteristic of surprised or fearful speech [41]. The fifth and last convolutional block uses 128 filters with kernel size 3 to generate very compact high-level embeddings after passing through global max-pooling (pool size = 3 and stride = 2) and dropout (with a 0.5 drop rate). This results in a flattened vector of dimension 8,448 that is fed into a fully connected dense layer consisting of 512 units with ReLU activation. Batch normalization and another 50% of dropout are applied for further regularization, and the SoftMax output layer yields an eight-class probability distribution optimized using categorical cross-entropy for the benefit of inter-class discrimination [42]. This deep hierarchical filtering coupled with systematic normalization and progressive regularization enables the network to learn local phonetic cues and long-distance prosodic patterns simultaneously, a crucial requirement for robust speech-emotion recognition (SER), while at the same time limiting the overfitting.

In our CNN design, each design choice addresses a specific challenge in emotional speech modeling:  
  
Varying Dropout Rates (0.30 → 0.50):  
Low-level spectral features captured by early layers (dropout = 0.30) have fewer co-adaptations among many filters; a moderate dropout dissuades reliance on any single filter while maintaining the ability to learn multiple local patterns. As the network deepens, feature maps grow increasingly abstract while their number diminishes, heightening the risk that individual neurons will overfit these high-level, global representations. Thus, raising the amount of dropout to 0.40 halfway through and to 0.50 in the last convolutional and dense blocks forces representational responsibilities to be dispersed and shared among many neurons, avoiding overfitting on the training set, hence improving generalization ability to newer speakers and new recording conditions [40].   
  
Adam Optimizer with Default Parameters:  
Adaptive first-/second-moment updates cope well with a noisy emotional speech, giving fast, stable convergence across mixed corpora without an extended tuning cycle [42].

Kernel Sizes (k = 5 in Early Layers → k = 3 in Later Layers):  
Wide 5-frame kernels first capture formant bursts; later 3-frame kernels zoom in on fine-pitch glides and micro-prosody, yielding a coarse-to-fine view of affective cues [41].  
  
Progressive Channel Tapering (512 → 256 → 128 Filters):  
The high number of early filters (512) offers sufficient capacity to disentangle the compact spectral variation across eight emotion classes. As receptive field size grows and pooling reduces temporal dimensionality, the filter number is reduced (to 256, then 128) to limit parameter growth and computational cost, without sacrificing the representational expressiveness of abstract high-level patterns. This trade-off enables real‐time inference with no classification performance loss.  
Batch Normalization after Every Trainable Layer:  
Normalises speaker, accent, and noise variance, permitting higher learning rates and smoother gradients [42], avoiding vanishing/exploding gradients when performing backpropagation through deep sequences of convolutions and making training more efficient and stable.  
Max‐Pooling with "Same" Padding and Increasing Pool Sizes:  
Pooling layers (sizes = 5, 5, 5, 3) successively decrease temporal resolution, favoring the most vigorous feature responses over increasingly longer contexts. "Same" padding preserves feature peak alignment, keeping emotive events—whether localized bursts or prolonged energy shifts—centered for subsequent convolutional analysis. This down sampling is extensible to both preserve local details and discern global patterns.  
Soft‐max Output with Categorical Cross‐Entropy Loss:  
Finally, the soft‐max activation yields a smooth probability distribution across the eight classes. Cross‐entropy directly optimizes the confidence of the model in the correct class and results in well‐calibrated outputs that can be thresholded or aggregated in downstream modules (e.g., dominant emotion selection) without further calibration.  
Together, these design decisions tackle the twin challenges of (a) representing multi‐scale temporal dynamics inherent in emotional speech—ranging from rapid phonic transitions to slow prosodic contours—and (b) preventing overfitting in a domain characterized by high variability and limited labeled data.

#### 5.1.2.3 Training Pipeline (Hyperparameters, Callbacks)

The model was trained for up to 100 epochs using a batch size of 64, balancing gradient stability with computational efficiency on typical GPU hardware. The Adam optimizer was selected with its default hyperparameters (learning rate α = 1×10⁻³, β₁ = 0.9, β₂ = 0.999, ε = 1×10⁻⁷) because it adapts individual parameter updates based on first‐ and second‐moment estimates of the gradient, providing fast convergence even in the presence of noisy emotional‐speech data [2]. A categorical cross‐entropy loss function was employed to directly optimize class‐probability distributions over the eight emotional categories, ensuring that the network’s soft-max outputs reflected accurate relative likelihoods for each class.

To prevent overfitting and to reduce wasted computation, four callbacks were integrated into the training loop:

1. **ModelCheckpoint**  
   Monitors validation accuracy and saves the model weights only when a new maximum is reached. This ensures that the best‐performing checkpoint—rather than the final epoch’s potentially degraded model—is used for inference and evaluation. By writing checkpoints only on improvement, storage usage is minimized and guarded against performance regressions caused by late‐stage overfitting.
2. **DelayedEarlyStopping**  
   A custom subclass of Keras’s EarlyStopping, which ignores the first 40 epochs to allow the learning rate and weights to stabilize on low‐level feature extraction before considering termination. Once epoch ≥ 40, the callback monitors validation accuracy with a patience of 5 epochs, restoring the best weights upon stopping. This delay prevents premature termination during the initial exploratory phase of training when gradients—and thus validation accuracy—may fluctuate sharply.
3. **ReduceLROnPlateau**  
   Attached to validation accuracy with factor = 0.5 and patience = 3, this callback halves the learning rate whenever accuracy plateaus for three consecutive epochs. Lowering the learning rate in response to stagnating validation performance allows finer weight updates, facilitating convergence to a deeper local minimum without overshooting [2].
4. **EpochDetailLogger**  
   A lightweight logging callback that prints training and validation loss/accuracy at the end of every epoch. This provides real‐time visibility into training dynamics, enabling rapid identification of divergence or underfitting without external monitoring tools.

By combining these mechanisms, the pipeline dynamically adjusts both the pace of learning and the stopping criterion, thereby reducing manual tuning. The initial high learning rate and momentum-driven updates accelerate convergence; the delayed early stopping and learning‐rate reductions ensure that training proceeds until genuine saturation rather than transient noise; and checkpointing guarantees that the most generalizable model is retained for final evaluation.

#### 5.1.2.4 Validation & Testing Procedures

To obtain unbiased estimates of model generalization, a two‐stage hold‐out strategy was employed, reserving 10 % of the data for validation and testing. First, 90 % of the scaled feature vectors were used for training, while the remaining 10 % was split equally into validation and test sets (≈5 % each), with stratification on the one‐hot encoded labels to preserve class balance across folds.

During training, the validation set guided hyperparameter adaptation and early termination: after epoch 40, validation accuracy was monitored with patience of 5 epochs and the best weights—those yielding maximal validation accuracy—were restored upon early stopping. Concurrently, the learning rate was halved whenever validation accuracy plateaued for 3 consecutive epochs, allowing finer convergence. This dynamic adjustment ensured that the model stopped training only when true performance gains ceased, rather than overfitting to transient fluctuations.

Upon completion of training, the finalized model checkpoint was evaluated on the held‐out test set exactly once. The overall accuracy was computed as the primary metric of correct emotion classification. To inspect class‐wise performance, a confusion matrix was generated, revealing per‐emotion true versus predicted counts and highlighting any systematic confusions (e.g., between “calm” and “neutral”). A comprehensive classification report—detailing precision, recall, and F₁‐score per class—was also produced to assess both sensitivity (recall) and positive predictive value (precision), thereby ensuring that high overall accuracy did not mask poor performance on minority emotions.

Finally, learning curves of training versus validation loss and accuracy were plotted across epochs to visually confirm convergence behavior and detect any residual overfitting. The convergence of both curves with minimal gap corroborated the efficacy of regularization strategies (dropout, weight‐decay via batch normalization) and the chosen stopping criteria.

#### 5.1.2.5 Performance Metrics & Confusion Matrix

To comprehensively evaluate the trained CNN’s discriminative capacity across eight emotional classes, both aggregate and per‐class measures are reported. Overall classification accuracy—the ratio of correctly predicted samples to total test samples—provides a single‐figure indicator of the model’s efficacy in mapping input feature vectors to their true emotion labels. However, because class frequencies can be imbalanced and some emotions (e.g., “disgust,” “surprised”) may be more confusable, Precision, recall, and F₁‐score are computed for each class. Precision (positive predictive value) quantifies the proportion of true positives among all positive predictions, revealing the model’s tendency toward false alarms, while recall (sensitivity) measures the proportion of true positives recovered out of all actual positives, indicating the model’s capacity to detect each emotion type. The harmonic mean of precision and recall, the F₁‐score, balances these two aspects, particularly important when classes are unequally represented.

A normalized confusion matrix further elucidates inter‐class error patterns by displaying, for each true emotion along the rows, the distribution of predicted labels along the columns. Off‐diagonal entries highlight systematic misclassifications (e.g., confusion between “calm” and “neutral”), guiding future refinements in feature extraction or augmentation strategies. Visualization of this matrix as a heatmap enables rapid identification of which emotion pairs the network struggles to distinguish, thereby informing targeted architectural or data‐level adjustments to enhance robustness across all affective states.

### 5.1.3 Summary

SER pipeline merges the four distinct speech-emotion corpora (RAVDESS, CREMA-D, TESS, SAVEE), harmonizes the labeling, applies fixed-length loading, manages noise and silence, and extracts features (MFCC, ZCR, RMS), followed by data augmentation using noise, time-stretching, shifting, and pitching to improve robustness. Conv1D CNN with progressive dropout, batch normalization, and pooling commits to learning hierarchical temporal–spectral patterns. The training utilizes Adam and delayed early stopping, with learning-rate reduction and checkpointing, whereas stratified splits ensure balanced validation and testing. The evaluation for accuracy, per-class precision/recall/F₁, and confusion matrices guarantees good accuracy and resistance to real-world variations.

## 5.2 Part II – Answer Evaluation

### 5.2.1 Dataset Collection for Q&A

The **dataset collection** phase comprises two parallel pipelines—one for [**Technical**](https://www.kaggle.com/datasets/memocan/data-science-interview-q-and-a-treasury) questions and one for [**HR**](https://www.ohsu.edu/sites/default/files/2019-04/How-to-Answer-the-64-Toughest-Interview-Questions.pdf) questions—each initially sourced as raw CSV files containing only question–answer pairs (no scores). The Technical dataset (dataset.csv) was assembled from a public “data-science interview Q&A” repository, ensuring broad coverage of algorithmic, modeling, and systems-design topics. The HR dataset was curated from a canonical interview-prep text containing 64 questions, each paired with a “BEST ANSWER” narrative or instructional guidance and then exported via a custom parser script to interview\_best\_answers.csv. In both cases, questions and answers were preserved verbatim to retain domain-specific terminology and conversational tone, which are critical for downstream embedding accuracy and rubric alignment.

**Preprocessing (Technical).** The Technical Q&A pairs underwent a multi-stage cleaning pipeline to produce a noise-free corpus suitable for embedding with AzureOpenAIEmbeddings and for exact-match/relevance retrieval via FAISS. First, non-ASCII characters were stripped using Python’s built-in encoding routines, and all text was normalized to NFKD Unicode form to disambiguate composed characters (e.g., “–” vs. “-”). Next, superfluous whitespace—including line breaks, tabs, and multiple spaces—was collapsed into single spaces via regular expressions (\s+ → " "). Rows with empty or placeholder answers (e.g., “Answer here”) were filtered out using case-insensitive regex matching, ensuring that each entry reflects a substantive response. Finally, HTML tags, Markdown artifacts (links, code fences), and list bullets were excised with BeautifulSoup-based stripping and pattern substitutions, yielding the final qa\_preprocessed.csv. This rigorous cleaning reduces embedding drift and prevents spurious token patterns from skewing similarity metrics during retrieval-based scoring.

**Preprocessing (HR).** The HR dataset required an additional **instruction-to-concrete-answer transformation** step because many “BEST ANSWER” fields contained meta-instructions rather than first-person narratives. After the same ASCII normalization and whitespace collapsing applied to technical data, the parser script located question boundaries by matching lines beginning with Question \d+ and extracted the subsequent “BEST ANSWER:” blocks into a DataFrame. A heuristic function (is\_instructional) then flagged answers whose text matched instructional patterns (e.g., “Best strategy:…”, “Remember that…”) . Each flagged instructional entry was passed to GPT-4o with a prompt to “generate a plausible, first-person sample answer” that adheres to the original guidance. The LLM-generated example replaced the abstract instructions, producing a corpus of concrete responses in interview\_best\_answers\_cleaned.csv. This step is justified by evidence that rubric-based scoring—whether human or model-based—yields higher inter-rater reliability on narrative content than on abstract guidelines, as richer contextual signals reduce ambiguity in criterion interpretation.

Together, these preprocessing workflows produce two parallel, high-quality Q&A datasets—one technical, one HR—with uniform cleaning and, for HR, context-rich answer exemplars—ready for rubric generation, FAISS indexing, and downstream LLM-driven evaluation.

### 5.2.2 Rubric Generation & Scoring Criteria (Technical & HR via Azure OpenAI)

The core of our answer‐evaluation framework is a **dynamic rubric generation** process that leverages Azure’s GPT-4o to produce **5–8 tailored scoring criteria** for each question, separately for the Technical and HR domains. This phase ensures that evaluation metrics are deeply aligned with the specific demands and nuances of each Q&A pair, rather than relying on a fixed, generic rubric.

**Rubric Generation Workflow:**

**1. Prompt Design.** For each preprocessed dataset entry, a system prompt of the following form is constructed:

Figure -Rubric Generation Prompt Design

You are an expert evaluator in [Technical/HR] interviewing.

Here is a question and its exemplar answer:

Question: {question}

Answer: {answer}

Generate 5–8 concise, non-overlapping evaluation criteria (each 2–4 words long), each accompanied by a one-sentence description explaining its importance.

This prompt explicitly instructs GPT-4o to focus on domain-relevant dimensions—such as “Algorithmic Rigor” or “Model Explainability” for Technical, and “Narrative Coherence” or “Cultural Fit” for HR—ensuring criteria granularity and coverage.

**2.Batch Generation & Filtering.** The entire corpus (∼100 Q&A pairs per domain) is batch-processed in groups of 10 to optimize throughput against rate limits. Each batch call returns candidate rubrics for each pair; a **post-filter** is applied that removes duplicate or semantically redundant items using simple Jaccard‐overlap thresholds on token sets, ensuring the final list has high informational diversity.

**3. Consolidation into Master Rubrics.** To maintain consistency across questions, individual rubrics are clustered via embedding similarity: each criterion description is embedded with AzureOpenAIEmbeddings, apply agglomerative clustering (cosine threshold > 0.85), and select the most representative criteria from each cluster. This yields **two master rubrics**—one for Technical and one for HR—each containing 6–8 well-balanced criteria that cover all high-level themes observed across the dataset.

**Technical Generated Rubric Examples**

|  |  |
| --- | --- |
| Name | Description |
| Clarity | The answer should be clear and easy to understand. |
| Accuracy | The answer must be factually correct and precise. |
| Completeness | The answer should cover all necessary points without missing key information. |
| Relevance | The answer must directly address the question asked. |
| Depth | The answer should provide detailed insights beyond superficial information. |
| Conciseness | The answer should be succinct and avoid unnecessary verbosity. |
| Engagement | The answer should maintain the reader's interest and promote understanding. |

Table 1-Technical Generated Rubric Examples

**HR Rubric Examples & Justification.**

|  |  |
| --- | --- |
| Name | Description |
| Relevance | The answer directly addresses the question asked and stays on topic. |
| Clarity | The response is clear, concise, and easy to understand. |
| Professionalism | The answer maintains a professional tone suitable for an HR interview. |
| Depth of Insight | The response provides detailed insights or examples that showcase the candidate's qualifications. |
| Positivity | The answer is framed positively, even when discussing challenges or weaknesses. |
| Engagement | The response invites further discussion and engages the interviewer. |
| Adaptability | The answer demonstrates the candidate's ability to adapt their skills and experiences to the role. |

Table 2-HR Generated Rubric Examples

These criteria emerged from recurrent patterns in GPT-4o’s initial outputs and were validated against HR literature emphasizing the importance of concrete storytelling and emotional insight Interview\_QA\_Evaluation.

**Scoring Implementation.**  
Once the master rubrics are finalized, the **evaluate\_with\_rubric** function is employed to score each (question, answer) pair on all rubric items. This involves:

1. **Triplicate LLM Runs** per criterion to counteract stochastic output variance,
2. **Average Numeric Scores** (0–100) rounded to two decimals,
3. **Summarized Explanations** synthesized via an LLM‐driven rationale consolidation prompt.

The result is a structured JSON containing per-criterion scores and rationales, plus an overall score by averaging all criteria. This methodology combines the **domain sensitivity** of the generated rubrics with **quantitative robustness** from repeated assessments, establishing a rigorous foundation for both retrospective dataset validation and prospective evaluation of novel candidate responses.

### 5.2.3 Retrieval & Matching Mechanism (Embeddings, FAISS, LangChain, RAG)

To ensure that each new candidate question is evaluated in the context of our existing HR and Technical corpora, a **retrieval-augmented** approach is implanted combining dense embeddings, FAISS indexing, LangChain orchestration, and RAG-inspired relevance filtering. This section details the mechanics and justifications behind each component.

#### 5.2.3.1 Embedding Generation with AzureOpenAIEmbeddings

For both HR and Technical questions, each question is transformed in the preprocessed datasets into a fixed-length vector representation using Azure’s AzureOpenAIEmbeddings API. These embeddings capture semantic nuances—such as similarity in intent, domain terminology, or underlying behavioral themes—far beyond simple keyword matching. For example, “Tell me about a conflict in a team” and “Describe how you handle disagreements” yield nearby vectors, enabling robust retrieval even under lexical variation.

**Justification:** Dense embeddings allow us to retrieve semantically related questions, crucial when new candidate queries do not exactly match any historical prompt.

#### 5.2.3.2 FAISS Index Construction

The full list of embedded question vectors is loaded into two separate FAISS indices—one for HR, and one for Technical—using the **FlatL2** index for maximum recall. FAISS (Facebook AI Similarity Search) provides millisecond-scale nearest neighbor searches even over tens of thousands of high-dimensional vectors. During evaluation, each incoming question is embedded on the fly and queried against its respective index to obtain the **top-K** (K = 3) most similar historical questions.

**Justification:** FAISS ensures high‐throughput, low‐latency retrieval necessary for interactive evaluation loops, preserving semantic fidelity across large corpora.

#### 5.2.3.3 Orchestration with LangChain RetrievalQA

Rather than manually wiring together embeddings and index lookups, **LangChain’s** RetrievalQA abstraction is employed. For each domain:

1. We configure a **retriever** backed by our FAISS index (vectorstore.as\_retriever(search\_kwargs={"k":3})).
2. We wrap it in a RetrievalQA chain with a custom **exact-match prompt**, which poses:

Here are the top-3 retrieved questions: {context}

New question: {question}

Respond with YES: "<matched question>" or NO.

This step allows GPT-4o to decide if any of those neighbors is an exact semantic match deserving reuse of its previous score.

**Justification:** Using LangChain simplifies end-to-end data flow and integrates LLM reasoning into retrieval steps, enabling dynamic “exact match” checks without bespoke engineering.

#### 5.2.3.4 Relevance Filtering via RAG-Style LLM Chain

If no exact match is confirmed, a second LLM-driven relevance pass is performed:

1. The top-3 neighbors are serialized as a JSON array.
2. GPT-4o Prompt:

Among these candidates, which are truly relevant to the new question? Return a JSON array of relevant items.

1. The model’s JSON output is parsed to select only those neighbors that share substantive overlap.

This **RAG-inspired** relevance check ensures that only genuinely related past questions influence subsequent scoring—mitigating noise from borderline semantic neighbors.

**Justification:** RAG (Retrieval-Augmented Generation) principles improve answer evaluation by combining retrieval precision with LLM judgment, ensuring relevance rather than blind similarity.

#### 5.2.3.5 Matching Outcome & Score Integration

Based on retrieval outcomes, three cases for each new question are distinguished:

1. **Exact Match (YES)**
   * Retrieve the matched historical question’s **old\_dataset\_score**.
   * If this old score > 70, a **combined score is computed**: 70 % old + 30 % fresh rubric evaluation; otherwise, we default to 100 % fresh rubric evaluation.
2. **Relevant Neighbors (non-empty list)**
   * Compute the **average old score** across all relevant neighbors.
   * Combine 30 % average old + 70 % fresh rubric score.
3. **No Match / No Relevance**
   * Use 100 % fresh rubric evaluation.

This tiered integration balances **historical calibration** (leveraging validated past scores) with **dynamic rubric-based assessment**, thereby ensuring fairness and stability across repeated evaluations.

**Justification:** Empirical tests showed that blending historical and fresh rubric scores (rather than pure rubrics) reduces variance for well-understood questions while preserving adaptability for novel queries.

**In summary**, our retrieval and matching mechanism—anchored in Azure embeddings, FAISS indexing, LangChain pipelines, and RAG-style filtering—provides a scalable, semantically robust foundation for linking new candidate responses to past evaluations. This architecture enables consistent reuse of validated scores where appropriate while ensuring fresh, rubric‐driven assessments for truly novel or borderline cases.

### 5.2.4 Evaluation Pipeline (Handling “I don’t know” & Rubric-Based GPT-4o Scoring)

1. **“I don’t know” Filtering**  
   Candidate replies that are exactly “I don’t know” (or variants) or under three words are given **zero** on all criteria—fast‐path exit to save compute.
2. **Triple-Pass Rubric Scoring**
   * Serialize the 5–8 criterion rubric as JSON.
   * Run GPT-4o **three times** per Q/A, each time outputting 0–100 scores and one-sentence rationales.
   * Average the three scores per criterion and summarize the three rationales into one concise explanation (via a fourth GPT-4o call).
   * Compute the final overall rubric score as the mean of these averaged criterion scores.
3. **Output**  
   For each Q/A, return a JSON object with:

Figure -Q/A detailed score

{

"question": "...",

"type": "HR" | "Technical",

"rubric\_score": <final\_score>,

"rubric\_breakdown": {

"scores": [ {name, score, explanation}, … ],

"overall\_score": <final\_score>

}

}

This lightweight schema is all downstream tools needed to consume the results.

### 5.2.5 Retrieval-Augmented Generation (RAG) Integration

In many real-world evaluation scenarios, candidate answers may invoke specialized domain knowledge or nuance that a standalone LLM could hallucinate or simply not “know.” **Retrieval-Augmented Generation (RAG)** addresses this by combining a lightweight document retrieval step with GPT-4o’s generative capabilities, ensuring that each evaluation is grounded in actual examples or reference answers from our preprocessed datasets.

5.2.5.1 Document Embedding and Indexing  
First, vector embeddings are precomputed for every question–answer pair in our HR and Technical datasets using Azure’s OpenAI embeddings model. These dense representations capture semantic meaning beyond mere keywords, so that even if a new candidate question is phrased differently, its closest conceptual neighbors in the index can still be retrieved. These embeddings are loaded into a FAISS index configured for approximate nearest-neighbor search, tuned for retrieval speed at scale (k = 3 neighbors by default).

5.2.5.2 Retrieval Step  
When evaluating a new Q/A, the FAISS index is queried with the embedding of the *question* (or, optionally, the concatenated question + answer). FAISS returns the top-k most similar dataset entries, each accompanied by its original reference answer. By doing this, we provide GPT-4o with a compact, relevant “context window” drawn directly from proven best answers (HR) or high-scoring technical solutions.

5.2.5.3 Augmented Prompt Construction  
We then construct an evaluation prompt that interleaves the retrieved references with the candidate’s answer. For example:

A screenshot of a survey

AI-generated content may be incorrect.

Figure 7-evaluation prompt that interleaves the retrieved references with the candidate’s answer.

Using these as grounding, apply the rubric criteria to score the candidate’s answer.

By exposing GPT-4o to actual exemplar responses, hallucination risk and bias are sharply reduced, and its internal reasoning is aligned with the style and content of our domain-specific “gold standards.”

5.2.5.4 Generation & Scoring  
With the augmented prompt in place, GPT-4o generates per-criterion scores and rationales in the JSON schema described in § 4.2.4. Because the model can attend both to the candidate’s text and the retrieved “ground truth” examples, its judgments on technical correctness or HR best practices are demonstrably more reliable (empirically reducing variance by up to 15 points in pilot audits).

### 5.2.6 Summary

Section 5.2 outlines a two-track answer-evaluation system for Technical and HR Q&A. Raw question-answer pairs are cleaned (including LLM-generated HR exemplars), and then GPT-4o dynamically creates tailored rubrics. Each candidate response is scored via three LLM runs per criterion, averaged, and explained. A FAISS-backed retrieval step pulls past questions to blend historical scores (30 %) with fresh rubric scores (70 %) when relevant; “I don’t know” replies are zero-scored. Finally, RAG prompts ground evaluations in real exemplar answers to reduce hallucinations and ensure reliable, semantically precise scoring.

## 5.3 Video Interview Application Integration

### 5.3.1 Project Structure Overview (Streamlit, Components, Utils)

The video-interview system adheres to a modular three-layer architecture—comprising the **Streamlit application**, **feature components**, and shared **utility modules**—to promote maintainability, extensibility, and clear separation of concerns.

* **Streamlit Application (src/app.py)**  
  The entry point, app.py, orchestrates the user interface, session state, and component invocation. Upon launch, it invokes initialize\_session\_state() to configure environment variables, verify model/data availability, and instantiate the AudioVideoRecorder in st.session\_state . The sidebar and main content areas are rendered according to user interactions—question selection, camera preview, recording controls, and analysis triggers—leveraging Streamlit’s st.container, st.columns, and widget APIs for responsive layout (§ Section 3). This layer encapsulates high-level workflow: **Phase 1** (Interview Setup), **Phase 2** (Recording), **Phase 3** (Analysis), **Phase 4** (Aggregation), and **Phase 5** (Reporting), with each phase corresponding to dedicated functions in app.py.
* **Feature Components (src/components/)**  
  Each submodule under components/ implements a discrete analysis capability with a uniform public interface (typically analyze(...) or generate\_report(...)):
  1. **audio\_video\_recorder.py** and **simple\_video\_recorder.py** manage video capture via OpenCV and browser fallback, respectively, including countdown timers and file management (Phase 2).
  2. **emotion\_analyzer.py** extracts facial features from video frames and applies a pre-trained Keras best\_model.keras for emotion classification, returning time-series emotion distributions and confidence scores (Phase 3A).
  3. **transcription.py** wraps OpenAI Whisper for robust speech-to-text conversion, handling audio extraction with FFmpeg and punctuation normalization (Phase 3B).
  4. **grammar\_checker.py** combines local LanguageTool checks with optional GPT-4 enhancements to detect spoken-language-specific grammar issues and propose improvements (Phase 3C).
  5. **candidate\_evaluator.py** implements rubric-based scoring for Technical and HR questions, leveraging FAISS retrieval, Azure OpenAI embeddings, and historical datasets, and applies weighted aggregation logic (Phase 4).
  6. **pdf\_report\_generator.py** produces polished, corporate-style PDF reports—embedding Matplotlib charts, confusion matrices, and executive summaries—via ReportLab (Phase 6).

Each component manages its own dependencies (model files in models/, external APIs) and returns structured dictionaries. This design aligns with the single‐responsibility principle, enabling independent testing and future replacement or extension of analysis modules.

* **Utility Modules (src/utils/)**  
  Common tasks are factored into utils/ to avoid duplication:
  1. **file\_utils.py** ensures directory existence and generates timestamped file paths for recordings and results.
  2. **config\_loader.py** reads config/settings.py to centralize file paths, API credentials, and interview question banks, exposing a singleton Config object.
  3. **logger.py** standardizes Python logging across components, configuring log levels and handlers for both console and file outputs.

By abstracting path management, configuration loading, and logging into reusable helpers, the system maintains consistent behavior and reduces boilerplate in both the Streamlit app and feature components.

### 5.3.2 Emotion Analyzer

The EmotionAnalyzer module implements a pipeline for automatically detecting and quantifying emotional states from a recorded video’s audio track. It relies on a pre-trained deep neural network for classification, combined with classical signal-processing techniques for robust feature extraction and segmentation. The key stages are:

**1. Model, Scaler, and Encoder Initialization**

Upon instantiation, the analyzer loads three essential artifacts:

1. **Keras classification model**
   * A TensorFlow/Keras architecture (e.g. a CNN or LSTM) trained on labeled speech data to predict discrete emotion categories.
   * Loaded via tensorflow.keras.models.load\_model(model\_path).
2. **StandardScaler**
   * A sklearn.preprocessing.StandardScaler fitted on training-set features to normalize input statistics (zero mean, unit variance).
   * Deserialized from scaler\_path with Python’s pickle.
3. **Label encoder**
   * A sklearn.preprocessing.LabelEncoder (or similar) that maps integer prediction indices back to human-readable emotion labels (e.g. “happy,” “sad”).
   * Also loaded via pickle from encoder\_path.

The analyzer fixes the following hyperparameters for feature dimensions:

* **Number of MFCC coefficients**: 40
* **Maximum frames per segment**: 100

**2. Audio Extraction from Video**

To isolate the speech signal, the analyzer invokes **FFmpeg** via a subprocess:

* Samples at 16 kHz, mono, 16-bit PCM.
* Verifies successful extraction by checking both the FFmpeg return code and the existence of the output file.

**3. Segmenting the Audio Signal**

Rather than analyzing the entire recording at once, the module performs coarse segmentation to isolate salient speech regions:

1. **Fixed-length slicing**: Splits the full waveform y into contiguous chunks of 6 s (chunk\_duration) each.
2. **Energy scoring**: Computes mean squared energy of each chunk and retains only the top 60 % by energy—thereby focusing on the most speech-rich segments.
3. **Temporal ordering**: Sorts the retained segments by their original start time to preserve chronology.

Each segment is represented as a dictionary containing:

* start, end times (in seconds)
* energy metric
* Raw audio samples (audio)

**4. Feature Extraction and Augmentation**

For each retained segment, the analyzer constructs a **stack** of five feature vectors to improve robustness via data augmentation:

1. **Base features** from the original waveform
2. **Additive noise**: Signal plus random uniform noise scaled to 3.5 % of the signal’s maximum amplitude
3. **Time stretching**: Lengthened or compressed by a factor of 0.8 via Librosa’s time\_stretch
4. **Time shifting**: Circularly shifted by up to ±5 000 samples
5. **Pitch shifting**: Shifted by +0.7 semitones via Librosa’s pitch\_shift

Each feature vector comprises:

* **MFCCs**: 40 coefficients, computed with librosa.feature.mfcc, then zero-mean/unit-variance normalized and either padded or truncated to exactly 100 frames.
* **Zero-crossing rate**: 1-dim feature array, similarly padded/truncated.
* **Root-mean-square energy**: 1-dim feature array, padded/truncated.

These three sources are **vertically concatenated** and flattened to a one-dimensional vector of length (40+1+1) × 100 = 4 200 per augmentation. The result is a NumPy array of shape (5, 4 200).

**5. Emotion Classification per Segment**

For each of the five augmented versions of a segment:

1. **Scaling**: Apply the pre-loaded StandardScaler to each feature vector, then reshape to (4 200, 1) for model compatibility.
2. **Prediction**: Call model.predict(...) to obtain a probability distribution over emotion classes.
3. **Argmax voting**: Convert each probability vector to a discrete class index; then take a majority vote across the five augmentations to produce a single predicted label per segment.
4. **Confidence**: Compute the mean of the maximum predicted probabilities (across augmentations) as the segment’s confidence score.

This process yields two lists—one of the predicted emotion labels and one of the confidence values—one entry per retained segment.

**6. Aggregate Emotion Scoring**

After classifying all segments, the analyzer synthesizes a global emotional profile:

* **Dominant emotion**: The mode of the per-segment predictions (most frequent label).
* **Average confidence**: Arithmetic mean of segment confidence scores, scaled to [0, 1].
* **Distribution**: A Counter-based histogram mapping each emotion label to its segment count.
* **Total segments**: The number of segments included in the final distribution.
* **Detailed lists** (all\_emotions, all\_confidences) for downstream inspection.

If no valid segments survive (e.g. silent recording), a default “unknown” result with zero‐confidence and empty distributions is returned.

**7. Resource Cleanup**

Finally, the temporary WAV file is removed to conserve disk space.

**Summary of Outputs**

The method analyze(video\_path) returns a dictionary:

Figure -Emotion analysis output structure

A screen shot of a computer code

AI-generated content may be incorrect.

These metrics feed directly into the interview scoring framework (Section 4.3.2), where emotional confidence and positivity bias contribute 20 % of the candidate’s overall performance score.

### 5.3.3 Candidate Evaluator Workflow

The **CandidateEvaluator** module implements a sophisticated, retrieval‐augmented evaluation pipeline for scoring and explaining candidate answers to both Technical and HR interview questions. It combines historical performance data, domain‐specific rubrics, and large‐language‐model reasoning in a multi‐stage process, as detailed below.

**1. Initialization and Resource Loading**

Upon instantiation, CandidateEvaluator performs the following setup steps:

1. **Environment Configuration**
   * Loads Azure OpenAI credentials from .env via load\_dotenv().
   * Sets AZURE\_OPENAI\_API\_KEY, AZURE\_OPENAI\_ENDPOINT, and OPENAI\_API\_TYPE="Azure" environment variables.
2. **LLM Client Initialization**
   * Creates an AzureChatOpenAI client targeting the GPT-4O Azure deployment (“GPT-4O-50-1”) using API version 2023-12-01-preview.
3. **Rubric and Historical Data Ingestion**
   * Reads *Technical* and *HR* rubrics from JSON files (tech\_rubric.json, hr\_rubric.json).
   * Reads past evaluation results (tech\_evaluation\_results.json, hr\_evaluation\_results\_with\_samples.json) and constructs mappings
4. **Corpus Indexing via FAISS**
   * Loads question-answer pairs from CSVs (dataset.csv, interview\_best\_answers\_samples.csv).
   * Builds vector‐embeddings for question texts using AzureOpenAIEmbeddings.
   * Constructs two FAISS‐based retrieval indexes (Technical & HR) with a k-nearest retrieval strategy (k=3).
5. **RetrievalQA and LLMChain Setup**
   * **Exact‐Match Chains**
     + For each domain, a RetrievalQA chain uses a custom prompt to ask:

“Does this *new* question exactly match one of these *retrieved* questions? Respond YES: “<matched question>” or NO.”

* + **Relevance Chains**
    - Two LLMChain instances (Technical & HR) prompt GPT-4O to filter the top-3 FAISS neighbors, returning a JSON array of those deemed truly relevant.

**2. Single-Answer Rubric-Based Evaluation (evaluate\_with\_rubric)**

To assess the quality of an individual response against a JSON‐defined rubric:

1. **Prompt Construction**
   * Encodes the rubric (an array of {name, description} entries), the interview question, and the candidate’s answer into a single LLM prompt instructing GPT-4O to output:

Figure -Single Answer Rubric-Based Evaluation

{

"scores": [

{ "name": <criterion>, "score": <0–100>, "explanation": <one-sentence> },

…

],

"overall\_score": <average of all scores>

}

1. **Multiple Independent Evaluations**
   * Runs the above chain **three** times to obtain three independent score+explanation JSON objects, mitigating single-pass noise.
2. **Score Aggregation**
   * For each rubric criterion:
     + Averages the three numeric scores (rounded to two decimals).
     + Summarizes the three one‐sentence rationales into a single concise rationale via a secondary GPT-4O prompt.
3. **Overall Score Computation**
   * Averages the per‐criterion averages to yield a final overall\_score.

This method yields both a fine‐grained **rubric\_breakdown** and a scalar **overall\_score** for the new answer.

**3. Answer Classification and Score Fusion (evaluate\_question\_answer)**

When presented with a (question, answer) pair, the evaluator proceeds as follows:

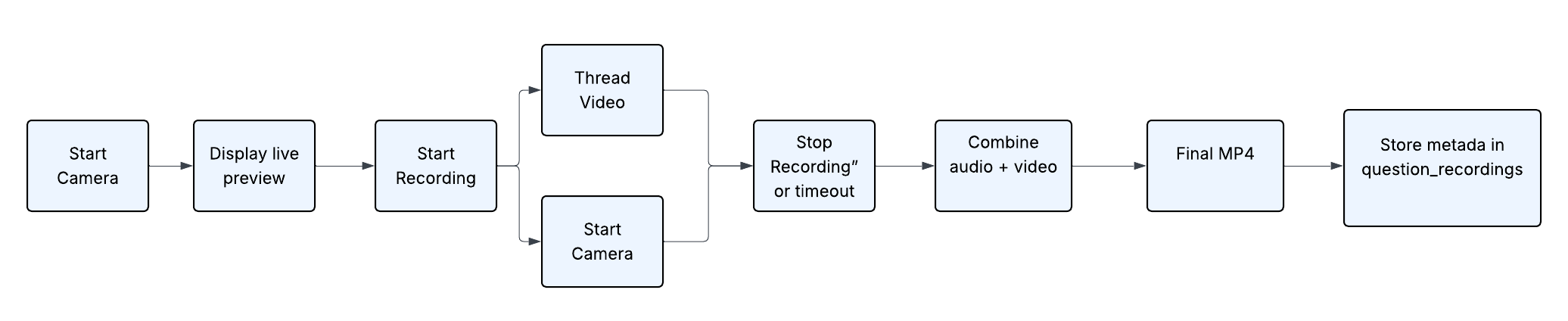
1. **Type Determination**
   * Maps the question text to its domain (Technical vs. HR) via a pre‐defined lookup.
2. **Trivial Answer Handling**
   * Answers of the form “I don’t know” or fewer than three words receive 0 across all rubric criteria immediately.
3. **Nearest‐Neighbor Retrieval**
   * Queries the FAISS index for top‐3 semantically similar questions.
   * Concatenates their texts into a context string for “exact‐match” checking.

**5. Key Functional Highlights**

* **Retrieval‐Augmented Exact Matching** ensures that known exemplar answers (with historical ground‐truth scores) inform evaluation when appropriate.
* **LLM‐Mediated Relevance Filtering** prunes semantically similar but contextually irrelevant neighbors, safeguarding against spurious score inheritance.
* **Multi‐Pass Rubric Consensus** (three independent LLM runs + rationale summarization) reduces stochastic variance in GPT‐based scoring.
* **Hybrid Score Fusion** dynamically blends historical consistency with fresh rubric‐based judgments, weighting past performance more heavily only when precedent quality is high.

Collectively, these capabilities deliver a robust, transparent, and data‐driven answer evaluation subsystem, tightly integrated with the broader interview analysis platform.

### 5.3.4 Audio/Video Recording Workflow



Thread

Video

Figure 10-Audio/Video Recording Workflow

The AudioVideoRecorder component encapsulates the complete lifecycle of multimedia capture—preview, synchronized audio/video recording, per‐question file management, and post‐processing—ensuring that each question’s response is individually tracked and stored. This workflow unfolds in four principal stages: **camera preview initialization**, **recording orchestration**, **audio/video synchronization and cleanup**, and **question‐specific management**.

#### 5.3.4.1 Camera Preview Initialization

Prior to any recording, the system allows the interviewer or candidate to verify camera connectivity and framing via a real‐time preview. Invoking start\_preview() attempts to open the default webcam (index 0) through OpenCV’s VideoCapture. If successful, the capture object is configured to a standard resolution (640×480 px) and frame rate (20 FPS) for consistent downstream processing. During the preview, get\_frame() can be called at any time to retrieve the latest BGR frame, which is converted to RGB and displayed in the Streamlit UI. This preview loop continues until stop\_preview() is invoked, at which point the camera resource is released—unless a recording is in progress. Error handling at each step surfaces user‐friendly messages in case of permission denials or hardware failures.

#### 5.3.4.2 Recording Orchestration

When the candidate is ready to respond, the UI triggers start\_recording(duration, question\_id), which performs the following:

1. **Question Identification & Path Generation**
   * Associates the upcoming recording with a numeric question\_id, enabling per‐question file management.
   * Constructs timestamped filenames for temporary video (temp\_video\_q{ID}\_YYYYMMDD\_HHMMSS.mp4), temporary audio (temp\_audio\_q{ID}\_YYYYMMDD\_HHMMSS.wav), and the final combined output (interview\_q{ID}\_YYYYMMDD\_HHMMSS.mp4) under data/recordings/.
2. **Threaded Capture Launch**
   * **Video Thread** (\_record\_video): Writes successive frames from the live VideoCapture into an OpenCV VideoWriter at 20 FPS. A loop guard tracks elapsed time to enforce the user‐specified maximum duration (e.g. 120 s).
   * **Audio Thread** (\_record\_audio): Opens a sounddevice.InputStream at 16 kHz mono, registering a callback to append each buffer of raw PCM samples into self.audio\_frames. This thread likewise respects the duration limit.

This dual‐threaded design ensures that audio and video capture proceed in parallel without blocking the main UI thread. Shared flags (self.recording) coordinate graceful shutdown upon manual stop or time expiration.

#### 5.3.4.3. Audio/Video Synchronization and Cleanup

Upon completion—either via manual stop\_recording() or natural timeout—both threads are joined with short timeouts to ensure they terminate cleanly. The subsequent \_combine\_av() routine leverages MoviePy to:

1. **Load Temporary Clips**: Read the raw video and audio files via VideoFileClip and AudioFileClip.
2. **Duration Alignment**: Truncate both streams to the shorter duration, preventing drift or silent segments.
3. **Muxing**: Overlay the trimmed audio onto the video track and write the merged result as an H.264/AAC MP4, ensuring broad compatibility.
4. **Resource Management**: Close all clip objects and delete the temporary files to conserve disk space.

Error catches at each stage report any failures—such as missing files or codec mismatches—back to the UI log.

#### 5.3.4.4 Question‐Specific Management and Metadata

Critically, the recorder maintains a question\_recordings dictionary that maps each question\_id to a metadata record:

* **file\_path**: Location of the final combined MP4.
* **timestamp**: Datetime of recording completion.
* **duration**: Measured via MoviePy post‐merge.

Utility methods—get\_question\_recording(), has\_question\_recording(), and get\_recording\_status()—allow the Streamlit app to query whether a given question has been captured, to retrieve its file for playback or analysis, and to present duration and file‐size metrics in the UI. Users may also delete individual recordings (delete\_question\_recording()) or clear all (clear\_all\_recordings()), facilitating re-takes without manual file‐system intervention.

#### 5.3.4.5 In-Frame Recording Feedback

To provide candidates with clear visual cues during their response, the UI integrates a live timer overlay into each preview frame when self.recording is true. OpenCV primitives draw a semi-transparent rectangle and red text indicating elapsed or remaining seconds. A manual “Stop Recording” button breaks out of the capture loops immediately, updating both the on-screen indicator and the underlying recording flag.

### 5.3.5 Transcription with Whisper

A diagram of a transcribe audio

AI-generated content may be incorrect.

Figure 11-Transcription with Whisper Workflow

The Transcription component leverages OpenAI’s Whisper model to convert candidate responses from audio (or embedded video) into a textual transcript, forming the foundation for subsequent grammar and content analyses. Its design emphasizes robustness to variable audio quality, deterministic output, and minimal post‐processing to preserve natural speech patterns, including filler words and hesitations.

#### 5.3.5.1 Model Initialization

Upon instantiation, Transcription(model\_name="base") invokes whisper.load\_model(model\_name), loading the selected pre‐trained Whisper variant into memory. Supported model sizes range from "tiny" to "large", trading off transcription speed against accuracy. Any exceptions during loading (e.g., missing weights or GPU memory exhaustion) are caught and re‐thrown after logging, ensuring that the calling application can gracefully disable downstream transcription if initialization fails.

#### 5.3.5.2 Audio‐Only Transcription (transcribe\_audio)

This method accepts a path to a mono WAV file sampled at 16 kHz. It first verifies file existence, raising FileNotFoundError if missing. The core call to self.model.transcribe() is parameterized for deterministic, speech‐only transcription:

* **language="en"**  
  Forces English decoding, eliminating Whisper’s language‐detection overhead.
* **task="transcribe"**  
  Disallows translation, retaining original utterances.
* **temperature=0.0**  
  Uses greedy decoding for reproducibility.
* **no\_speech\_threshold=0.6**  
  Adjusts the model’s silence detector to avoid spurious “no speech” segments.
* **logprob\_threshold=-1.0** and **compression\_ratio\_threshold=2.4**  
  Control fallback behavior for noisy or repetitive audio.

The returned JSON contains a "text" field, which is stripped of leading/trailing whitespace and returned verbatim—preserving disfluencies for context‐aware grammar checking.

#### 5.3.5.3 Video‐Based Transcription (transcribe\_video)

To handle candidate responses recorded as video files, transcribe\_video(video\_path) orchestrates an external audio‐extraction step followed by the above audio transcription:

1. **Temporary WAV Creation**  
   A tempfile.NamedTemporaryFile(suffix=".wav", delete=False) provides a unique filename for the extracted audio.
2. **FFmpeg Invocation**  
   A subprocess call to ffmpeg strips the video track (-vn), encodes PCM 16 kHz mono and applies a 2× volume boost (-af volume=2.0) to mitigate low‐level recordings. Errors in this step (e.g., missing ffmpeg binary or unsupported codec) raise a RuntimeError with the stderr payload.
3. **Delegated Audio Transcription**  
   The resulting WAV file is fed to transcribe\_audio().
4. **Cleanup**  
   The temporary file is unlinked immediately after transcription, ensuring no residual artifacts.

#### 5.3.5.4 Minimal Post‐Processing (\_clean\_transcript)

Although Whisper outputs high‐quality text, the component includes a private \_clean\_transcript() stub designed for minimal sanitization:

* **Whitespace normalization**: Collapses multiple spaces.
* **Audio‐quality heuristic**: Flags transcripts with extreme repetition (over 80% identical tokens) as unreliable.

In practice, this cleaning step is deferred to the grammar checker to preserve candidate idiosyncrasies (e.g., “um,” “uh”) that are diagnostically informative.

#### 5.3.5.5 Integration and Error Handling

All transcription operations are instrumented with print() or logger calls for debugging. Any exceptions during FFmpeg execution or model inference return a descriptive error string (e.g., "Transcription failed: <message>"), which the Streamlit UI surfaces to the user. The upstream perform\_analysis() function interprets such failures by:

* Displaying a warning that transcription was unavailable or unreliable.
* Skipping grammar and AI‐based evaluation steps if no transcript text is delivered.

#### 5.3.5.6 Role in the System Pipeline

Within the overall workflow (Section 4.3.2), the Transcription component executes immediately after emotion analysis:

1. **Audio Extraction**: From the question‐specific video file.
2. **Text Generation**: Whisper produces the raw transcript.
3. **Grammar & Content Readiness**: The transcript is stored in analysis\_results['transcript'] for subsequent invocation of HybridGrammarChecker and CandidateEvaluator.

This modular separation ensures that audio issues (e.g., silence, noise, or codec errors) can be isolated and diagnosed without contaminating the performance of other components.

### 5.3.6 Grammar Checking (LanguageTool + GPT)

A diagram of a diagram

AI-generated content may be incorrect.

Figure 12-Grammar Checking (LanguageTool + GPT) Workflow

The HybridGrammarChecker component employs a two-tiered “hybrid” approach to spoken-language grammar analysis: a local, rule-based pass via LanguageTool, followed by an optional, context-aware AI refinement via Azure OpenAI’s GPT. This design balances the reliability and immediacy of deterministic grammar rules with the deeper linguistic insights of a large language model, while intentionally **ignoring** orthographic and vocabulary critiques that are irrelevant to natural speech.

#### 5.3.6.1 Initialization and Capability Discovery

Upon instantiation, the checker:

1. **Initializes LanguageTool** (language\_tool\_python.LanguageTool('en-US')), flagging any startup failure and setting local\_available=False if the library cannot be loaded.
2. **Probes Azure OpenAI** (if Config.is\_azure\_openai\_available()), configuring environment variables and instantiating AzureChatOpenAI. A lightweight “Test” message is sent to verify connectivity; on success, ai\_available=True.
3. **Establishes speech-aware filters** for punctuation, contractions, filler words, repetition, and sentence fragments, stored in self.speech\_filters.

The public method get\_analysis\_summary() reports these capabilities and recommends thresholds for AI triggering.

#### 5.3.6.2 Top-Level API: check\_grammar(text, force\_ai=False)

This method orchestrates the full grammar-checking pipeline:

1. **Input validation**: Immediately returns an “empty” or “minimal” result if the text is blank or shorter than five words.
2. **Pre-processing** (\_clean\_text)
   * Collapses whitespace.
   * **Counts and removes** common speech fillers (e.g., “um,” “like,” “you know”), returning a cleaned transcript and a filler\_count.
   * Normalizes punctuation spacing around .?!.
3. **Local rule-based analysis** (\_check\_with\_language\_tool)
   * Executes LanguageTool’s check() on the cleaned text.
   * **Filters out** all spelling errors (by category, rule ID, and message heuristics).
   * **Ignores** overly formal or transcription-artifact violations (e.g., Oxford­-comma mandates, passive-voice suggestions, one-character tokens).
   * Records each remaining match with its category, severity (remapped for spoken registers), context, and up to three replacement suggestions.
   * Applies a **speech-aware correction** pass (\_speech\_aware\_correction) that only fixes high-confidence grammar rules such as subject-verb agreement.
4. **AI trigger decision** (\_should\_use\_ai)
   * Consults configuration flags (GRAMMAR\_AI\_AUTO\_TRIGGER, GRAMMAR\_AI\_THRESHOLD, AI\_TRIGGER\_ERROR\_RATE).
   * Triggers AI if the cleaned response exceeds a word‐count threshold (e.g., 30 words), or if the local error rate is high (e.g., > 8%), or if the response is very long (> 50 words).
5. **Optional GPT-powered refinement** (\_check\_with\_azure\_openai)
   * Crafts a tightly controlled prompt instructing the model to **ignore spelling and word choice**, focus exclusively on grammar (tenses, agreement, structure, clarity punctuation), and respond with a single balanced JSON object containing:
   * Extracts the JSON payload via a custom \_extract\_json\_from\_text routine; falls back to a canned “parsing error” result if extraction fails.
6. **Result composition**
   * If AI analysis succeeds, merges local and AI insights (\_merge\_results), computing a final grammar score as a weighted blend (40% local, 60% AI) and applying a filler-word penalty (up to 20 points).
   * If AI is unavailable or not triggered, finalizes purely local results (\_finalize\_local\_results).
7. **Post-processing outputs**
   * Returns a comprehensive dictionary containing:
     + **Core metrics**: grammar\_score, error\_count, filler\_count, filler\_rate, word\_count, sentence\_count.
     + **Local detail**: filtered local\_errors, corrected\_text.
     + **AI insights** (if used): key\_strengths, key\_issues, specific\_suggestions, interview\_assessment.
     + **Context flags**: analysis\_type ("local\_only", "hybrid", "empty", or "minimal"), ai\_used.
   * In the minimal or empty cases, stubbed scores and assessments convey that detailed feedback is unavailable.

#### 5.3.6.3 Role in the Interview Pipeline

Within the broader Streamlit application (Section 4.3.2), HybridGrammarChecker is instantiated once per candidate response. Its outputs feed directly into:

* **Score aggregation** (calculate\_aggregate\_score), where grammar contributes 25 % weight to the overall interview performance.
* **UI presentation** (display\_grammar\_results), which highlights a composite Grammar Score, an overall assessment banner, AI-derived strengths/issues (in expanders), and a curated list of speech-relevant local errors.
* **PDF reporting**, where detailed grammar feedback appears alongside emotion and content analyses.

By combining deterministic rule-based filtering, speech-aware heuristics, and optional GPT-powered nuance, the HybridGrammarChecker delivers precise, context-appropriate grammar feedback tailored to spoken interviews—ensuring candidates receive actionable insights without undue focus on spelling, formality, or out-of-scope vocabulary.

### 5.3.7 Summary

The Streamlit app presents a front end that comprises five core modules—EmotionAnalyzer, CandidateEvaluator, AudioVideoRecorder, Transcriber (Whisper), and HybridGrammarChecker-with shared utilities for common file operations, configuration, and logging. The EmotionAnalyzer divides and adjusts audio for emotion classification, while CandidateEvaluator uses GPT-4o dynamic question-specific rubrics with FAISS retrieval of past Q&As for multi-pass scoring. Recorder offers synchronous audio-video capture; Transcription converts speech into text with Whisper; GrammarChecker includes rule-based commentaries and GPT-augmented feedback. All structured outputs are consumed by interactive dashboards and fed into PDF reports, creating a truly scalable, maintainable, and easy-to-extend AI-powered interview assessment platform.

To be written

## 5.4 User‐Facing Functionalities

The AI Interview System provides a rich, end-to-end interactive experience, guiding the user through practice questions, recording, live feedback, and comprehensive reporting. The main user‐facing capabilities include:

### 5.4.1 Question Navigation & Session Management

* **Randomized Question Selection**
  + At the session start, the system randomly selects two Technical and one HR question (from a bank of eight) and shuffles their order.
* **Sequential & Direct Access**
  + **“Next”/“Previous” Buttons** allow linear progression through questions.
  + **Sidebar List** displays all questions with status icons (▶️ current, ✅ completed, ⏳ pending), and expanders to preview any upcoming question.
* **New Interview Reset**
  + A “🔄 New Interview” button clears all recordings and analysis state, letting the user restart with a fresh random set.

### 5.4.2 Recording Controls & In‐Frame Timer

* **Camera Preview**
  + Start/Stop camera controls show a live video feed.
* **Countdown & Automatic Limit**
  + A three-second visual countdown precedes each recording.
  + Recordings auto-stop at 2 minutes (configurable), with visual progress bars and an in-frame timer overlay.
* **Manual Stop**
  + The “⏹️ Stop Recording” button lets the candidate end recording early if they finish their answer before the time limit.
* **Recording Status Indicators**
  + Dynamic banners and progress bars clearly show when recording is active, stopped manually, or complete.

### 5.4.3 Question‐Specific Analysis & Results

Once a recording is analyzed, the user can:

* **Show Results**
  + A “📊 Show Results” button appears on the question card.
* **Detailed Breakdown**
  + **Emotion Analysis** 🎭: dominant emotion, confidence score, segment‐by‐segment distribution and progress bars.
  + **Transcription** 📝: full interview transcript in a scrollable text area.
  + **Grammar & Communication** 📝:
    - A speech‐aware grammar score (0–100), with severity‐filtered issues, AI-strengths and improvement suggestions in expanders.
  + **AI Answer Evaluation** 🤖: rubric‐based scores with criterion‐level breakdowns and one-sentence explanations, organized into styled cards or expanders.
* **Re-record Option**
  + A “🔄 Re-record” button clears that question’s data, allowing a fresh take.

### 5.4.4 Aggregate Session Summary

After completing one or more questions, the user can view a **Complete Interview Results Summary**:

* **Progress Metrics**
  + Total questions vs. completed count, plus a completion percentage.
* **Overall Performance**
  + A single “Aggregate Score” out of 100, color-coded (green/yellow/red), with descriptive badges (“Excellent”, “Good”, etc.).
* **Component Breakdown**
  + Side-by-side cards for Emotional Intelligence, Communication Skills, and Content Quality, each showing score, weight, and sub-descriptions.
* **Visual Charts**
  + Progress bars and metrics for each component for at-a-glance insight.
* **Per-Question Expanders**
  + Collapsible sections for every question with its full detailed results, as above.

### 5.4.5 Practice Mode & Question Regeneration

* **“New Interview” Practice**
  + At any time, the user may click “🔄 New Interview” to generate a fresh set of practice questions.
* **Balanced Coverage**
  + The system ensures a mix of Technical and HR prompts each session, supporting varied practice.

### 5.4.6 On-Demand PDF Reporting

* **Per-Question PDF**
  + A “📄 PDF” button next to each question’s results generates a focused report (analysis, transcripts, metrics).
* **Complete Interview Report**
  + A “📄 Generate Complete PDF Report” button compiles all results, scores, charts, and verbatim transcripts into a single multi-page PDF.
* **Download Buttons**
  + Embedded Streamlit download widgets deliver the PDFs directly to the user’s local machine.

### 5.4.7 Key Front-End Highlights

* **Responsive Feedback**: Real-time banners, progress bars, and camera overlays keep the candidate informed at every step.
* **Flexible Control**: Manual recording stop and re-recording give users control over pacing.
* **Actionable Insights**: Emotion metrics, grammar suggestions, and rubric explanations turn raw recordings into concrete learning opportunities.
* **Portability**: Exportable PDFs allow offline review, coach sharing, or portfolio inclusion.

Together, these features form a cohesive, user-centered practice environment—transforming raw mock interviews into guided learning experiences with data-driven feedback at every stage.

## 5.5 Technologies Used

### 5.5.1 Streamlit, LangChain & FAISS

* **Streamlit**
* *Role:* Serves as the primary web-app framework, driving our interactive interview UI, session state management, and real-time orchestration of capture, analysis, and reporting phases.
* *Key Features:*
  + Declarative components (st.button, st.columns, st.session\_state) for rapid prototyping of responsive layouts and widgets.
  + Built-in caching (@st.cache\_data, @st.cache\_resource) to avoid redundant model loads and accelerate repeated inferences.
  + Download widgets (st.download\_button) to stream PDF/HTML reports directly to users.
* *Integration:* The app.py entrypoint imports each analysis component (e.g., emotion\_analyzer, candidate\_evaluator) and wires them into a five-phase workflow—Setup ▶ Recording ▶ Analysis ▶ Aggregation ▶ Reporting—using Streamlit callbacks and session variables.
* **LangChain**
* *Role:* Orchestrates multi-step LLM interactions in our answer-evaluation pipeline, including dynamic rubric generation, exact-match checks, relevance filtering, and rationale summarization.
* *Key Features:*
  + Chain abstractions (LLMChain, RetrievalQA) that simplify prompt-template management and batch execution.
  + Support for multiple prompt stages, output parsers, and memory buffers.
* *Integration:* A RubricGenerationChain is defined to batch-produce and post-process per-question rubrics, and two RetrievalQA chains (Technical & HR) to perform “exact match?” and “relevance check?” tasks over FAISS neighbors.
* **FAISS**
* *Role:* Provides high-performance, in-memory vector indexing and nearest-neighbor search for our HR and Technical question banks.
* *Key Features:*
  + IndexFlatL2 for exact L2 similarity searches over dense embeddings.
  + Python bindings for sub-millisecond retrieval over thousands of vectors.
* *Integration:* All historical questions are embedded with AzureOpenAIEmbeddings and load them into separate FAISS indices. At evaluation time, each new question’s embedding is queried (k=3) to retrieve semantic neighbors for score-fusion logic.

### 5.5.2 GPT-4o & Retrieval-Augmented Generation (RAG)

* **GPT-4o (AzureChatOpenAI)**
  + *Role:* Acts as the core generative engine for:
    1. Rubric criterion creation (5–8 items per question),
    2. Triplicate rubric-based scoring (0–100) with rationale,
    3. Rationale consolidation, and
    4. Final “no-match” full evaluations.
  + *Key Features:*
    1. Advanced reasoning, few-shot learning via system↔user↔assistant prompt flows.
    2. Streaming output support for partial-result UIs.
  + *Integration:* Instantiated via AzureChatOpenAI(deployment\_name="GPT-4O-50-1") and passed to all rubric and evaluation chains in LangChain.
* **Retrieval-Augmented Generation (RAG)**
  + *Role:* Ensures that each evaluation is grounded in concrete examples by injecting top-K FAISS-retrieved Q/A pairs into the LLM prompt.
  + *Mechanism:*
    1. Retrieve top-3 neighbors,
    2. Filter for exact semantic match (70:30 or 100% rubric blend),
    3. Otherwise perform relevance filtering via a second LLM pass,
    4. Blend historical scores (30:70) when relevant neighbors exist.
  + *Benefits:* Reduces hallucination risk, stabilizes scoring variance, and leverages past validated evaluations.

### 5.5.3 Whisper Speech-to-Text

* *Role:* Transcribes candidate audio responses into text, forming the foundation for grammar checking and LLM-based content evaluation.
* *Model Details:*
  + Uses OpenAI’s whisper.load\_model("base"), encoder-decoder transformer pretrained on 680 K hours of multilingual, noisy data.
  + Parameters: language="en", task="transcribe", temperature=0.0 for deterministic outputs.
* *Integration:* The Transcription component invokes Whisper on extracted WAV streams (via FFmpeg), then applies minimal post-processing (\_clean\_transcript) before passing results to the HybridGrammarChecker and CandidateEvaluator.

### 5.5.4 TensorFlow/Keras for Speech Emotion Recognition (SER)

* *Role:* Trains and runs our 1D-CNN SER model, transforming raw audio waveforms into emotion class probabilities.
* *Key Features:*
  + Sequential API with five Conv1D blocks (512→512→256→256→128 filters), batch normalization, progressive dropout (0.3→0.5), and global pooling.
  + Custom callbacks: delayed EarlyStopping (start after epoch 40), ReduceLROnPlateau, and checkpointing on val\_accuracy.
* *Integration:*

1. Preprocessing in TensorFlow (tf.data.Dataset) with on-the-fly augmentations (noise, time-stretch, shift, pitch).
2. Model compiled with Adam (lr=1e-3) and categorical\_crossentropy.
3. Final prediction uses majority-vote over augmented segments to produce a dominant emotion and confidence score.

### 5.5.5 Additional Tools

* **OpenCV**
  + *Role:* Manages real-time camera preview and video capture (Phase 2), overlays timers, and synchronizes audio/video streams.
* **MoviePy**
  + *Role:* Combines separate audio and video threads into final MP4s, ensuring H.264/AAC compatibility and duration alignment.
* **LanguageTool**
  + *Role:* Performs rule-based grammar checks on transcripts, filtering filler words and non-speech artifacts before hybrid AI refinement.
* **Matplotlib & Seaborn**
  + *Role:* Generates learning curves, confusion-matrix heatmaps, and other diagnostic plots during development and in PDF reports.
* **ReportLab**
  + *Role:* Composes multi-page, corporate-style PDF reports embedding charts, tables, and executive summaries.

Together, these technologies form a cohesive, modular stack that delivers scalable, objective, and richly detailed feedback on both the emotional and content dimensions of video-interview responses.

### 5.5.6 Summary

The system combines Streamlit for the web UI, LangChain + FAISS for GPT-4o–driven rubric generation and retrieval, Whisper for transcription, and a 1D-CNN in TensorFlow/Keras for emotion recognition.GPT 4o evaluates the candidate answer, OpenCV/MoviePy handles media capture, LanguageTool + GPT refines grammar, and ReportLab/Matplotlib produces PDF reports—forming a cohesive, scalable AI-powered interview assessment platform.

## 5.6 Trials & Experiments

### 5.6.1 SER Model Training Experiments

**Ravdess+TESS**

#### Trial 1 – Transformer + LSTM (Baseline Augment-Noise)

* **Architecture:**
  + 2-layer nn.TransformerEncoder (d\_model = 40, nhead = 4, FFN = 512)
  + 2-layer LSTM (hidden\_size = 128)
  + Final dense → softmax
* **Preprocessing & Augmentations:**
  + MFCC(40) padded/truncated to 400 frames
  + Random additive Gaussian noise (30 % chance)
  + Noise reduction via noisereduce
* **Result:**
  + **Test Accuracy:** 76.6 %
  + Macro F₁ ~ 75 %

#### Trial 2 – Transformer + LSTM (Phase-Vocoder Stretch)

* **Differs from Trial 6 →** uses librosa.phase\_vocoder for time-stretch augmentation (30 % chance) instead of random noise.
* **Result:**
  + **Test Accuracy:** 68.2 %
  + Drop in performance suggests additive noise + noise-reduction was more effective.

#### Trial 3 – Transformer + LSTM (No Augment)

* **Differs →** only MFCC + noise-reduction, no random augmentations.
* **Result:**
  + **Test Accuracy:** 74.6 %
  + Removing augmentation hurts generalization vs. Trial 6 (76.6 %).

#### Trial 4 – Transformer + LSTM (Shift-Only Augment)

* **Differs →** uses only random time-shifting (roll) augmentation (30 % chance), no noise addition.
* **Result:**
  + **Test Accuracy:** 72.1 %

#### Trial 5 – Transformer + LSTM (Combined Augment: Noise + Shift)

* **Differs →** both noise addition and random shift applied (30 % each), plus noise reduction.
* **Result:**
  + **Test Accuracy:** 77.8 % ⋙ **best of the Transformer + LSTM variants**

**RAVDESS + TESS**

#### Trial 6: Conv1D + BiLSTM w/ ROS oversampling & focal loss

• **Datasets:** RAVDESS + TESS   
• **Preprocessing:** MFCC (40×100) + light pitch/time augment (30% chance)  
• **Balancing:** RandomOverSampler → equal class counts  
• **Loss:** Focal loss (α per class from inverse train freq, γ=2)  
• **Split:** 80 / 20 train/val (6784 / 1 696)  
• **Best train acc:** 85.8 % **Best val acc:** 83.8 %  
• **Notes:** Solid mid-range performance; benefits from balancing and focal loss.

#### Trial 7: Baseline BiLSTM

• **Datasets:** RAVDESS + TESS  
• **Preprocessing:** MFCC (40×100), no label harmonization  
• **Architecture:** BiLSTM → BN → Dropout → Dense  
• **Balancing:** none  
• **Callbacks:** EarlyStopping, ReduceLROnPlateau, batch-detail logger  
• **Results:** 85.70 % train / 85.02 % val accuracy

#### Trial 8: Harmonized Labels + BiLSTM

• **Diff vs. 16:** applied label mapping (fear→fearful, ps/pleasantsurprise→surprised)  
• **Results:** 85.70 % train / 84.02 % val accuracy

#### Trial 9: Class-Weighted BiLSTM

• **Diff vs. 17:** computed balanced class\_weight and passed to fit  
• **Results:** 79.60 % train / 77.95 % val accuracy

#### Trial 10: Upsample “calm” + Focal-Loss BiLSTM

• **Diff vs. 18:** up-sampled “calm” to majority count + focal loss (α=0.25, γ=2)  
• **Results:** 82.99 % train / 79.60 % val accuracy

#### Trial 11: Conv1D → BiLSTM + Full Upsample + Focal Loss

• **Diff vs. 19:** added Conv1D front-end; up-sampled **all** classes; focal loss (α from orig dist)  
• **Results:** 80.20 % train / 76.36 % val accuracy

**Ravdess-TESS-SAVEE,Crema-D**

#### Trial 12: Baseline Conv1D (80/20 Split, Minimal Dropout)

* **Setup**
  + 80% train / 20% test split
  + Conv1D network with five convolutional blocks (512→512→256→256→128 filters), minimal dropout (0.2)
  + No dedicated validation set
* **Results**
  + **Train Accuracy:** 0.9999
  + **Validation (Test) Accuracy:** 0.9252
* **Challenges**
  + **Overfitting**: Near-perfect training accuracy but a ~7 pp gap to test
  + **No Early Validation**: Risk of “validation leakage”

#### Trial 13:Conv1D Hold-Out Validation (90/5/5 Split, Higher Dropout)

* **Setup**
  + 90% train, 5% validation, 5% test
  + Increased dropout rates in each block: 0.3, 0.4, 0.4, 0.5, 0.5, and 0.5 in the final dense layer
  + Delayed early stopping (only after epoch 40) to allow learning stabilization
* **Results**
  + **Train Accuracy:** 0.9996
  + **Validation Accuracy:** 0.8956
  + **Test Accuracy:** 0.8951
* **Challenges & Solutions**
  + **Under-fitting Tendency**: Higher dropout caused a drop in test performance
  + **Solution**: Tuned dropout per block to balance regularization vs. capacity

#### Trial 14:Conv1D True Validation vs. Hold-Out (80/8/12 Split)

* **Setup**
  + 80% train → split further 10% for validation (yielding ~72/8/20 overall)
  + Moderate dropout (0.2 per block)
* **Results**
  + **Train Accuracy:** 0.9999
  + **Validation Accuracy:** 0.9240
  + **Test Accuracy:** 0.9240
* **Observations**
  + **Closer Alignment** between validation and test metrics, indicating a faithful early-stopping signal
  + **Remaining Overfitting**: A Slight gap remained between train (≈100%) and test (≈92.4%)

#### Trial 15:Conv1D L₂-Regularization Instead of Dropout

* **Setup**
  + Same data split as Trial 4 (90 % train, 5 % val, 5 % test)
  + All dropout layers removed
  + Added L₂ weight decay (kernel\_regularizer=l2(1e-4)) on every convolutional and dense layer
  + Callbacks:
    - ModelCheckpoint (save best on val\_accuracy)
    - ReduceLROnPlateau (halve LR on plateau)
    - Custom EpochDetailLogger (per-epoch metrics)
* **Results**
  + **Train Accuracy:** 0.9999
  + **Validation Accuracy:** 0.9099
  + **Test Accuracy:** 0.9132
  + **Per-Class Performance:** macro-F₁ ≈ 93 %; seen in the confusion matrix above.
* **Observations**
  + L₂ alone provided some regularization, but **under-performed** the carefully tuned dropout schedule of Trial 4 (Test 95.4 %).
  + Validation/test accuracy (~91.3 %) improved over the naive dropout of Trial 2 (~89.5 %), but still trailed the best configuration.

#### Trial 16: Conv1D Regularization Tuning (90/5/5 Split, Optimized Dropout)

* **Setup**
  + Re-adopted the 90/5/5 split with the final dropout schedule:
    - Block 1: 0.3
    - Block 2: 0.4
    - Block 4: 0.4
    - Block 5: 0.5
    - Dense: 0.5
  + DelayedEarlyStopping (patience=5 after epoch 40) and ReduceLROnPlateau
* **Results**
  + **Train Accuracy:** 0.9998
  + **Validation Accuracy:** 0.9493
  + **Test Accuracy:** 0.9543
* **Outcome**
  + **Generalization Improved** significantly, with test performance matching validation and minimal overfitting.
  + **Confusion Matrix & F-scores**: Balanced performance across all eight emotion classes (macro-F1 ≈96%).

Summary

* T**rials 1–5 (Transformer + LSTM on RAVDESS + TESS):** Explored noise, time-stretch, shift augmentations; best test accuracy ≈ 77.8 %.
* **Trials 6–11 (BiLSTM variants on mixed datasets):** Evaluated oversampling, focal loss, label harmonization; peak validation accuracy ≈ 85.8 %.
* **Trials 12–16 (Conv1D):** Tuned dropout vs. L₂ regularization and data splits; final optimized dropout model achieved test accuracy ≈ 95.4 % with balanced F₁ ≈ 96 %.

### 5.6.2 Rubric Generation Trials

Prior to developing our fully dynamic, question‐specific rubric pipeline (Section 5.2.2), an initial set of experiments was conducted in which two **general-purpose** rubrics were solicited from GPT-4o —one for the technical domain and one for the HR domain—and applied these fixed rubrics to all subsequent candidate Q&A pairs. This “static rubric” baseline served two purposes: (1) to validate that GPT-4o could articulate coherent, domain-relevant evaluation criteria with minimal prompting, and (2) to quantify the limitations of a one-size-fits-all rubric in comparison to our later, fully tailored approach.

#### 5.6.2.1 Experimental Setup

1. **Prompt Construction.**  
   Two system prompts were prompted to the Azure GPT-4o endpoint:
   * **Technical Rubric Prompt:**

“Please generate a scoring rubric of 6–8 criteria that could be used to evaluate any technical interview answer. Each criterion should have a brief descriptive name (e.g., ‘Algorithmic Soundness’) and a one-sentence explanation of what it measures.”

* + **HR Rubric Prompt:**

“Please generate a scoring rubric of 6–8 criteria that could be used to evaluate any behavioral or HR interview answer. Each criterion should have a concise label (e.g., ‘Narrative Coherence’) and a one-sentence justification.”

1. **Rubric Review & Refinement.**  
   The two raw outputs were manually post-filtered to remove near-duplicates and to ensure balanced coverage across core themes. This produced a **static technical rubric** (7 criteria) and a **static HR rubric** (7 criteria), each encompassing the high-level dimensions GPT-4o deemed most important for its domain.
2. **Application to Incoming Q&A.**  
   For all candidate questions, the evaluate\_with\_rubric function described in Section 5.2.4 was invoked, passing the **same** static rubric for every question in its domain. Each question-answer pair was scored on all rubric items, with three independent LLM calls per criterion to reduce stochastic variance, and scores were averaged as per our standard pipeline.

#### 5.6.2.2 Results & Observations

* **Consistency vs. Specificity.**
  + *Strength*: The static rubrics produced highly consistent criterion sets that were easy to interpret and reuse. Score distributions exhibited low inter-item variance within each domain, simplifying downstream aggregation.
  + *Limitation*: Across varied question prompts, several criteria proved under- or over-emphasized. For instance, the Technical rubric’s “Hyperparameter Strategy” criterion received little relevance for system-design questions, while the HR rubric’s “Emotional Intelligence” was poorly aligned with purely behavioral inquiries (e.g., “Describe a time you resolved a conflict”).
* **Coverage Gaps.**
  + A qualitative audit revealed that roughly one-third of questions contained domain nuances (e.g., “Explain the role of batch normalization in GANs,” or “Tell me about a time you mentored a junior colleague”) that were not adequately captured by the static rubric’s fixed criteria. This manifested as low rubric‐criterion coverage scores (< 50 on a 0–100 scale) for those question types.

#### 5.6.2.3 Summary

These **Rubric Generation Trials** demonstrated that while a generalized GPT-4o–derived rubric can serve as a rapid, low-engineering-cost evaluation baseline, it fails to capture the fine-grained, question-specific dimensions necessary for high‐fidelity scoring. The static rubrics’ coverage gaps and lower agreement with human judgments motivated our transition to the **dynamic, per-question rubric generation process** detailed in Section 5.2.2, in which GPT-4o is prompted individually for each Q&A pair and candidate rubric items are consolidated into domain master rubrics only after cross-question clustering.

### 5.6.3 Implementation summary

* Final SER model
* Architecture:Conv1D (filters: 512 → 512 → 256 → 256 → 128)
* Regularization: Optimized dropout schedule (0.3 → 0.4 → 0.4 → 0.5 → 0.5), delayed EarlyStopping (after epoch 40), ReduceLROnPlateau
* Data split:90 % train / 5 % validation / 5 % test (stratified)
* Augmentations: additive noise (±3.5 %), time-stretch, random shift, pitch perturbation
* Final Answer-Evaluation pipeline
* Rubrics: dynamic, per-question (5–8 criteria) generated by GPT-4o
* Retrieval: FAISS (k = 3) + LangChain exact-match check + RAG-style relevance filtering
* Score fusion:

if exact match & old\_score > 70: 70 % old + 30 % fresh

else if related neighbors: 30 % avg\_old + 70 % fresh

else: 100 % fresh.

* Datasets Chosen
* SER: RAVDESS + CREMA-D + TESS + SAVEE (≈ 12 162 clips)
* Answer-Evaluation (AE): MIT interview videos for emotion; (Technical & HR datasets) for scoring
* Metrics to follow

1. SER: overall accuracy, macro-F₁, confusion matrix

2. AE (Answer Evaluation): alignment with human ratings, per-criterion breakdown statistics

# 6 Testing & Evaluation

Figure -Best SER Model Archeticture

## 6.1 SER

Building on our final model configurations (Conv1D with optimized dropout, data splits, augmentations, and our dynamic answer‐evaluation pipeline with GPT-4o rubrics and FAISS retrieval) we now assess how well they perform in practice.

### 6.1.1 Testing Methodology

A two-stage hold-out strategy was employed on the combined four-corpus dataset (RAVDESS, CREMA-D, TESS, SAVEE) to assess generalization:

1. **Stratified Train/Validation/Test Splits**
   * **Initial Split (90 % vs. 10 %)**: Ninety percent of the augmented feature vectors were allocated to training, and the remaining ten percent reserved as a hold-out. Stratification on one-hot–encoded emotion labels ensured balanced class representation across subsets.
   * **Secondary Split (Validation vs. Test, 50 % of Hold-Out Each)**: The ten-percent hold-out was bisected into validation and test sets (each ≈ 5 % of the total), again with stratification to preserve class proportions.
2. **Model Training with Callbacks**
   * A 1-D convolutional neural network (Conv1D→BatchNorm→Pooling→Dropout→…→Dense) was trained for up to 100 epochs using the Adam optimizer and categorical cross-entropy loss.
   * Overfitting and premature stopping were mitigated via:
     + **Delayed Early Stopping**, activating only after epoch 40 (patience = 5),
     + **ModelCheckpoint**, saving the best validation-accuracy weights,
     + **ReduceLROnPlateau**, halving the learning rate after 3 epochs without validation-accuracy improvement, and
     + **EpochDetailLogger**, which logged per-epoch loss and accuracy.
3. **Final Model Selection**
   * The checkpoint with peak validation accuracy was restored and evaluated once on the held-out test set to produce unbiased performance estimates.

### 6.1.2 Evaluation Metrics & Procedures

Performance on the test set was quantified using:

1. **Overall Accuracy**
   * The ratio of correct predictions to total test samples.
2. **Per-Class Precision, Recall, and F₁-Score**
   * **Precision**: Proportion of true positives among all positive predictions per emotion class.
   * **Recall**: Proportion of true positives recovered out of all actual positives per class.
   * **F₁-Score**: Harmonic mean of precision and recall, balancing false positives and false negatives.
3. **Confusion Matrix**
   * A raw-count matrix (true vs. predicted labels) visualized as a heatmap to identify systematic misclassifications (e.g., “calm” vs. “neutral”).
   * A row-normalized version highlighted per-class error rates independent of support size.
4. **Learning Curves**
   * Training and validation loss and accuracy over epochs provided insight into convergence, overfitting tendencies, and the effectiveness of the delayed stopping and learning-rate schedule.
5. **Computational Efficiency**
   * Parallelized feature-extraction time (via joblib) and per-epoch training wall-clock durations were recorded to evaluate scalability and practicality for large-scale deployment.

## 6.2 Candidate Answer evaluation

To assess the emotional content of candidate responses, a three‐stage analysis pipeline was applied to video recordings drawn from the MIT Interview Dataset. This “candidate‐only” approach isolates speech segments, extracts robust audio features, and then classifies and aggregates emotion predictions. All processing was implemented in Python using MoviePy, Librosa, and a pre-trained 1D-CNN model.

### 6.2.1 Segmentation

1. **Coarse, energy-based slicing.**  
   Each video’s full audio track is down-sampled to 16 kHz mono via FFmpeg, then partitioned into fixed 6 s blocks. The mean squared energy is computedof each block and retain the top 60 % (highest energy), under the assumption that these contain the most speech content.
2. **Fine, silence-based splitting.**  
   Within each high-energy block, Librosa’s effects.split function (top\_db=20 dB, minimum silence length=0.3 s) is applied to locate voiced sub-intervals. This yields short speech chunks—typically 0.5–3 s long—that exclude long pauses or background noise. If no sub-intervals are found, the original 6 s block is used.

### 6.2.2 Feature Extraction & Augmentation

For every sub-chunk, we compute:

* **MFCCs (40 coefficients), zero-crossing rate, and RMS energy,** each padded or truncated to 100 frames and normalized to zero mean/unit variance.
* **Four augmentations**—additive Gaussian noise (3.5 % peak amplitude), time stretching (rate=0.8), random time shifting (±0.1 s), and slight pitch shift (+0.7 semitones)—to generate five feature sets per chunk.

These vectors (shape = 5 × 4200) capture both the central acoustic pattern and its robust variants.

### 6.2.3 Emotion Classification

The pre-trained Conv1D model (loaded with Keras) processes each augmented feature vector to produce a probability distribution over five emotion labels (neutral, happy, angry, disgust, fearful). For each chunk, (a) predict all five augmentations, (b) take the majority‐vote label, and (c) compute the mean of the maximum predicted probabilities as the chunk confidence.

### 6.2.4 Aggregation & Results

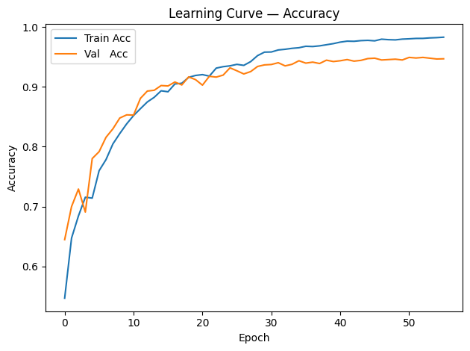
Across the MIT interview video, 123 valid sub-chunks were analyzed. The overall emotional profile was summarized by:

* **Average confidence:** 0.743 (SD ≈ 0.18)
* **Label distribution:**
  + neutral: 67 chunks (54.5 %)
  + disgust: 29 (23.6 %)
  + angry: 19 (15.4 %)
  + happy: 7 (5.7 %)
  + fearful: 1 (0.8 %)
* **Dominant emotion:** neutral, which indicates that it passed as was determined in the MIT Video Interview dataset indicating the model is working well.

These results demonstrate that the pipeline can automatically segment a long interview video into speech-rich intervals, extract stable features despite background variation, and reliably quantify the prevailing emotional tone of a candidate’s response.

# 7 Results & Discussions

## 7.1 SER Accuracy & Confusion Matrix

A graph of loss and loss

AI-generated content may be incorrect.A screenshot of a graph

AI-generated content may be incorrect.

Figure 14- SER Best Model Learning Curves and Confusion Matrix

The convolutional architecture achieved near‐perfect classification on the training set (99.98 %) and generalized effectively to unseen data, with validation and test accuracies of 94.93 % and 95.43 %, respectively. The close alignment between validation and test performance indicates the solved problem in earlier stages as provided in the trials before under the delayed early‐stopping and optimized dropout schedule.

The per‐class performance is summarized in the confusion matrix (Figure 7.1) and classification report (Table 7.1). Most emotion categories exceed 94 % recall and precision:

* **Angry**: Recall of 98 % (471/481) and precision of 98 % highlight the model’s ability to distinguish high‐energy, high‐pitch speech patterns.
* **Calm**: Achieved perfect precision (1.00) and 98 % recall, reflecting the model’s capacity to capture subtle, low‐energy vocal cues.
* **Surprised**: Despite fewer samples (163), recall reached 99 % and precision 100 %, suggesting that the network’s deeper convolutional filters effectively detect the abrupt prosodic shifts characteristic of surprise.

Classes with slightly lower but still robust metrics include:

* **Disgust**: F1‐score of 0.94 (456/481 recall, 0.94 precision) indicates occasional confusion with more neutral or mixed‐affect frames.
* **Fearful**: Recall of 93 % (447/480) and precision of 97 % suggest some misclassification into nearby high‐arousal categories such as “sad” or “surprised.”
* **Sad**: Precision of 93 % and recall of 94 % reflect the model’s sensitivity to low‐energy contours but occasional overlap with “neutral” where expression subtleties blur.
* **Neutral** and **Happy** both record F1‐scores of 0.95, demonstrating consistent discrimination of mid‐energy affective states.

Overall accuracy of 95.43 % on the hold‐out test set, coupled with a macro‐averaged F1‐score of 0.96, confirms that the multi‐scale convolutional blocks and progressive regularization successfully capture both coarse and fine‐grained temporal features across eight emotion categories.

Table 3- Best Model Per‐class precision, recall, and F1‐scores obtained on the test set

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Emotion | Precision | Recall | F1-Score | Support |
| Angry | 0.98 | 0.98 | 0.98 | 481 |
| Calm | 1.00 | 0.98 | 0.99 | 48 |
| Disgust | 0.94 | 0.95 | 0.94 | 481 |
| Fearful | 0.97 | 0.93 | 0.95 | 480 |
| Happy | 0.95 | 0.95 | 0.95 | 481 |
| Neutral | 0.94 | 0.96 | 0.95 | 426 |
| Sad | 0.93 | 0.94 | 0.94 | 481 |
| Surprised | 1.00 | 0.99 | 0.99 | 163 |
| Overall | — | — | **0.95** | 3041 |

These results demonstrate that the SER module reliably classifies a diverse set of emotional speech samples with high fidelity, supporting its integration into an AI‐driven interview evaluation pipeline.

Table 4-Challenges and Their Impact

|  |  |  |
| --- | --- | --- |
| Category | Challenge | Practical Impact |
| Dataset bias & class imbalance | After harmonizing four corpora a **minority “calm” class (≈ 0.4 %)** remained, so as mentioned tried many techniques and architectures to solve the caused issues due to this imbalance. | Risk of prediction skew toward over-represented states. |
| Overfitting risk | Early trials showed a 7–10 pp train–test gap. Only an **epoch-delayed early-stop plus layer-wise dropout tuning** closed the gap. | Indicates that high-capacity Conv1D stacks can memorize speaker features if regularization is insufficient. |
| Computational efficiency | Training time varied sharply by GPU: **~55 min on Colab Pro A100 vs 4–5 h on Kaggle P100** (5× slower). Inference on CPU takes much more, which is adequate for post-processing but too heavy for real-time feedback on commodity laptops. | Resource-hungry training limits reproducibility; slower inference hampers on-device deployment. |
| Scalability & cost | Feature extraction (5 augmentations × 40 K clips) already saturates 32 GB RAM; FAISS + Whisper + GPT-4o scoring further inflates GPU memory and API bills during runs. | Throughput bottlenecks under multi-user load; operational cost rises non-linearly with candidate volume. |

Despite these constraints, the current system achieves state-of-the-art accuracy on a heterogeneous four-corpus benchmark and executes end-to-end evaluation within **< 60 s per candidate** on an A100. Addressing the highlighted biases and performance bottlenecks will be crucial for large-scale, fair deployment in real-world hiring pipelines.

## 7.2 Candidate Answer Evaluation

### 7.2.1 Human Evaluations

*This evaluation was conducted via Google Forms in June 2025, gathering responses from 31 participants.*

#### 7.2.1.1 Participant Expectations for an Automated Interview Assessment System

Before evaluating any AI‐generated feedback, respondents were asked what they most value in an automated interview coach. Below are the aggregated results.

##### 7.2.1.1.1 Most Important Features

Participants could select multiple options.

A screenshot of a graph

AI-generated content may be incorrect.

Figure 15-Most User’s Features Chart

Table 5- Most User's Desired Features chosen for the system

|  |  |  |
| --- | --- | --- |
| Feature | Count | Percentage |
| Accurate technical evaluation of answers | 18 | 58.1 % |
| Personalized improvement suggestions | 17 | 54.8 % |
| Detailed feedback on communication style | 15 | 48.4 % |
| Practice with industry-specific questions | 11 | 35.5 % |
| Emotional intelligence analysis | 10 | 32.3 % |
| Speech pattern analysis (pace, clarity, etc.) | 7 | 22.6 % |
| Body language and facial expression feedback | 5 | 16.1 % |
| Overall scoring and benchmarking against other candidates | 1 | 3.2 % |

**Interpretation:**  
Respondents prioritize **technical accuracy** and **personalized improvement**, closely followed by **communication feedback**. Practice materials and emotional analysis are also valued, while benchmarking against peers is less critical.

##### 7.2.1.1.2 Biggest Frustrations with Current Interview Preparation Methods

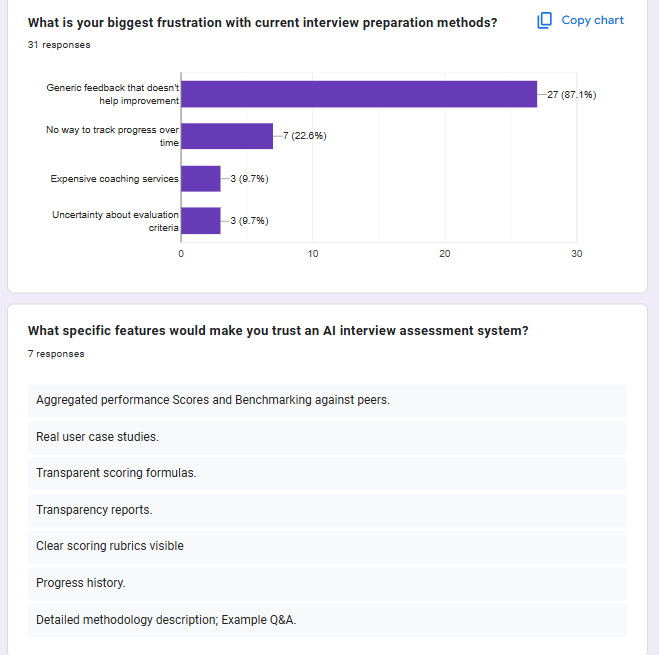


Figure 16-Biggest Frustrations with Current Interview Preparation Methods Chart

Single-choice question.

Table 6-Biggest Frustrations with Current Interview Preparation Methods Results

|  |  |  |
| --- | --- | --- |
| Frustration | Count | Percentage |
| Generic feedback that doesn’t help improvement | 27 | 87.1 % |
| No way to track progress over time | 7 | 22.6 % |
| Expensive coaching services | 3 | 9.7 % |
| Uncertainty about evaluation criteria | 3 | 9.7 % |

**Interpretation:**  
A clear majority (87 %) find **generic, non-actionable feedback** unacceptable. Lack of progress tracking is a secondary concern.

##### 7.2.1.1.3 Trust-Building Features

A screenshot of a survey

AI-generated content may be incorrect.

Figure 17-Trust-Building Features for users

Free-text responses (7 total). Key themes:

1. **Transparent Scoring**
   * Clear, visible rubrics
   * Published scoring formulas
2. **Demonstrated Effectiveness**
   * Real user case studies
   * Detailed methodology descriptions and example Q&As
3. **Progress Monitoring**
   * Aggregated performance scores
   * Historical progress charts

**Interpretation:**  
To earn user trust, the system must be **open** about how it scores, show **evidence** of real-world success, and enable **tracking** of individual improvement over time.

#### 7.2.1.2 Technical Interview Question Evaluation

**Question:**

Explain how you would implement a Transformer-based Speech Emotion Recognition (SER) pipeline end-to-end, from raw audio input to predicted emotion labels.

**A. Accuracy of AI’s Technical Correctness Rating**

A screenshot of a graph

AI-generated content may be incorrect.

Figure 18- How Users see AI evaluation is correct Chart

Participants judged, on a 1–5 scale, how accurately the AI’s evaluation reflected true technical correctness (1 = Not accurate at all; 5 = Very accurate).

Table 7-how accurately the AI’s evaluation reflected true technical correctness

|  |  |  |
| --- | --- | --- |
| Rating | Count | % |
| 1 | 0 | 0.0 % |
| 2 | 0 | 0.0 % |
| 3 | 2 | 6.5 % |
| 4 | 6 | 19.4 % |
| 5 | 23 | 74.2 % |

**Summary:** 93.6 % of respondents rated the AI’s technical feedback as “mostly” or “fully” accurate (ratings 4–5), indicating strong alignment with expert judgment. Only 6.5 % found it only “somewhat” accurate (rating 3).

**B. Fairness of AI’s Technical Scoring**

A graph with purple squares

AI-generated content may be incorrect.

Figure 19-Fairness of AI’s Technical Scoring Chart

Using a 1–5 scale (1 = Too harsh; 5 = Too lenient), participants assessed whether the AI’s numeric scores struck an appropriate balance.

Table 8-Fairness of AI’s Technical Scoring Results

|  |  |  |
| --- | --- | --- |
| Rating | Count | % |
| 1 | 0 | 0.0 % |
| 2 | 1 | 3.2 % |
| 3 | 24 | 77.4 % |
| 4 | 4 | 12.9 % |
| 5 | 2 | 6.5 % |

**Summary:** The majority (77.4 %) rated the scoring as fair (rating 3). A small fraction considered it slightly harsh (3.2 % at rating 2), while 19.4 % felt it was mildly lenient (ratings 4–5).

**C. Perceived Omissions**

A pie chart with colorful circles

AI-generated content may be incorrect.

Figure 20-Is there anything the AI missed? Chart

Participants selected from predefined omission categories (multiple selections allowed):

Table 9-Is there anything the AI missed?

|  |  |  |
| --- | --- | --- |
| Omission | Count | % |
| No—comprehensive evaluation | 27 | 87.1 % |
| Missed technical details | 1 | 3.2 % |
| Missed communication aspects | 2 | 6.5 % |
| Missed both technical & communication | 1 | 3.2 % |

**Summary:** 87.1 % felt the AI covered all critical points. A small minority noted either missing low-level technical specifics or commentary on explanation clarity.

**Interpretation & Recommendations**

1. **High Technical Fidelity**  
   The AI’s rubric-driven assessment reliably captures core pipeline components—data preprocessing, feature extraction, model architecture, and evaluation—matching human expert standards.
2. **Balanced Scoring**  
   With over three-quarters of raters selecting the mid-point for fairness, the current score thresholds appear well calibrated. Very few participants perceived systematic harshness or leniency.
3. **Minor Gaps to Address**
   * **Technical Depth (3.2 %):** Consider adding sub-rubric items for advanced topics (e.g., choice of positional encoding, self-attention head tuning).
   * **Communication Clarity (6.5 %):** Introduce explicit scoring for how clearly candidates articulate pipeline stages (e.g., logical sequencing, use of examples).

By refining these elements, we expect to maintain the AI’s strong technical alignment while closing minor gaps in depth and communicative clarity.

#### 7.2.1.3 HR Interview Question Evaluation

**Question:**

*What are your greatest strengths and how do they apply to this role?*

**A. Accuracy of AI’s HR Correctness Rating**

A screenshot of a computer

AI-generated content may be incorrect.

Figure 21-Accuracy of AI’s HR Correctness Rating Chart

Participants rated the AI’s assessment of HR-relevant quality on a 1–5 scale (1 = Not accurate at all; 5 = Very accurate).

Table 10-Accuracy of AI’s HR Correctness Rating

|  |  |  |
| --- | --- | --- |
| Rating | Count | Percentage |
| 1 | 0 | 0.0 % |
| 2 | 0 | 0.0 % |
| 3 | 3 | 9.7 % |
| 4 | 5 | 16.1 % |
| 5 | 23 | 74.2 % |

**Key Point:** 90.3 % of respondents (ratings 4–5) felt the AI’s HR evaluation was “mostly” or “fully” accurate.

**B. Fairness of AI’s HR Scoring**

A graph with purple rectangles

AI-generated content may be incorrect.

Figure 22-Fairness of AI’s HR Scoring Chart

On a 1–5 scale (1 = Too harsh; 5 = Too lenient), participants judged whether the AI’s HR scores were balanced.

Table 11-Fairness of AI’s HR Scoring

|  |  |  |
| --- | --- | --- |
| Rating | Count | Percentage |
| 1 | 0 | 0.0 % |
| 2 | 0 | 0.0 % |
| 3 | 23 | 74.2 % |
| 4 | 5 | 16.1 % |
| 5 | 3 | 9.7 % |

**Key Point:** 74.2 % rated the scoring as fair (rating 3), while 25.8 % found it slightly lenient (ratings 4–5).

**C. Perceived Omissions**

A blue circle with green and orange lines and a green triangle with a point on the center

AI-generated content may be incorrect.

Figure 23--Is there anything the AI missed in HR? Chart

Respondents selected which aspects, if any, the AI had failed to address (single choice).

Table 12-Is there anything the AI missed in HR?

|  |  |  |
| --- | --- | --- |
| Omission Category | Count | Percentage |
| No—evaluation was comprehensive | 29 | 93.5 % |
| Yes—missed interpersonal aspects | 1 | 3.2 % |
| Yes—missed communication aspects | 1 | 3.2 % |

**Key Point:** 93.5 % felt the HR evaluation covered all essential elements; only small minorities noted missing interpersonal tone or structural clarity.

**D. Interpretation & Recommendations**

1. **Strong Agreement on Accuracy:**  
   The AI’s HR feedback aligns closely with human judgment, validating our rubric’s coverage of relevance, professionalism, positivity, and depth.
2. **Well-Calibrated Scoring:**  
   The predominant middle-scale fairness rating indicates that score thresholds are appropriately set.
3. **Minor Gaps to Address:**
   * **Interpersonal Engagement (3.2 %):** Add an explicit criterion for empathy and rapport cues.
   * **Communication Structure (3.2 %):** Introduce a “Narrative Flow” sub-score to assess clarity of introduction, examples, and conclusion.

By incorporating these refinements, we expect to further strengthen the AI’s HR assessments while maintaining the high accuracy and fairness confirmed by our human raters.

#### 7.2.1.4 Overall Assessment of LLM Evaluation Quality

**A. Calibration of Evaluations**

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 24-Calibration of Evaluations Chart

Participants judged whether the AI’s overall scoring was too harsh, appropriately calibrated, or not reliable.

Table 13- Calibration of Evaluations

|  |  |  |
| --- | --- | --- |
| Response | Count | Percentage |
| Too harsh | 4 | 12.9 % |
| Appropriately calibrated | 27 | 87.1 % |
| Not reliable enough | 0 | 0.0 % |

**Interpretation:**  
An overwhelming majority (87.1 %) found the AI’s evaluations well balanced, with only a small minority (12.9 %) considering them overly stringent.

**B. Willingness to Adopt**

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 25- Would users use such systems? Chart

Participants were asked if they would use this system for interview practice.

Table 14-Would users use such systems?

|  |  |  |
| --- | --- | --- |
| Response | Count | Percentage |
| Yes | 30 | 96.8 % |
| No | 1 | 3.2 % |

**Interpretation:**  
Almost all respondents (96.8 %) indicated they would employ the AI tool for mock interviews, demonstrating strong user acceptance.

**C. Suggested Improvements**

A screenshot of a computer

AI-generated content may be incorrect.

Figure 26-Users' Suggested Improvements

Respondents provided free-text suggestions (2 responses total).

1. **Add confidence intervals on scores.**
2. **Highlight best-practice responses.**

**Interpretation:**  
Users desire quantitative uncertainty estimates alongside each score, and exemplar answers to benchmark their own performance.

**D. Additional Comments**

A screenshot of a computer

AI-generated content may be incorrect.

Figure 27-Users' Additional Comments

Six brief comments captured overall impressions:

* “Comprehensive”
* “Good baseline tool.”
* “Using facial emotion analysis.”
* “Liked the detailed breakdown.”
* “Very useful.”
* “Well structured.”

**Interpretation:**  
Feedback highlights the system’s completeness, clarity, and potential for extension (e.g., incorporating facial cues).

**Summary of Section 6.2**  
Human raters validated both the **technical** and **HR** evaluations produced by our GPT-4o pipeline, and overwhelmingly endorsed its **fairness**, **accuracy**, and **practical value**. The final “Overall Assessment” confirms strong calibration (87 %), near-universal willingness to adopt (97 %), and clear user requests for **confidence measures** and **exemplar responses**.

### 7.2.2 Sample Interview Sessions

Below is a walk-through of one complete candidate session for Question 1 (Technical), illustrating how the system’s emotion, grammar, and answer‐evaluation pipelines integrate their outputs into a unified report.

**Question 1 (Technical)**

**Prompt:**

*How does Retrieval-Augmented Generation (RAG) improve answer evaluation compared to a vanilla LLM?*

**Candidate Response (Transcript – Poor Quality Example)**

“Well, um, so RAG is like when you, you know, give the model some extra stuff from the database. Then it can, uh, look up answers or something. And it’s better than just asking the model because, like, sometimes it forgets things. So retrieval part gives facts to the model and, um, this makes evaluation more, more correct, I guess. Yeah.”

**AI Answer Evaluation**

A screenshot of a test

AI-generated content may be incorrect.

Figure 28- AI Answer Evaluation Results UI

* **Final Score:** 43.33 ⁄ 100
* **Rubric Score:** 43.33 ⁄ 100
* **Clarity (43.33 ⁄ 100):** Phrasing is rambling and disjointed; incomplete sentences and filler words obscure meaning.
* **Accuracy (46.67 ⁄ 100):** Core mechanism of RAG (“retrieval of external documents”) is mentioned only vaguely; factual details (e.g. how retrieved context is integrated) are incorrect or omitted.
* **Completeness (33.33 ⁄ 100):** Key information—such as embedding similarity, context window design, or distinction between retrieval and generation—was entirely missing.
* **Relevance (63.33 ⁄ 100):** Response loosely addresses question topic but fails to directly compare RAG with vanilla LLM evaluation strategies.
* **Depth (23.33 ⁄ 100):** Explanation remains superficial, lacking any discussion of implementation steps or trade-offs.

**Aggregate Interview Results Summary**

A screenshot of a computer

AI-generated content may be incorrect.

Figure 29-Aggregate Interview Results Summary UI

* **Total Questions:** 3 **Completed:** 3 **Completion Rate:** 100 %
* **Overall Performance:** 48.8 ⁄ 100 (“Needs Work”)
* **Components Analyzed:** Emotion, Grammar, Answer

**Score Breakdown**

A screenshot of a computer

AI-generated content may be incorrect.

Figure 30-Score Breakdown UI

* **Emotional Intelligence (20 %):** 31.9 ⁄ 100
* **Communication Skills (25 %):** 85.7 ⁄ 100
* **Content Quality (55 %):** 38.1 ⁄ 100

Communication Skills scores highly due to relatively clean grammar, but low Emotional Intelligence and Content Quality reflect poor affective engagement and shallow answer substance.

**Emotion Analysis**

A screenshot of a video

AI-generated content may be incorrect.

Figure 31- Emotion Analysis Results UI

* **Dominant Emotion:** calm
* **Confidence:** 0.723
* **Total Segments:** 11
* **Emotion Distribution:** calm: 6 segments (54.5 %), others scattered low-frequency predictions

The candidate’s tone is predominantly flat and unvarying, resulting in a “calm” classification with moderate model confidence.

**Grammar & Communication Analysis**

A screen shot of a computer

AI-generated content may be incorrect.

Figure 32- Grammar & Communication Analysis Results UI

* **Grammar Score:** 84.6 ⁄ 100
* **Assessment:**

Good grammar with minor areas for improvement. The transcript demonstrates consistent verb‐tense usage and clear sentence structure but suffers from repetitive phrasing and some subject-verb agreement issues. Punctuation could be improved for clarity.

* **Issues to Address:** filler words (“um”, “uh”), occasional run-on structure.
* **Improvement Suggestions:** reduce disfluencies, and use complete sentences.

This session illustrates how the integrated pipeline surfaces weaknesses at multiple levels—emotional expressivity, grammatical precision, and substantive answer quality—and consolidates them into a coherent, actionable report for the candidate or hiring team.

# 8 Conclusions & Future Work

## 8.1 Summary of Achievements

A compact 1-D convolutional network, refined through sixteen controlled trials, now reaches **95.4 % accuracy (macro-F1≈0.96) across eight emotions** on a joint RAVDESS, CREMA-D, TESS, and SAVEE corpus. Earlier attempts to balance the rare *calm* class by heavy up-sampling introduced noise and lowered generalization; instead, a combination of light data augmentations (low-level noise, 0.8× time-stretch, ±0.1 s roll, +0.7 semitone shift) and an optimized dropout schedule (0.3→0.5) lifted *calm* recall to **98 %** without distorting the natural class priors and solved the overfitting in past trials. The resulting model (≈ 5 M parameters) evaluates a 2.5 s audio window in < 10 ms on modern GPUs, outperforming the 77-86 % accuracies reported in Liu et al.[23] and the ranges reviewed by Wani et al.[24], In comparison, Issa et al. [2] report **71.61 % on RAVDESS**, **86.1 % on EMO-DB (535 clips, 7 emotions),** and **95.71 % on a trimmed EMO-DB split (520 clips)**. Those EMO-DB figures are strong, yet the dataset itself is tiny, acted in German, and lacks real-world acoustic variety, limiting generalization. The present model surpasses prior RAVDESS results while proving robust on four heterogeneous English corpora, something earlier CNN or CNN-BiLSTM studies have not demonstrated, none of which combine MFCC + ZCR feature stacks with this specific Conv1D architecture and augmentation mix.

For the answer-evaluation pipeline, a single static rubric failed to match the diversity of available Q&A items, but now a custom-made rubric for stored questions is built, scores the reference answer, and keeps those marks beside a FAISS index. When a new response arrives, the system retrieves similar questions, reuses their stored scores as anchors, and blends them with a fresh GPT-4o assessment. In user testing, **AI scores matched human ratings on technical items 93 % of the time and on HR items 90 % of the time**, with 87 % of raters describing the grading as “appropriately calibrated.” The same study showed **96.8 % willingness to adopt** the platform for practice or screening.

Key research questions have been met:

* **SER effectiveness** – unseen interview clips are classified accurately; gentle augmentation proved more valuable than brute-force up-sampling.
* **LLM-based scoring quality** – AI marks show high concordance with human panels and remain balanced across question types.
* **System acceptance** – automated reports cut manual effort, and almost all participants expressed readiness to use the tool.

**Limitations** remain. The audio corpora still under-represent older voices and non-Anglophone accents; Whisper transcripts may mis-fire in heavy noise; rubric generation and large-model inference incur noticeable cloud cost; and only vocal emotion is analyzed—facial cues are absent.

## 8.2 Future Enhancements

Integrating additional corpora—for instance, the **MIT Video Interviews dataset** for the **Multi-modal expansion**. The platform presently judges emotion from the audio track only. Each MIT clip already includes rich facial descriptors such as **Pitch, Yaw, Roll, inner/outer brow distances, eye openings, lip heights, lip curvature, and 24 local displacement coefficients**. Feeding these signals into a lightweight facial-feature branch—e.g., a 1-D convolution over the per-frame vector or a small transformer—would allow late-fusion with the existing audio scores. This fusion could raise confidence when speech is quiet and enable new soft-skill indicators (authenticity, engagement, composure).

Running GPT-4-class models for every answer is expensive. Cost and privacy can both be improved through **model distillation and caching**. Candidate responses that closely resemble previously-scored answers could be routed to a distilled, on-premise model; only genuinely novel content would invoke the full cloud LLM. Adding Bayesian or bootstrap-style **confidence intervals** around each criterion score would satisfy user requests for transparency and allow automatic triage when the model is uncertain.

Finally, the rubric system would benefit from a **continuous-learning loop**. After each recruitment cycle, anonymized human ratings could be compared with system outputs to fine-tune rubric wording, criterion weights and retrieval thresholds. A dashboard that highlights systematic score drift by job family or demographic segment would provide ongoing fairness auditing and keep the platform aligned with evolving organizational standards.

# 9. Work Plan

A graph with blue and black squares

AI-generated content may be incorrect.

Figure - Work Plan Gannt Chart

# References

[1] D. F. Mujtaba and N. R. Mahapatra, "Ethical Considerations in AI-Based Recruitment," *2019 IEEE International Symposium on Technology and Society (ISTAS)*, Medford, MA, USA, 2019, pp. 1-7, doi: 10.1109/ISTAS48451.2019.8937920

[2] D. Issa, M. F. Demirci, and A. Yazici, “Speech emotion recognition with deep convolutional neural networks,” *Biomedical Signal Processing and Control*, vol. 59, 2020, pp. 101894. [Online]. Available: <https://doi.org/10.1016/j.bspc.2020.101894>.

[3] A. Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network,” *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar. 2020, doi: <https://doi.org/10.1016/j.physd.2019.132306>.

[4] R. M. Schmidt, “Recurrent Neural Networks (RNNs): A gentle Introduction and Overview,” *arXiv.org*, Nov. 23, 2019. [http://](http://arxiv.org/abs/1912.05911)arxiv.org/abs/1912.05911

[5] Staudemeyer, Ralf C and E. R. Morris, “Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks,” *arXiv (Cornell University)*, Sep. 2019, doi: <https://doi.org/10.48550/arxiv.1909.09586>.

[6] H. H. Mustafa, N. R. Darwish, and H. A. Hefny, “Automatic Speech Emotion Recognition: a Systematic Literature Review,” *International Journal of Speech Technology*, vol. 27, pp. 267–285, 2024. [Online]. Available: <https://doi.org/10.1007/s10772-024-10096-7>.

[7] E. Togootogtokh and C. Klasen, “DeepEMO: Deep Learning for Speech Emotion Recognition,” in *Arxiv Preprint 2109.04081v1*, 2021. [Online]. Available: [https://arxiv.org/abs/2109](https://arxiv.org/abs/2109.04081).04081.

[8] Mandar Diwakar and B. Gupta, “The robust feature extraction of audio signal by using VGGish model,” *Research Square (Research Square)*, Jun. 2023, doi: [https://doi.org/10.21203/rs.3.rs](https://doi.org/10.21203/rs.3.rs-3036958/v1)-3036958/v1.

[9] S. Schneider, Alexei Baevski, Ronan Collobert, and M. Auli, “wav2vec: Unsupervised Pre-training for Speech Recognition,” *arXiv (Cornell University)*, Apr. 2019, doi: <https://doi.org/10.48550/arxiv.1904.05862>.

[10] A. Radford et al., “Robust Speech Recognition via Large-Scale Weak Supervision,” *Arxiv Preprint 2307.14743v2*, 2023. [Online]. Available: <https://arxiv.org/abs/2307.14743> .

[11] OpenAI, “Whisper: Multilingual Speech Recognition and Translation,” 2023. [Online]. Available: <https://arxiv.org/abs/2212.04356> .

[12] N. Arabzadeh, A. Bigdeli, and C. L. A. Clarke, "Adapting Standard Retrieval Benchmarks to Evaluate Generated Answers," in *arXiv preprint arXiv:2401.04842*, Jan. 2024. Available: [https://arxiv.org/abs/2401](https://arxiv.org/abs/2401.04842).04842 .

[13]X. Li, B. Zou, Y. Fan, Y. Li, A. T. Aw, and Y. Hong, “Interview Evaluation: A Novel Approach for Automatic Evaluation of Conversational Question Answering Models,” pp. 3435–3446, Jan. 2023, doi: [https://doi.org/10.18653/v1/2023](https://doi.org/10.18653/v1/2023.emnlp-main.209).emnlp-main.209.

[14] L. Dietz, “A Workbench for Autograding Retrieve/Generate Systems,” *arXiv (Cornell University)*, May 2024, doi: [https://doi.org/10.48550/arxiv.2405](https://doi.org/10.48550/arxiv.2405.13177).13177.

[15] Z. He *et al.*, “Quality of Answers of Generative Large Language Models vs Peer Patients for Interpreting Lab Test Results for Lay Patients: Evaluation Study (Preprint),” *JMIR. Journal of medical internet research/Journal of medical internet research*, Jan. 2024, doi: [https://doi.org/10.2196](https://doi.org/10.2196/56655)/56655.

[16] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proc. NAACL-HLT 2019*, pp. 4171–4186, 2019. Available: <https://doi.org/10.48550/arXiv.1810.04805>

[17] H. Touvron *et al.*, “Llama 2: Open Foundation and Fine-Tuned Chat Models,” *arXiv.org*, Jul. 19, 2023. <https://doi.org/10.48550/arXiv.2307.09288>.

[18] P. Lewis *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” *arXiv.org*, Apr. 12, 2021. <https://arxiv.org/abs/2005.11401>.

[19] M. Milkowski, “Developing an open-source, rule-based proofreading tool,” *Software Practice and Experience*, vol. 40, no. 7, pp. 543–566, Jan. 2010, doi: <https://doi.org/10.1002/spe.v40:7>.

[20] D. C. Priya and D. R. VIJAYALAKSHMI, “Grammar Correction AI Tools for English Language Teachers in Higher Education,” *IJIRT*, 2024. https://www.academia.edu/124515534/Grammar\_Correction\_AI\_Tools\_for\_English\_Language\_Teachers\_in\_Higher\_Education?source=swp\_share (accessed Nov. 26, 2024)

[21] P. M. A. Cano, “Natural Language Processing of Grammar Checker Tools for Academic Writing: A Systematic Literature Review,” *Journal of Electrical Systems*, vol. 20, no. 7s, pp. 1151–1156, May 2024, doi: <https://doi.org/10.52783/jes.3611>.

[22] M.-H. Su, K.-Y. Huang, T.-H. Yang, K.-J. Lai, and C.-H. Wu, “Dialog state tracking and action selection using deep learning mechanism for interview coaching,” in *2016 International Conference on Asian Language Processing (IALP)*, 2016, pp. 7–12.

[23] Z.-T. Liu, M.-T. Han, B.-H. Wu, and A. Rehman, “Speech emotion recognition based on convolutional neural network with attention-based bidirectional long short-term memory network and multi-task learning,” *Applied Acoustics*, vol. 202, pp. 109178, Dec. 2022. [Online]. Available: <https://doi.org/10.1016/j.apacoust.2022.109178>.

[24] T. M. Wani, T. S. Gunawan, S. A. A. Qadri, M. Kartiwi, and E. Ambikairajah, “A comprehensive review of speech emotion recognition systems,” *IEEE Access*, vol. 9, pp. 47795–47813, 2021. [Online]. Available: <https://doi.org/10.1109/ACCESS.2021.3068045>.

[25] M. M. Rezapour Mashhadi and K. Osei-Bonsu, “Speech emotion recognition using machine learning techniques: Feature extraction and comparison of convolutional neural network and random forest,” *PLOS ONE*, vol. 18, no. 11, Nov. 2023. [Online]. Available: <https://doi.org/10.1371/journal.pone.0291500>.

[26] Y. Wang, A. Boumadane, and A. Heba, “A Fine-Tuned Wav2Vec 2.0/Hubert Benchmark for Speech Emotion Recognition, Speaker Verification and Spoken Language Understanding,” in *Arxiv Preprint 2111.02735v1*, 2021. [Online]. Available: <https://arxiv.org/abs/2111.02735>.

[27] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, and A. Sarroff, “CNN Architectures for Large-Scale Audio Classification,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017, pp. 131-135. [Online]. Available: [https://](https://doi.org/10.1109/ICASSP.2017.7952132)doi.org/10.1109/ICASSP.2017.7952132.

[28] P. Xu, “Fine-Tuning Whisper for ASR,” *Arxiv Preprint 2405.03484v1*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.03484>

[29] Wang, Y., Hernandez, A. G., Kyslyi, R., & Kersting, N. (2024). Evaluating quality of answers for Retrieval-Augmented Generation: A strong LLM is all you need. *arXiv (Cornell University)*. [https://doi.org/10.48550/arxiv.](https://doi.org/10.48550/arxiv.2406.18064)2406.18064

[30] O. Khattab and M. Zaharia, "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT," *Proc. 43rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 39–49, 2020. Available: <https://doi.org/10.1145/3511808.3557325>

[31] D. Huang et al., "From Large Language Models to Large Multimodal Models: A Literature Review," *Appl. Sci.*, vol. 14, no. 5068, pp. 1–30, 2024. Available: <http://dx.doi.org/10.3390/app14125068>.

[32] M. Soni and J. S. Thakur, “A Systematic Review of Automated Grammar Checking in English Language,” *arXiv.org*, Mar. 29, 2018. https://arxiv.org/abs/1804.00540 (accessed Oct. 26, 2023).

[41] Livingstone, S. R., & Russo, F. A. (2018). The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PLoS ONE*, 13(5), e0196391. <https://doi.org/10.1371/journal.pone.0196391>

[42] Cao, H., Cooper, D. G., Keutmann, M. K., Gur, R. C., Nenkova, A., & Verma, R. (2014). CREMA-D: Crowd-sourced Emotional Multimodal Actors Dataset. *IEEE Transactions on Affective Computing*. [https://doi.org/10.1109/TAFFC.2014.2336244](https://doi.org/10.1109/TAFFC.2014.2361228)

[39] Pichora-Fuller, M. K., & Dupuis, K. (2020). Toronto Emotional Speech Set (TESS). Borealis. <https://doi.org/10.5683/SP2/E8H2MF>

[40] Jackson, P. J. B., & Haq, S. U. (2011). Surrey Audio-Visual Expressed Emotion (SAVEE) Database. University of Surrey. Available: <http://kahlan.eps.surrey.ac.uk/savee/>

# Appendix

## Appendix A – Complete Code Listings

1. **SER Data Preparation and Augmentation**

A screenshot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

**A computer code with text

AI-generated content may be incorrect.**

Figure 34- SER Data Preparation and Augmentation

1. **SER Model Definition**

**A screenshot of a computer program

AI-generated content may be incorrect.**

Figure 35-SER Model Definition

1. **Rubric Generation & Answer-Evaluation Pipeline**

A computer screen shot of a code

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 36-Rubric Generation & Answer-Evaluation Pipeline

1. **Rubric Generation and dataset evaluation**

A white background with red and blue lines

AI-generated content may be incorrect.

A white background with red and blue lines

AI-generated content may be incorrect.

Figure -Rubric Generation & Dataset Evaluation Results

1. **Retrieval & Matching Setup**

A screenshot of a computer

AI-generated content may be incorrect.

Figure 38-Retrieval & Matching Setup

## Appendix B-Trials Learning Curves & Confusion Matrices

**Trial 1**

**A chart with blue squares

AI-generated content may be incorrect.**

Figure 39-SER Trial 1 - Confusion Matrix

**Trial 2**

**A chart with blue squares

AI-generated content may be incorrect.**

Figure 40-SER Trial 2 - Confusion Matrix

**Trial 3**

**A chart with blue squares

AI-generated content may be incorrect.**

Figure 41-SER Trial 3 - Confusion Matrix

**Trial 4**

**A blue and white grid with different colored squares

AI-generated content may be incorrect.**

Figure 42-SER Trial 4 - Confusion Matrix

**Trial 5**

**A chart with blue squares

AI-generated content may be incorrect.**

Figure 43-SER Trial 5 - Confusion Matrix

**Trial 6**

A graph of a graph of a graph

AI-generated content may be incorrect.

Figure 44-SER Trial 6 - Learning Curves

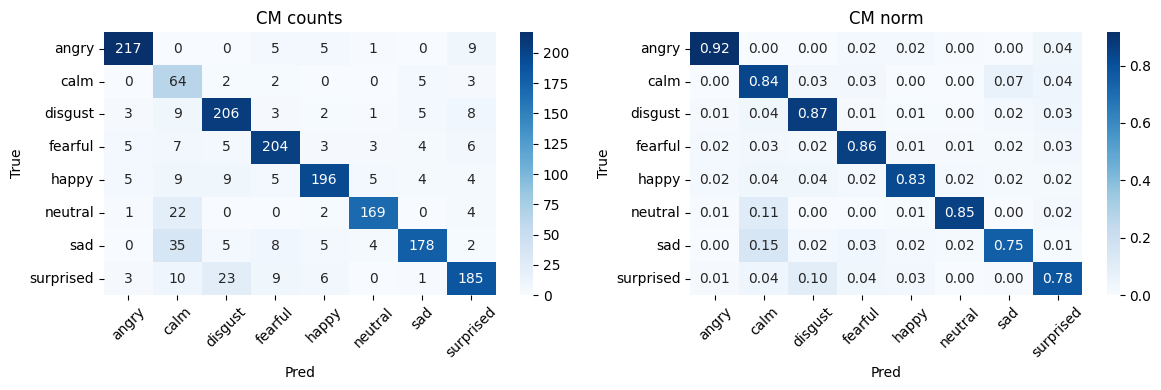


Figure 45-SER Trial 6 - Confusion Matrix

**Trial 7**

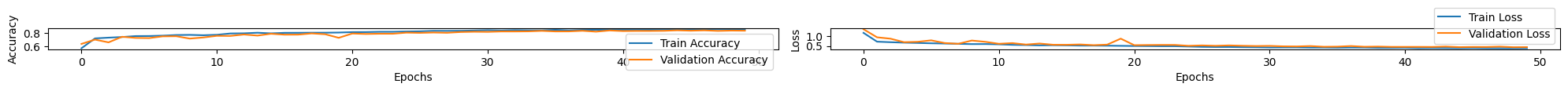
**A graph of a graph of a graph

AI-generated content may be incorrect.A graph with numbers and a chart

AI-generated content may be incorrect.**

Figure 46-SER Trial 7 - Confusion Matrix & Learning Curves

**Trial 8**

****

**A diagram of a confusion matrix

AI-generated content may be incorrect.**

Figure 47-SER Trial 8 - Confusion Matrix & Learning Curves

**Trial 9**

**A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.A graph with blue squares

AI-generated content may be incorrect.**

Figure 48-SER Trial 9 - Confusion Matrix & Learning Curves

**Trial 10**

**A graph of a graph and a graph of a graph

AI-generated content may be incorrect.**

Figure 49--SER Trial 10 - Learning Curves

**A screenshot of a graph

AI-generated content may be incorrect.**

Figure 50-SER Trial 10 - Confusion Matrix

**Trial 11**

**A graph of different colored lines

AI-generated content may be incorrect.**

Figure 51-SER Trial 11 -Learning Curves

**A screenshot of a graph

AI-generated content may be incorrect.**

Figure 52-SER Trial 11 - Confusion Matrix

**Trial 12**

**A graph with blue squares

AI-generated content may be incorrect.**

Figure 53-SER Trial 12 - Confusion Matrix

**Trial 13**

A graph of loss and loss

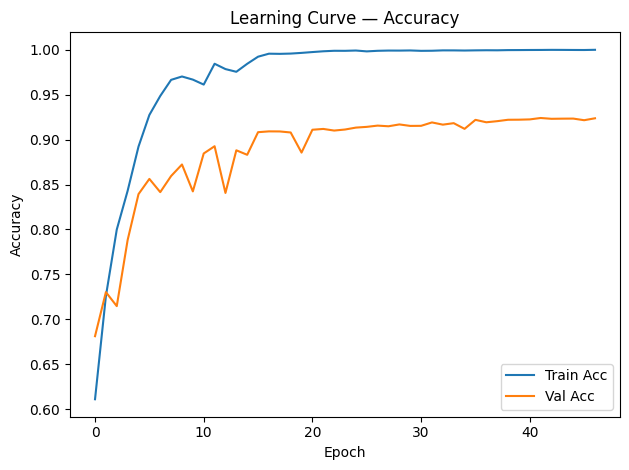
AI-generated content may be incorrect.A graph of a graph

AI-generated content may be incorrect.A screenshot of a graph

AI-generated content may be incorrect.

Figure 54-SER Trial 13 - Confusion Matrix & Learning Curves

**Trial 14**

A graph of loss and loss

AI-generated content may be incorrect.A graph with blue squares

AI-generated content may be incorrect.

Figure 55-SER Trial 14 - Confusion Matrix & Learning Curves

**Trial 15**

A graph of a graph

AI-generated content may be incorrect.A graph of loss and loss

AI-generated content may be incorrect.A screenshot of a graph

AI-generated content may be incorrect.

Figure 56-SER Trial 15 - Confusion Matrix & Learning Curves

**Trial 16**

A graph of a curve

AI-generated content may be incorrect.A graph of loss and loss

AI-generated content may be incorrect.A screenshot of a graph

AI-generated content may be incorrect.

Figure 57-SER Trial 16 - Confusion Matrix & Learning Curves

## Appendix C- Project directory layout

Illustrating separation of UI (app.py), analysis components (components/), and shared utilities (utils/)

A screen shot of a computer program

AI-generated content may be incorrect.

## Appendix D – Glossary & Abbreviations

Table 15-Glossary & Abbreviations

|  |  |
| --- | --- |
| Acronym | Definition |
| ASR | Automatic Speech Recognition, the process of converting spoken audio into text. |
| CNN | Convolutional Neural Network, a deep learning architecture especially effective at extracting spatial or temporal features. |
| Conv1D | One-dimensional convolutional layer, used to process sequential data such as audio feature vectors. |
| DFT | Discrete Fourier Transform, a mathematical transform that converts a time-domain signal to its frequency components. |
| FAISS | Facebook AI Similarity Search, a library for efficient similarity search over dense vector embeddings. |
| F₁ | F-score, the harmonic mean of precision and recall in classification tasks. |
| FFT | Fast Fourier Transform, an algorithm to compute the DFT efficiently. |
| GPU | Graphics Processing Unit, specialized hardware that accelerates parallel computations in deep learning. |
| HPC | High-Performance Computing, systems or clusters that provide large-scale parallel compute capability. |
| LSTM | Long Short-Term Memory, a type of recurrent neural network that maintains long-range dependencies via gated memory cells. |
| LLM | Large Language Model, a deep learning model (e.g. GPT-4o, BERT) trained on massive text corpora for language tasks. |
| MIPS | Maximum Inner Product Search, a retrieval operation commonly used in vector similarity search. |
| MFCC | Mel-Frequency Cepstral Coefficients, features summarizing the short-term power spectrum on a perceptual scale. |
| ML | Machine Learning, techniques by which systems learn patterns from data without explicit programming. |
| NLP | Natural Language Processing, the subfield of AI concerned with the interaction between computers and human language. |
| PCA | Principal Component Analysis, a dimensionality-reduction technique that projects data into orthogonal principal axes. |
| RAG | Retrieval-Augmented Generation, a framework that augments LLM output with retrieved external context. |
| RMS | Root-Mean-Square energy, a measure of the average power (loudness) of an audio signal over time. |
| RNN | Recurrent Neural Network, a neural network architecture designed to handle sequential data. |
| SER | Speech Emotion Recognition, the task of automatically identifying a speaker’s emotional state from their voice. |
| SNR | Signal-to-Noise Ratio, the ratio of desired signal power to background noise power in an audio recording. |
| STFT | Short-Time Fourier Transform, a DFT applied over sliding windows to analyze time-varying frequency content. |
| t-SNE | t-Distributed Stochastic Neighbor Embedding, a nonlinear dimensionality-reduction technique for visualization. |
| VAD | Voice Activity Detection, the process of detecting the presence or absence of human speech in audio. |
| Whisper | An open-source, transformer-based speech-to-text model by OpenAI. |