DESIGNING A SPEECH EMOTION RECOGNITION SYSTEM FOR CUSTOMER SERVICE PLATFORMS, IMPROVING RESPONSE QUALITY BY ANALYZING EMOTIONAL TONE AND TAILORING INTERACTIONS

A dissertation submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

In

Computer Science and Engineering (Data Science)

By

MASULA ASRA FATHIMA (23U61A6711)

Under the guidance of

Mrs. Noore Ilahi Assistant Professor

B. Tech., M. Tech.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

(Approved by AICTE, New Delhi & Affiliated to JNTUH)

(Recognized under section 2(f) of UGC Act 1956) An

ISO:9001-2015 Certified Institution

CHILKUR (V), MOINABAD (M), R.R. DIST. T.S-501504

June 2025

(Approved by AICTE & Affiliated to JNTUH)
(Recognized under Section 2(f) of UGC Act 1956) An
ISO:9001-2015 Certified Institution
Survey No. 179, Chilkur (V), Moinabad (M), Ranga Reddy Dist. TS.
JNTUH Code (U6) ECE –EEE-CSD-CSM – CSE - CIVIL – ME – MBA - M.Tech EAMCET Code - (GLOB)

Department of Computer Science and Engineering

Noore Ilahi
B. Tech., M. Tech. **Assistant Professor & Head**

Date: 02-06-2025

CERTIFICATE

This is to certify that the project work entitled "**DESIGNING A SPEECH EMOTION RECOGNITION SYSTEM FOR CUSTOMER SERVICE PLATFORMS, IMPROVING RESPONSE QUALITY BY ANALYZING EMOTIONAL RESPONSE QUALITY BY ANALYZING EMOTIONAL TONE AND TAILORING INTERACTIONS**", is a bonafide work of **MASULA ASRA FATHIMA (HT.No:23U61A6711),** submitted in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science and Engineering (Data Science)** during the academic year 2024-25. This is further certified that the work done under my guidance, and the results of this work have not been submitted elsewhere for the award of any other degree or diploma.

Internal Guide
Mrs. Noore Ilahi
Assistant Professor

Head of the Department
Mrs. Noore Ilahi
Assistant Professor

DECLARATION

I hereby declare that the project work entitled **DESIGNING A SPEECH EMOTION RECOGNITION SYSTEM FOR CUSTOMER SERVICE PLATFORMS, IMPROVING RESPONSE QUALITY BY ANALYZING EMOTIONAL RESPONSE QUALITY BY ANALYZING EMOTIONAL TONE AND TAILORING INTERACTIONS,** submitted to **Department of Computer Science and Engineering (Data Science), Global Institute of Engineering & Technology, Moinabad,** affiliated to **JNTUH, Hyderabad** in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering (Data Science)** is the work done by me and has not been submitted elsewhere for the award of any degree or diploma.

MASULA ASRA FATHIMA (23U61A6711)

ACKNOWLEDGEMENT

I am thankful to my guide **Mrs. Noore Ilahi,** Assistant Professor of CSE(DS) Department for her valuable guidance for successful completion of this project.

I express my sincere thanks to **Mrs. G.Pavani,** Project Coordinator for giving me an opportunity to undertake the project "**DESIGNING A SPEECH EMOTION RECOGNITION SYSTEM FOR CUSTOMER SERVICE PLATFORMS, IMPROVING RESPONSE QUALITY BY ANALYZING EMOTIONAL RESPONSE QUALITY BY ANALYZING EMOTIONAL TONE AND TAILORING INTERACTIONS**" and for enlightening me on various aspects of my project work and assistance in the evaluation of material and facts. She not only encouraged me to take up this topic but also given her valuable guidance in assessing facts and arriving at conclusions.

I am also most obliged and grateful to **Mrs. Noore Ilahi,** Assistant Professor and Head, Department of CSE(DS) for giving me guidance in completing this project successfully.

I express my heart-felt gratitude to our Vice-Principal **Prof. Dr. G Ahmed Zeeshan**, Coordinator Internal Quality Assurance Cell (IQAC) for his constant guidance, cooperation, motivation and support which have always kept me going ahead. I owe a lot of gratitude to him for always being there for me.

I also most obliged and grateful to our Principal **Dr. P. Raja Rao** for giving me guidance in completing this project successfully.

I also thank my parents for their constant encourage and support without which the project would have not come to an end.

Last but not the least, I would also like to thank all my class mates who have extended their cooperation during our project work.

MASULA ASRA FATHIMA (23U61A6711)

VISION

The Vision of the Department is to produce professional Computer Science Engineers who can meet the expectations of the globe and contribute to the advancement of engineering and technology which involves creativity and innovations by providing an excellent learning environment with the best quality facilities.

MISSION

M1. To provide the students with a practical and qualitative education in a modern technical environment that will help to improve their abilities and skills in solving programming problems effectively with different ideas and knowledge.

M2. To infuse the scientific temper in the students towards the research and development in Computer Science and Engineering trends.

M3. To mould the graduates to assume leadership roles by possessing good communication skills, an appreciation for their social and ethical responsibility in a global setting, and the ability to work effectively as team members.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in MNC's and / or to pursue postgraduate studies with an appreciation for lifelong learning.

PEO2: To provide graduates with analytical and problem solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multicultural team.

PEO3: To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting edge technologies of multi disciplinary nature for societal development.

PROGRAMME OUTCOMES:

PO1: Engineering knowledge: An ability toApply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: An ability to Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.

PO3: Design/development of solutions: An ability to Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.

PO4: Conduct investigations of complex problems: An ability to Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: An ability to Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: An ability to Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment sustainability: An ability to Understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and the need for sustainable development.

PO8: Ethics: An ability to Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and teamwork: An ability to Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: An ability to Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: An ability to Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Lifelong learning: An ability to Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change.

PROGRAMME SPECIFIC OUTCOMES

PSO1: An Ability to Apply the fundamentals of mathematics, Computer Science and Engineering Knowledge to analyze and develop computer programs in the areas related to Algorithms, System Software, Web Designing, Networking and Data mining for efficient Design of computer-based system to deal with Real time Problems.

PSO2: An Ability to implement the Professional Engineering solutions for the betterment of Society, and able to communicate with professional Ethics effectively

ABSTRACT

This project details the development of a real-time Speech Emotion Recognition (SER) system designed to enhance customer service interactions by analyzing and visualizing emotional states from spoken audio. The system features a robust Python-based FastAPI backend, meticulously engineered to perform efficient audio feature extraction, specifically utilizing Librosa to derive Mel-frequency Cepstral Coefficients (MFCCs), pitch, and energy. These features then feed into a pre-trained scikit-learn machine learning model, which accurately classifies emotions into discrete categories such as happy, sad, angry, and neutral. All analysis results, including detected emotion, confidence scores, and temporal metadata, are persistently logged to Google Firestore, enabling comprehensive historical tracking.

Complementing the backend is an intuitive HTML/JavaScript frontend that functions as an interactive widget. This user-friendly interface allows for seamless audio input via live microphone streaming or file uploads, displaying real-time emotion predictions and visually representing the detected sentiment. Furthermore, the frontend provides the capability to retrieve and display historical emotion logs from Firestore, offering valuable data-driven insights for supervisors and agents alike. The integration of highperformance audio processing, a scalable API, and a reliable cloud database underscores the system's capability to provide immediate feedback, inform agent training, and foster a more empathetic and efficient customer relationship management ecosystem. This solution establishes a strong framework for integrating advanced AI capabilities into practical business applications, thereby improving both operational efficiency and customer satisfaction.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

CHAPTER 1

Introduction and Project Overview

This chapter provides a high-level understanding of the project, its purpose, and its overall goals.

1.1 Existing System

In many traditional customer service and call center environments, the assessment of customer emotional states during interactions heavily relies on the subjective interpretation of the Customer Service Representative (CSR). Agents are trained to listen for verbal cues, tone of voice, and speech patterns to gauge customer sentiment. Post-call analysis, if conducted, often involves manual review of call recordings, relying on human listeners to identify instances of customer dissatisfaction, frustration, or satisfaction. Basic sentiment analysis might exist, primarily focusing on text transcripts (if available) and keyword spotting. Quality assurance (QA) processes for agent performance are often based on checklists and supervisors' subjective evaluations of agent adherence to scripts and general call handling, with limited objective data on customer emotional journeys throughout the call.

1.2 Disadvantages of Existing System

The existing traditional approach suffers from several critical disadvantages:

- Subjectivity and Inconsistency: Human perception of emotion is highly subjective and varies significantly from one CSR or QA reviewer to another. This leads to inconsistent emotional assessment and biased insights.
- Scalability Issues: Manually analyzing every call for emotional content is impractical and impossible at scale, especially for high-volume call centers. This limits the ability to identify pervasive issues or trends.
- Delayed Feedback: Post-call manual analysis means insights are often delayed, preventing real-time intervention or immediate adjustment of agent strategy during a live conversation.
- Lack of Granular Data: Traditional methods provide limited quantifiable data on emotion over time within a single call, making it difficult to pinpoint exact moments of emotional shifts or triggers.

1.3 Proposed System

The Proposed System is an automated, real-time Speech Emotion Recognition (SER) solution designed to objectively analyze and present customer emotional states during live and recorded interactions. This system consists of:

- A robust Python FastAPI Backend: This acts as the core processing engine, handling audio input, performing sophisticated feature extraction (such as Mel-frequency Cepstral Coefficients - MFCCs, pitch, and energy) using the librosa library. It then feeds these features into a pre-trained machine learning model (e.g., scikit-learn based RandomForestClassifier) for emotion classification (e.g., happy, sad, angry, neutral, surprised, fearful).
- Google Firestore Integration: All emotion analysis results, including detected emotion, confidence scores, timestamps, agent IDs, and call IDs, are securely logged to a NoSQL cloud database (Firestore) for persistent storage and future analytical purposes.
- An Intuitive HTML/JavaScript Frontend Widget: This user-friendly interface is designed to seamlessly integrate into a CSR's workspace. It allows for live audio input via microphone, file uploads, and provides immediate visual feedback on the detected emotion. It also includes functionality to fetch and display recent emotion logs from Firestore, offering historical context.

The system aims to move beyond subjective human interpretation by employing computational linguistics and machine learning to provide objective, data-driven emotional insights.

1.4 Advantages of Proposed System

The Proposed System offers significant advantages over the existing manual and subjective methods:

- Objectivity and Consistency: Provides an impartial, data-driven assessment of customer emotions, eliminating human bias and ensuring consistent analysis across all interactions.
- Real-time Insights and Proactive Intervention: Offers instant emotional feedback to CSRs during live calls, enabling them to adapt their approach, de-escalate situations, or emphasize empathy in real-time.

- Scalability and Efficiency: Automates emotion analysis, allowing for the processing of a vast number of calls without the need for extensive manual review, significantly reducing operational overhead.
- Granular Data Collection: Captures precise emotional data points throughout a call, enabling detailed post-call analytics to pinpoint exact emotional triggers and shifts.
- Enhanced Agent Performance: Equips CSRs with a powerful tool to better understand customer needs, leading to improved communication, higher first-call resolution rates, and increased customer satisfaction.

Problem Description

In many customer service environments, interactions often lack real-time awareness of the customer's emotional state. CSRs frequently rely on verbal cues and their subjective interpretation, which can be inconsistent, especially during high-pressure or emotionally charged calls. This often leads to generic or inappropriate responses that fail to de-escalate tension, build rapport, or address the underlying emotional needs of the customer. The focus remains heavily on *what* is said (keywords, intents) rather than *how* it is said (emotional tone and intensity).

Proposed Solution Overview

The proposed Speech Emotion Recognition system will act as an intelligent assistant for CSRs. By analyzing the acoustic properties of customer speech, it will classify emotions and display them in real-time. This objective data empowers CSRs to proactively adapt their approach, providing more empathetic, effective, and personalized service.

JUSTIFICATION

The implementation of an SER system is justified by its potential to:

- Significantly enhance customer satisfaction (CSAT) scores.
- Reduce average handle time (AHT) for complex emotional interactions.
- Decrease call escalation rates.
- Provide valuable data for targeted CSR training and coaching.
- Identify systemic issues leading to customer frustration.

- Improve overall operational efficiency and reduce costs associated with poor customer service.

CHAPTER 2
LITERATURE SURVEY

CHAPTER 2

2.1 Speech Emotion Recognition

Speech Emotion Recognition (SER) is a subfield of affective computing that aims to automatically identify and categorize the emotional state of a speaker from their voice. Emotions play a crucial role in human communication, conveying nuances and intentions beyond the literal meaning of words. Automating the detection of these emotional cues can significantly enhance human-computer interaction, improve customer service, facilitate mental health assessments, and enable more empathetic AI systems.

The process of SER generally involves several key stages:

1. Speech Signal Pre-processing: Preparing the raw audio for analysis (e.g., noise reduction, silence removal, normalization).
2. Feature Extraction: Deriving quantitative acoustic features from the speech signal that are indicative of emotional content.
3. Emotion Classification: Using machine learning or deep learning models to map the extracted features to predefined emotional categories.

2.2 Historical Context and Traditional Approaches

Early research in SER primarily relied on hand-crafted acoustic features and traditional machine learning algorithms.

- Prosodic Features: These relate to the suprasegmental aspects of speech, reflecting changes in pitch, energy, and rhythm over time.
    - Pitch (F0): The fundamental frequency of vocal fold vibration. Variations in pitch (mean, min, max, range, contour) are strong indicators of emotion (e.g., high pitch for anger/excitement, low pitch for sadness).
    - Energy/Intensity: The loudness or amplitude of the speech signal. High energy often correlates with strong emotions like anger or joy, while low energy might indicate sadness or boredom.
    - Duration/Speech Rate: The length of speech segments, pauses, and the speed of articulation. Faster speech rates can indicate excitement or nervousness, while slower rates might suggest sadness or contemplation.

- Spectral Features: These describe the frequency content and timbre of the speech signal.
    - Mel-Frequency Cepstral Coefficients (MFCCs): Widely used in speech processing, MFCCs represent the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a non-linear Mel scale of frequency. They are robust for capturing vocal tract characteristics relevant to emotion. The project uses MFCCs as a primary feature. ○ Linear Predictive Coding Coefficients (LPCCs): Represent the vocal tract filter.
- Voice Quality Features: Relate to the characteristics of the vocal cords.
    - Jitter and Shimmer: Measures of pitch and amplitude perturbation, respectively, which can indicate voice roughness or instability associated with certain emotional states. ○ Harmonic-to-Noise Ratio (HNR): Indicates the periodicity of the vocal signal.

Traditional Machine Learning Classifiers

Before the advent of deep learning, several conventional machine learning algorithms were dominant for emotion classification:

- Support Vector Machines (SVMs): Widely popular due to their effectiveness in high-dimensional spaces and their ability to find an optimal hyperplane for classification. SVMs have shown strong performance with various acoustic feature sets.
- Gaussian Mixture Models (GMMs): Probabilistic models that represent the probability distribution of observations. GMMs are particularly effective for modeling speaker-dependent emotional characteristics.
- Hidden Markov Models (HMMs): Used for modeling sequential data, HMMs capture the temporal dynamics of emotional speech, where the emotional state can transition over time.
- K-Nearest Neighbors (KNN): A non-parametric, instance-based learning algorithm that classifies new data points based on the majority class of their 'k' nearest neighbors in the feature space.
- Decision Trees and Random Forests: Decision trees recursively split the data based on features, forming a tree-like structure. Random Forests, an ensemble method, combine multiple decision trees to improve accuracy

and robustness by reducing overfitting. This project utilizes a RandomForestClassifier for its robustness and interpretability.

- Artificial Neural Networks (ANNs) / Multilayer Perceptrons (MLPs): Early neural network architectures, typically shallow, were used to learn non-linear mappings from features to emotions.

2.3 Advancements with Deep Learning

The past decade has seen a paradigm shift in SER research with the rise of deep learning, primarily due to their ability to automatically learn hierarchical features directly from raw or minimally processed audio data, bypassing the need for extensive hand-crafted feature engineering.

- Convolutional Neural Networks (CNNs): Excel at capturing local patterns and spatial hierarchies, often applied to spectrograms (2D representations of audio features over time and frequency). They effectively extract robust features from these time-frequency representations.
- Recurrent Neural Networks (RNNs) and their variants (LSTMs, GRUs): Designed to process sequential data, RNNs (especially Long Short-Term Memory - LSTMs, and Gated Recurrent Units - GRUs) are highly effective in modeling the temporal dependencies and long-range contextual information inherent in speech signals. They can capture how emotions evolve over an utterance.
- Hybrid Models (CNN-RNN/LSTM): Many state-of-the-art SER systems combine CNNs (for local feature extraction from spectrograms) with RNNs/LSTMs (for modeling temporal dynamics) to leverage the strengths of both architectures.
- Attention Mechanisms: Integrated into deep learning models, attention mechanisms allow the model to focus on the most relevant parts of the speech signal (e.g., specific emotional cues) over time, improving performance.
- Transformer Networks: Originally developed for natural language processing, Transformers, particularly architectures like Wav2Vec 2.0 or HuBERT, have shown remarkable success in learning powerful speech representations from large amounts of unlabeled audio data. These pretrained models can then be fine-tuned for SER tasks, achieving state-ofthe-art results by capturing complex long-range dependencies.

- End-to-End Learning: Deep learning facilitates end-to-end SER systems that can directly learn from raw audio waveforms to predict emotions, eliminating the need for explicit feature extraction steps. This simplifies the pipeline and can potentially capture more intricate patterns.

2.4 Common Datasets in SER Research

The availability of publicly accessible emotional speech datasets is critical for training and evaluating SER models. Challenges include the variability of emotional expression across speakers, languages, and cultures, and the difficulty in obtaining large volumes of truly spontaneous emotional speech.

Some widely used datasets include:

- IEMOCAP (Interactive Emotional Dyadic Motion Capture): A multimodal dataset (audio, video, motion capture) of dyadic (two-person) improvised and acted interactions. It's highly popular for its spontaneous emotional content.
- RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song): Features professional actors expressing 8 different emotions in both speech and song. It offers clean audio and controlled emotional expressions.
- EMO-DB (Berlin Emotional Speech Database): Contains acted emotional speech from professional German actors, covering 7 emotions.
- SAVEE (Surrey Audio-Visual Expressed Emotion): Features recordings from 4 male actors expressing 7 emotions.
- CREMA-D (Crowd-sourced Emotional Multimodal Actors Dataset): Contains over 7,000 clips from 91 actors expressing 6 emotions at different intensity levels.

2.5 Challenges in Speech Emotion Recognition

Despite significant progress, SER faces several ongoing challenges:

- Emotional Variability: Emotions are highly subjective, vary across individuals, cultures, and contexts, and are often blended or subtle, making accurate recognition difficult.

- Data Scarcity: Obtaining large, diverse, and well-annotated spontaneous emotional speech datasets is challenging due to privacy concerns, recording costs, and the difficulty of eliciting genuine emotions.
- Real-time Constraints: For applications like call centers, SER systems need to operate with very low latency, which can be challenging for computationally intensive models.
- Noisy Environments: Real-world audio often contains background noise, reverberation, and other distortions that degrade the quality of acoustic features and the accuracy of models.
- Cross-Corpus Generalization: Models trained on one dataset often perform poorly when tested on another due to differences in recording conditions, speakers, and emotional portrayals.

2.6 Applications of SER

SER has a wide range of practical applications:

- Customer Service and Call Centers: Real-time emotion detection can alert agents to frustrated or angry customers, allowing for immediate intervention and improved service. Post-call analysis can identify training needs for agents.
- Human-Computer Interaction (HCI): Making virtual assistants, chatbots, and robots more empathetic and responsive to user emotions, leading to more natural and intuitive interactions.
- Mental Health Monitoring: Detecting indicators of stress, depression, or anxiety in speech patterns, potentially aiding in early diagnosis or ongoing monitoring.
- Education: Assessing student engagement or frustration levels during elearning or tutoring sessions.
- Automotive Industry: Monitoring driver fatigue or stress levels to enhance safety systems.
- Gaming and Entertainment: Creating more immersive and responsive gaming experiences or dynamic content generation.

2.7 Conclusion of Literature Survey

The field of Speech Emotion Recognition has evolved significantly, moving from traditional hand-crafted features and machine learning models to sophisticated

deep learning architectures. While challenges remain, particularly concerning data scarcity, generalization across diverse real-world conditions, and ethical considerations, continuous research is pushing the boundaries of accuracy and applicability.This project, by leveraging established techniques like MFCC feature extraction with librosa and a robust RandomForestClassifier from sklearn, builds upon a solid foundation of traditional machine learning while remaining adaptable to future integrations with more advanced deep learning models or hybrid approaches if increased complexity and larger datasets become available.

CHAPTER 3 SYSTEM ANALYSIS

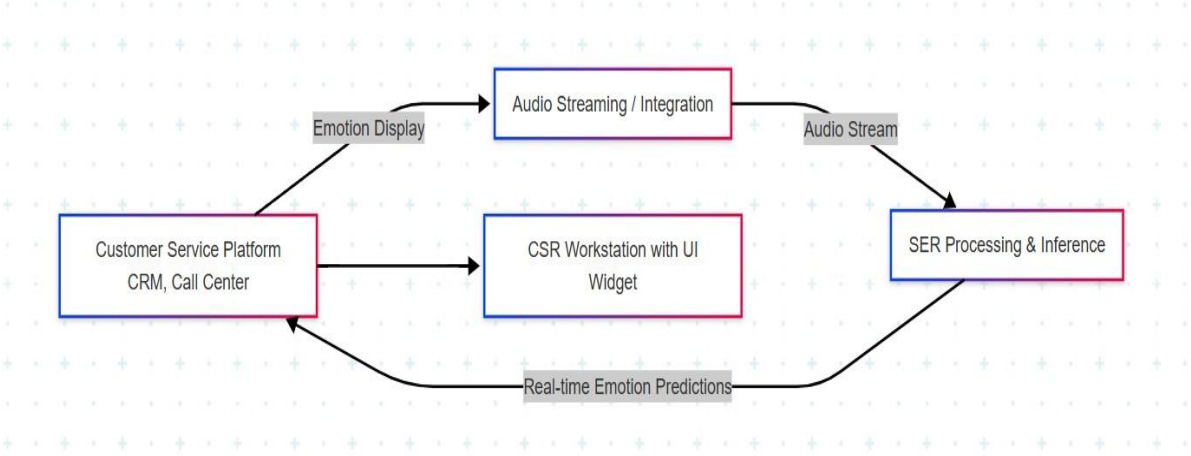
CHAPTER 3

Solution Design and Architecture

This chapter details the high-level and detailed design of the SER system, including its technical architecture and components.

3.1 High-Level Architecture

Diagram 3.1: High-Level System Architecture



Description: The customer's voice enters the Customer Service Platform (e.g., Call Center system). The audio stream is then routed to an Audio Streaming/Integration module. This module feeds the real-time audio to the SER Processing & Inference engine. This engine analyzes the audio and generates emotion predictions, which are then sent back to the Customer Service Platform. Finally, the CSR Workstation displays these emotions to the agent via an integrated UI widget. **3.2 Component Breakdown**

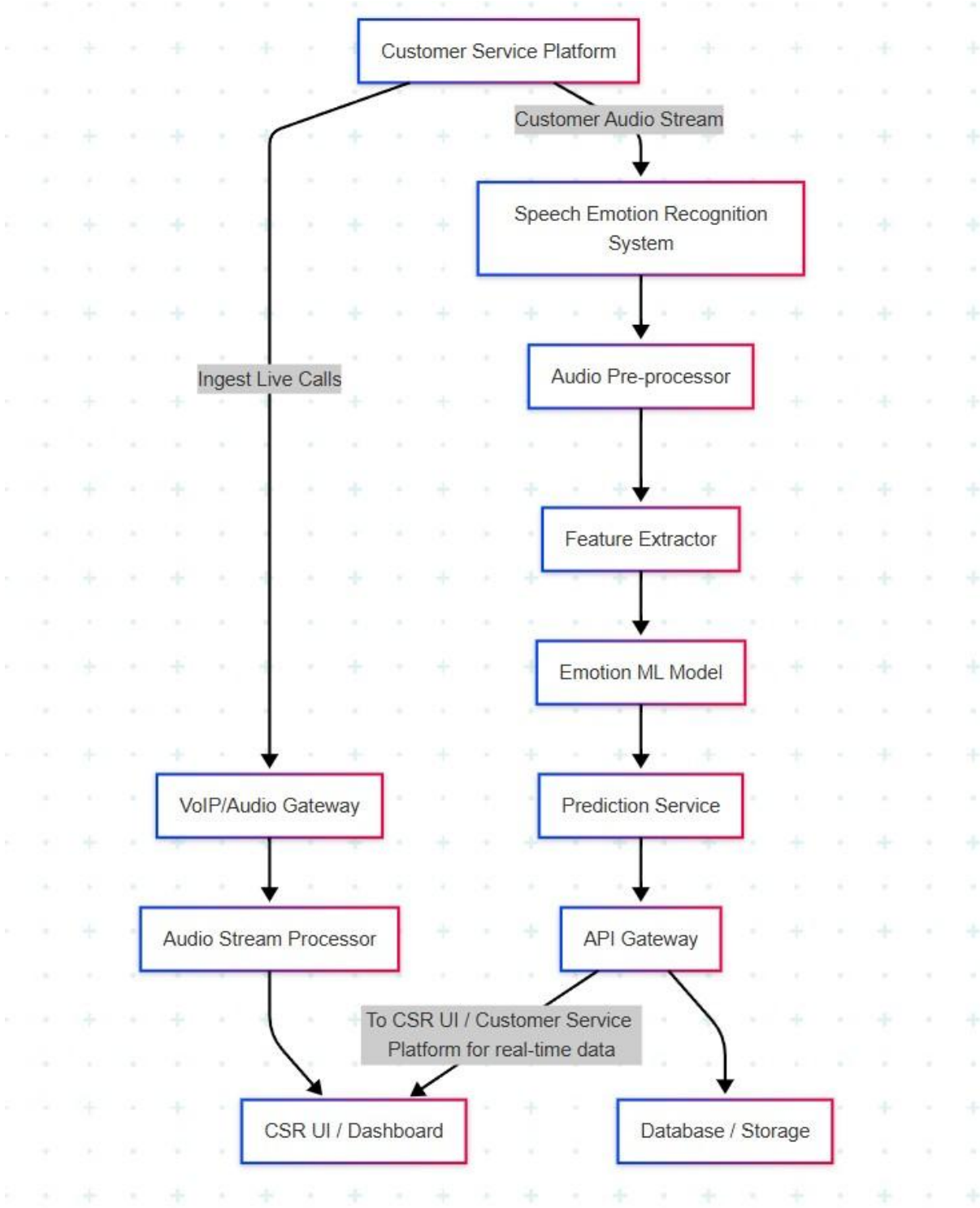
Description:

VoIP/Audio Gateway: Captures live audio streams from customer calls.

Audio Stream Processor: Responsible for routing the audio and potentially separating customer and agent speech (speaker diarization, a future enhancement).

Audio Pre-processor: Cleans the raw audio data (e.g., noise reduction, normalization, silence removal).

Diagram 3.2: Detailed Component Diagram



Feature Extractor: Extracts relevant acoustic features from the pre-processed audio (e.g., MFCCs, pitch, energy, formants). These features are crucial for the ML model.

Emotion ML Model: The core of the system. This is a pre-trained deep learning model (e.g., CNN for spatial features, RNN for temporal sequences, or a

combination like a CNN-LSTM) that takes the extracted features and predicts the emotional state.

Prediction Service: Takes the raw model output, processes it (e.g., applies thresholds, determines the primary emotion, calculates confidence scores), and formats it for consumption.

API Gateway: Provides a well-defined RESTful API for other systems (like the CSR UI or the main Customer Service Platform) to request and receive emotion predictions.

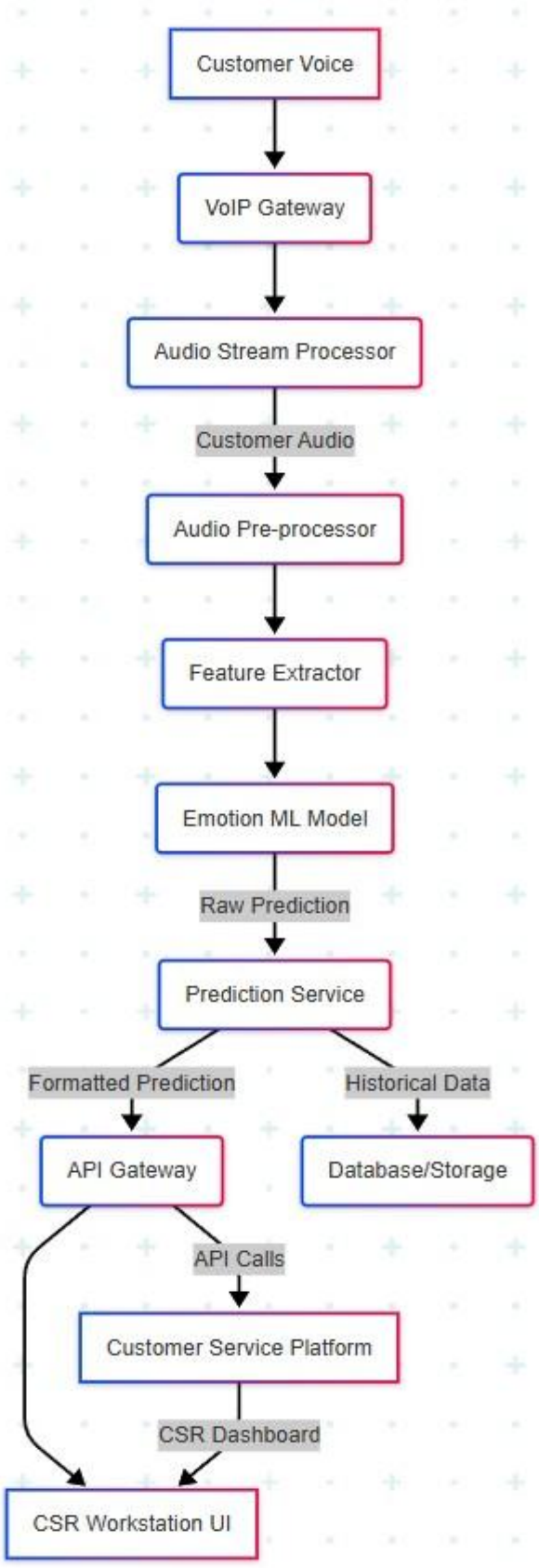
Database/Storage: Stores historical emotion data, call metadata, model versions, and audit logs for analytics and future model retraining.

CSR UI / Dashboard: The frontend component that displays the real-time emotion insights to the CSR.

3.3 Data Flow Description:

1. **Customer Voice:** The source of the audio data.
2. **VoIP Gateway:** Captures and digitizes the live voice stream.
3. **Audio Stream Processor:** Handles the audio stream, potentially separating speakers.
4. **Audio Pre-processor:** Cleans and normalizes the raw audio.
5. **Feature Extractor:** Derives acoustic features from the cleaned audio.
6. **Emotion ML Model:** Predicts emotion based on extracted features.
7. **Prediction Service:** Processes model output into actionable insights (e.g., primary emotion, confidence).
8. **API Gateway:** Exposes the emotion predictions via an API.
9. **CSR Workstation UI:** Displays real-time emotion to the CSR.
10. **Database/Storage:** Stores historical emotion data for analytics.
11. **Customer Service Platform:** Integrates with the SER system to embed the UI and potentially trigger actions.

Diagram 3.3: Data Flow Diagram (DFD)



3.4 Technologies Used (Proposed)

- **Programming Language:** Python (for ML, backend API)
- **Machine Learning Frameworks:** TensorFlow/Keras or PyTorch
- **Audio Processing Libraries:** Librosa, torchaudio, OpenSMILE (for feature extraction)
- **Web Framework (API):** FastAPI or Flask
- **Database:** PostgreSQL (for structured historical data) or MongoDB (for flexible logging)
- **Deployment:** Docker (containerization), Kubernetes (orchestration), AWS/GCP/Azure (cloud platform)
- **Real-time Streaming:** Kafka or Kinesis (for large-scale audio stream ingestion, if needed)
- **UI (CSR Widget):** JavaScript framework (React/Angular/Vue.js) for integration with existing platforms, or simple HTML/CSS/JS if it's a standalone component.

3.5 Emotion Categories

The system will initially focus on classifying the following core emotions:

- Neutral
- Happy
- Sad
- Angry
- Surprised
- Fearful
- Disgusted

Future extensions may include derived categories like "Frustrated," "Calm," or "Excited," or the concept of arousal-valence dimensions.

CHAPTER 4

SYSTEM DESIGN

CHAPTER 4

Detailed Design and Implementation

This chapter delves into the specific design choices and technical implementation aspects.

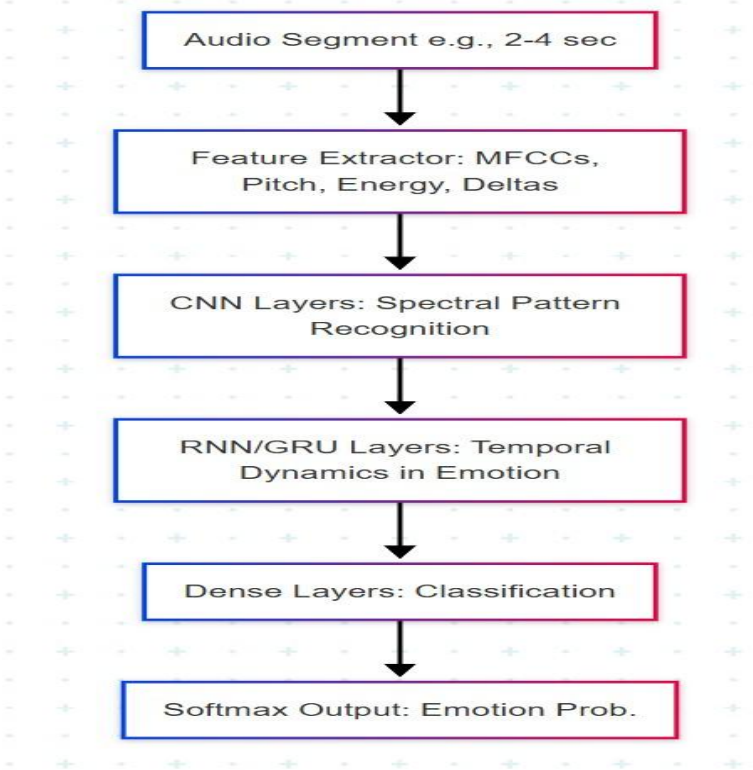
4.1 Machine Learning Model Design

- **Model Type:** A hybrid deep learning architecture, such as a **Convolutional Neural Network (CNN)** for extracting local spectral features, followed by a **Recurrent Neural Network (RNN) or Gated Recurrent Unit (GRU)**

layer for capturing temporal dependencies in the speech signal. A Transformer-based model could also be considered for advanced sequence modeling if computational resources permit.

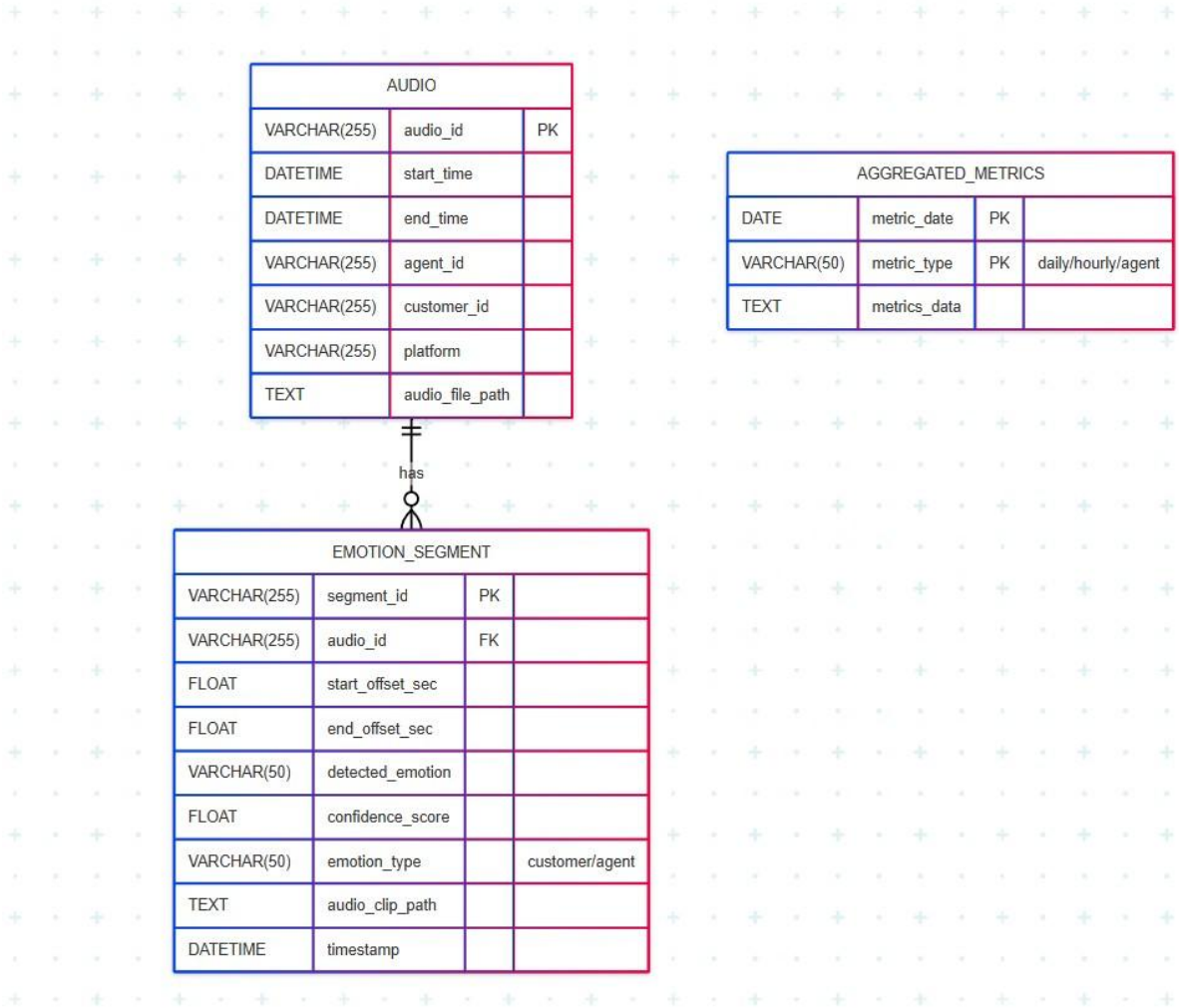
- **Input:** Mel-frequency cepstral coefficients (MFCCs), Pitch, Energy, and possibly their delta and delta-delta features, extracted from short audio frames (e.g., 25ms windows with 10ms hop).
- **Output Layer:** Softmax activation for multi-class emotion classification, providing probability scores for each emotion.
- **Loss Function:** Categorical Cross-entropy.
- **Optimization:** Adam optimizer.

Diagram 4.1: ML Model Architecture



4.2 Database Schema (Simplified)

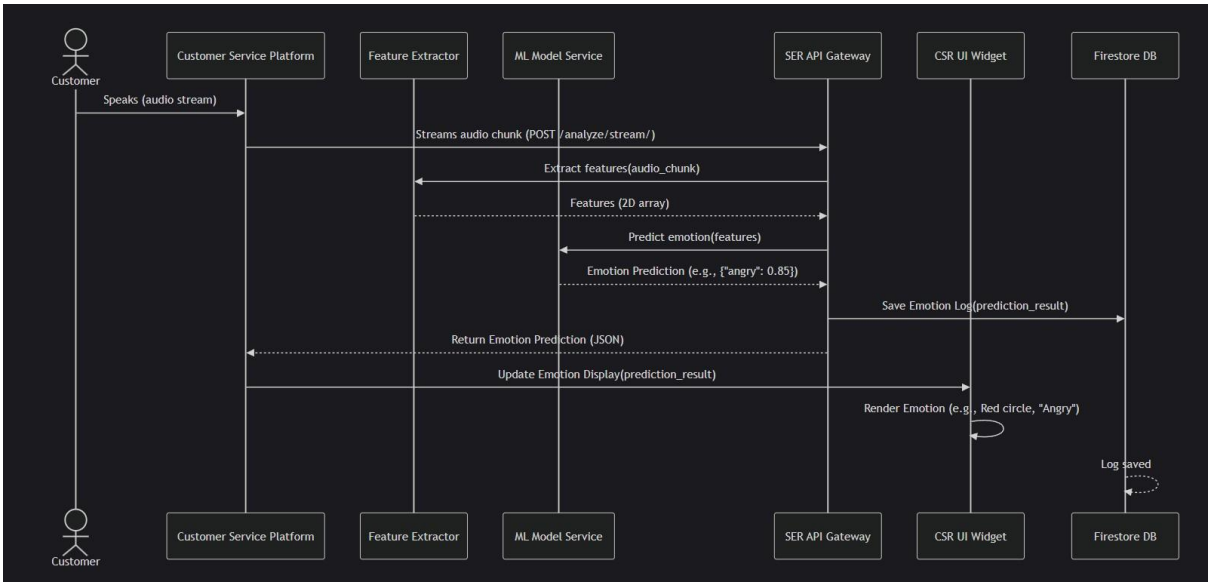
Diagram 4.2: Entity-Relationship Diagram (ERD)



Description:

- **EMOTION_SEGMENT:** Stores the emotion predictions for specific segments of a call. segment_id is PK, call_id is FK. Includes detected_emotion, confidence_score, start_offset_sec, end_offset_sec, and emotion_type (e.g., 'customer', 'agent').
- **AGGREGATED_METRICS:** Stores pre-calculated daily, hourly, or agent-specific aggregated emotional metrics for faster dashboard loading. metrics_data can be a JSONB field for flexible storage of various statistics.

Diagram 4.3: Sequence Diagram



Description:

Analyze Audio & Display Emotion

This sequence diagram depicts the flow of operations and messages between key actors and system components when an audio chunk is processed for emotion analysis and then displayed on the CSR's interface.

1. **Customer Speaks:** The process is initiated by the **Customer** providing audio input during a call.
2. **CSP Streams Audio:** The **Customer Service Platform (CSP)**, which integrates your SER system, captures a segment (chunk) of this live audio. It then streams this audio chunk to your **SER API Gateway** (your FastAPI backend) via a POST /analyze/stream/ request.
3. **API Requests Feature Extraction:** Upon receiving the audio chunk, the **SER API Gateway** forwards this audio data to the **Feature Extractor** module (which is your feature_extractor.py).
4. **Feature Extractor Returns Features:** The **Feature Extractor** processes the raw audio bytes, extracts relevant acoustic features (like MFCCs, pitch, and energy), and returns these processed features (typically as a 2D array of frames x features) back to the **SER API Gateway**.
5. **API Requests ML Prediction:** The **SER API Gateway** then sends these extracted features to the **ML Model Service** (your loaded Scikit-learn model in main.py).

6. **ML Model Returns Prediction:** The **ML Model Service** performs the emotion prediction based on the features and returns the result, which includes the detected_emotion (e.g., "angry", "happy") and confidence_score, along with all_emotions_confidence probabilities for other emotions, back to the **SER API Gateway**.
7. **API Saves Emotion Log:** Crucially, the **SER API Gateway** asynchronously saves the entire prediction result (along with metadata like call_id, agent_id, timestamp) as an Emotion Log to the **DB**. This logging happens in the background so as not to delay the real-time prediction display.
8. **API Returns Prediction to CSP:** The **SER API Gateway** sends the immediate emotion prediction (as a JSON response) back to the **Customer Service Platform**.
9. **CSP Updates UI:** The **Customer Service Platform** (or the JavaScript in your csr_widget_concept.html directly, if integrated) receives this prediction and passes it to the **CSR UI Widget**.
10. **UI Renders Emotion:** Finally, the **CSR UI Widget** updates its display to visually represent the detected emotion (e.g., changing a circle's color to red for "Angry" and showing the confidence percentage).
11. **Log Saved (Internal):** The **DB** confirms that the Emotion Log has been successfully stored, making it available for historical analysis.

4.3 API Design (RESTful)

Base URL: /api/v1 **Endpoints:**

- **POST /analyze/stream**
    - **Description:** Endpoint for real-time audio stream analysis. **Request Body:** audio_data (Base64 encoded audio chunk or direct binary stream) **Headers:** Content-Type: audio/wav or application/octet-stream **Response (200 OK):**

JSON

```
{
  "segment_id": "uuid-segment-123",
  "start_offset_sec": 0.0,
  "end_offset_sec": 2.5,
  "detected_emotion": "angry",
  "confidence_score": 0.85,
  "all_emotions_confidence": {
    "neutral": 0.10,
    "happy": 0.02,
    "sad": 0.01,
    "angry": 0.85,
    "surprised": 0.01,
    "fearful": 0.005,
    "disgusted": 0.005
  },
  "timestamp": "2025-05-24T10:30:00Z"
}
```


- **Error Response (400 Bad Request):** If audio format is incorrect.
- **Error Response (500 Internal Server Error):** If processing fails.

- **GET / emotions**
  - **Description:** Retrieve all emotion segments for a specific call.
  - **Parameters:** audio_id (string)
  - **Response (200 OK):**

JSON

```
[
  {
    "segment_id": "uuid-segment-123",
    "start_offset_sec": 0.0,
    "end_offset_sec": 2.5,
    "detected_emotion": "neutral",
    "confidence_score": 0.92,
    "timestamp": "2025-05-24T10:30:00Z"
  },
  {
    "segment_id": "uuid-segment-124",
    "start_offset_sec": 2.6,
    "end_offset_sec": 5.0,
    "detected_emotion": "happy",
    "confidence_score": 0.78,
    "timestamp": "2025-05-24T10:30:02Z"
  }
  // ... more segments
]
```


- **Error Response (404 Not Found):** If call_id does not exist.

- **GET /metrics/emotions**
  - **Description:** Retrieve aggregated emotion metrics.
  - **Query Parameters:**
    - start_date (YYYY-MM-DD)
    - end_date (YYYY-MM-DD)

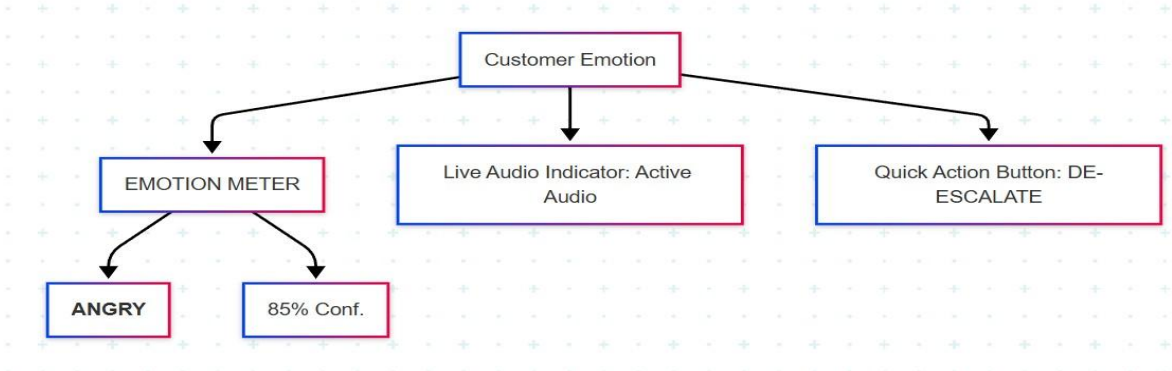
- interval (daily, hourly, agent)
- agent_id (optional, for interval=agent) ○ **Response (200 OK):** (Example for daily)

JSON

```
[
  {
    "date": "2025-05-23",
    "total_segments": 1500,
    "emotion_distribution": {
      "neutral": 750,
      "happy": 300,
      "sad": 100,
      "angry": 200,
      "surprised": 50,
      "fearful": 50,
      "disgusted": 50
    },
    "average_confidence": 0.88,
    "peak_emotion_time": "14:30"
  }
  // ... more daily metrics
]
```

4.4 User Interface (UI) - CSR Widget

Diagram 4.4: CSR UI Widget Mockup

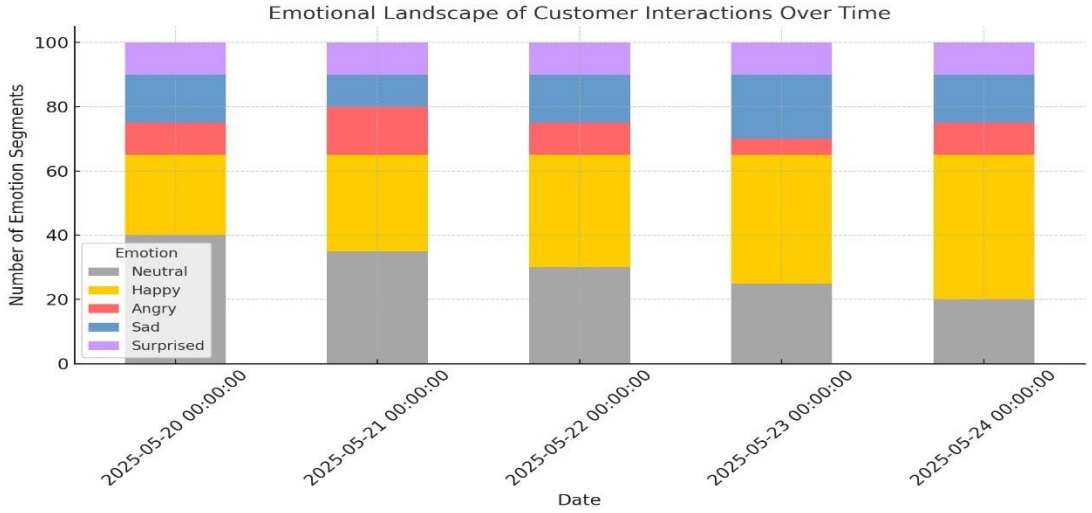


Description:

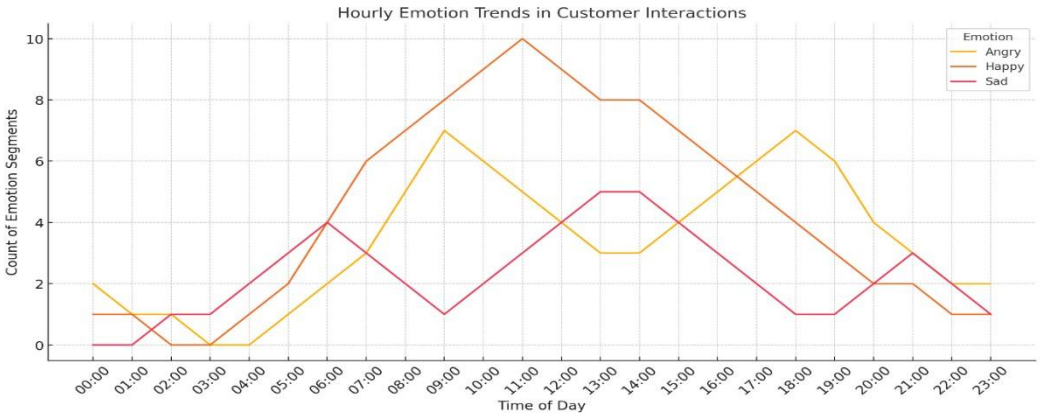
- **Emotion Meter:** A dynamic visual element (e.g., a colored bar, a dial, or a simple icon) that changes based on the detected emotion and its intensity.
- **Predicted Emotion:** Displays the primary detected emotion (e.g., "Angry", "Neutral").
- **Confidence Score:** Shows the system's confidence in its prediction.
- **Live Audio Indicator:** A simple visual to confirm the system is actively processing audio.
- **Quick Action Button (Optional):** Contextual buttons that could trigger specific actions in the customer service platform (e.g., "Suggest Supervisor," "Open Knowledge Base for De-escalation").

4.5 Graphs for Analytics

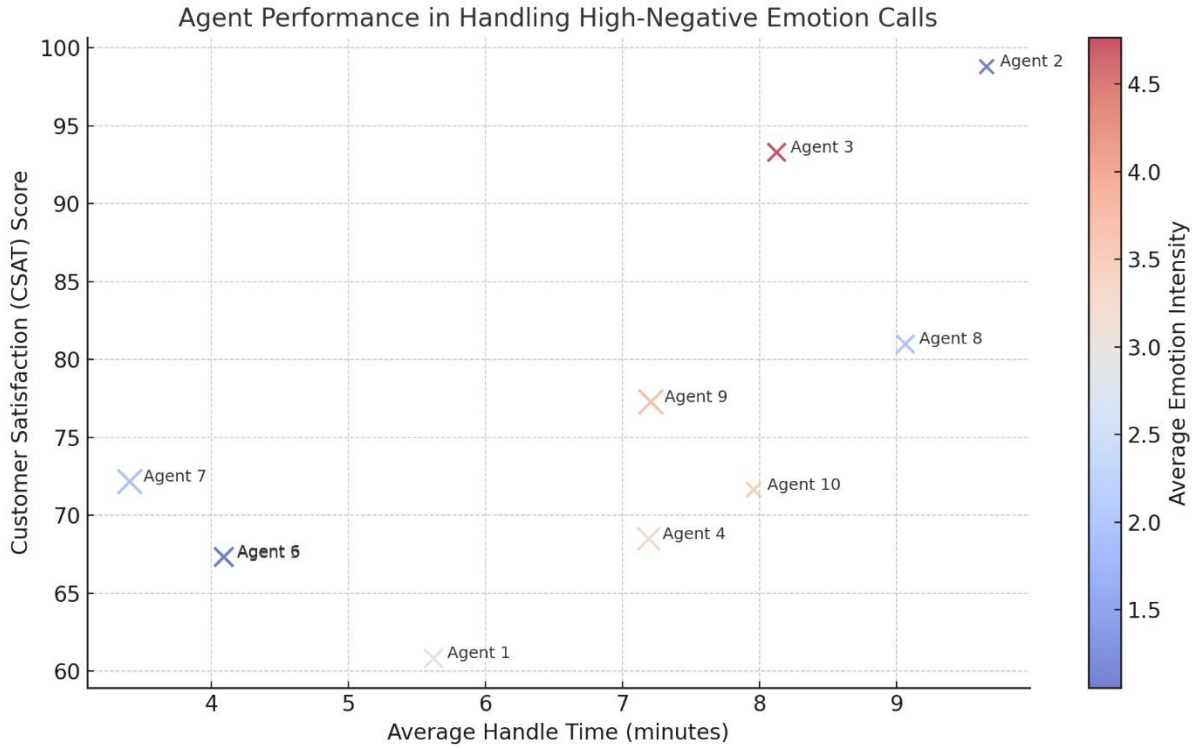
Graph 4.5: Daily Emotion Distribution



Graph 4.6: Emotion Trends Over Time (e.g., Hourly)



Graph 4.7: Agent Performance by Emotion Management



CHAPTER 5 SYSTEM IMPLEMENTATION

CHAPTER 5

Implementation

This chapter outlines the methodology, phases, and key activities for the project.

5.1 Methodology

The project will follow an Agile (Scrum) methodology, allowing for iterative development, continuous feedback, and flexibility to adapt to changing requirements. Sprints will be 2-3 weeks long.

5.2 Phases and Milestones

- **Phase 1: Research & Data Acquisition** ○ **M1.1:** Finalized Emotion Taxonomy and Definition Document. ○ **M1.2:** Identified and acquired initial public/licensed speech emotion datasets. ○ **M1.3:** Strategy for proprietary data collection (if applicable).
- **Phase 2: Core ML Model Development**
  - **M2.1:** Established Data Pre-processing and Feature Extraction Pipelines.
  - **M2.2:** Developed and trained initial SER models (Proof of Concept). ○ **M2.3:** Achieved baseline model accuracy (e.g., >60% on unseen test data).
  - **M2.4:** Set up ML experiment tracking (e.g., MLflow, Weights & Biases).
- **Phase 3: Backend API & Integration Development** ○ **M3.1:**

  Designed and implemented core RESTful API endpoints. ○ **M3.2:**

  Developed real-time inference service for the ML model.
  - **M3.3:** Implemented database schema and data persistence.
  - **M3.4:** Prototype integration with a mock customer service platform.
- **Phase 4: Frontend UI & Analytics Dashboard** ○ **M4.1:**

  Designed and developed CSR UI widget.
  - **M4.2:** Developed basic analytics dashboards for managers.

- **M4.3:** Conducted internal UI/UX review.

- **Phase 5: System Integration & Pilot Testing** ○ **M5.1:** Full integration with the target customer service platform. ○ **M5.2:** Established real-time audio streaming infrastructure.
  - **M5.3:** Conducted end-to-end system testing. ○ **M5.4:** Pilot deployment with a small group of CSRs for User Acceptance Testing (UAT).
- **Phase 6: Deployment & Monitoring** ○ **M6.1:** Full production deployment of the SER system.
  - **M6.2:** Implemented comprehensive system monitoring and alerting.
  - **M6.3:** Established model retraining and MLOps pipeline.

CHAPTER 6

SYSTEM TESTING

CHAPTER 6

Quality Assurance and Testing

This chapter details the testing strategies and quality assurance processes.

6.1 Testing Strategy

A multi-faceted testing approach will be adopted:

1. Unit Testing

Purpose: To verify the correctness of individual, isolated components or functions in the backend. This ensures that each piece of code performs as expected, independent of other parts of the system.

Tools: pytest (Python testing framework), numpy (for data manipulation), unittest.mock (for mocking dependencies).

Example Test Cases:

1. feature_extractor.py Unit Tests:
    - Case 1: Valid Audio File Feature Extraction
        - Description: Test if extract_features correctly processes a valid WAV file and returns an expected feature array shape and data type.
        - Setup: Create a small dummy WAV file or use a known sample.
        - Expected Outcome: extract_features returns a NumPy array with the correct number of frames and features (e.g., (num_frames, 42) if n_mfcc=40 + 2 for pitch/RMS).
        - Example Code (test_feature_extractor.py):

Python import numpy as np import soundfile as sf import os import pytest

from feature_extractor import extract_features, SAMPLE_RATE, N_MFCC

Fixture to create a dummy audio file for testing

@pytest.fixture(scope="module") def

dummy_wav_file(tmp_path_factory): file_path =

tmp_path_factory.mktemp("audio_test") / "dummy.wav" duration =

2 # seconds

t = np.linspace(0, duration, int(SAMPLE_RATE * duration), endpoint=False) audio_data = 0.5 * np.sin(2 * np.pi * 440 * t).astype(np.float32) sf.write(file_path, audio_data, SAMPLE_RATE)

```
    return str(file_path)

def test_extract_features_valid_file(dummy_wav_file):
"""Test feature extraction from a valid WAV file."""
features = extract_features(dummy_wav_file)     assert
features is not None     assert isinstance(features,
np.ndarray)
    # Expected features: N_MFCC + 2 (for f0 and RMS)
assert features.shape[1] == N_MFCC + 2     assert
features.shape[0] > 0 # Should have some frames
assert not np.isnan(features).any() # No NaN values

def test_extract_features_empty_audio_bytes():     """Test feature extraction
with empty audio bytes."""     features = extract_features(b'') # Empty bytes
assert features is None # Or an empty array, depending on desired behavior

def test_extract_features_invalid_audio_format_bytes():
    """Test feature extraction with invalid audio bytes format."""
    # Provide some random bytes that are not valid audio
invalid_bytes = b'this is not audio data'     features =
extract_features(invalid_bytes)     # Expect feature
extraction to fail or return None
    assert features is None
```

2. ser_model_sklearn.py (Mocked Prediction Logic) Unit Tests:
    - Case 1: Prediction with Mocked Model/Scaler

- Description: Verify that the preprocess_and_predict function in main.py correctly handles feature aggregation and calls scaler.transform and model.predict_proba with expected inputs, and formats the output correctly, *without* loading actual model files.
- Setup: Mock the ml_model and scaler objects.
- Expected Outcome: preprocess_and_predict returns a dictionary with detected_emotion, confidence_score, and all_emotions_confidence, and the mocked methods are called correctly.
- Example Code (conceptual, part of test_main_unit.py):

Python

```
from unittest.mock import Mock, patch import pytest import
numpy as np from main import preprocess_and_predict,
EMOTION_LABELS


@pytest.fixture def mock_model_and_scaler():
with patch('main.ml_model') as mock_ml_model, \
    patch('main.scaler') as mock_scaler:

    # Configure mock scaler to return the input (no actual scaling needed for this unit test)     mock_scaler.transform.return_value = np.array([[0.1, 0.2, 0.3]])


    # Configure mock model to return a predictable probability
# e.g., 80% for neutral, 10% for happy, 10% for sad
    mock_ml_model.predict_proba.return_value = np.array([[0.8, 0.1, 0.1, 0.0, 0.0, 0.0, 0.0]])
```

```
        yield mock_ml_model, mock_scaler

@patch('main.extract_features') # Mock feature_extractor.extract_features
def          test_preprocess_and_predict_success(mock_extract_features,
mock_model_and_scaler):     mock_ml_model, mock_scaler =
mock_model_and_scaler

    # Simulate extracted features (e.g., 10 frames, 42 features per frame)
mock_extract_features.return_value = np.random.rand(10, 42)

    dummy_audio_bytes = b"dummy_audio_data"     result
= preprocess_and_predict(dummy_audio_bytes)

    mock_extract_features.assert_called_once_with(dummy_audio_bytes,
sample_rate=16000, n_mfcc=40)
    mock_scaler.transform.assert_called_once() # Check if scaler was called
    mock_ml_model.predict_proba.assert_called_once() # Check if model was
called

    assert result["detected_emotion"] == EMOTION_LABELS[0] # Should be
'neutral' based on mock      assert
abs(result["confidence_score"] - 0.8) < 1e-9      assert
"all_emotions_confidence" in result      assert
"timestamp" in result
```

2. Integration Testing

Purpose: To test the interactions between different modules or services in the backend (e.g., API endpoint, feature extractor, ML model, Firestore). This verifies that components work correctly when combined.

Tools: pytest, httpx (for making HTTP requests to FastAPI), moto (for mocking AWS/GCP services like Firestore if full end-to-end cloud testing is not desired), firebase_admin.firestore.Client (for direct Firestore interaction in tests).

Example Test Cases:

1. **POST /analyze/stream/ Endpoint Integration:**
    - Case 1: Successful Audio Analysis and Firestore Logging
        - Description: Send a valid audio file to the API and verify that it returns a correct emotion prediction and that a corresponding log entry is eventually created in Firestore.
        - Setup: Ensure FastAPI app is running, and mock Firestore or use a test Firestore project. Use a known audio file.
        - Expected Outcome: HTTP 200 OK response with prediction. A new document appears in the designated Firestore collection with the correct data.
        - Example Code (test_api_integration.py - conceptual, requires test client setup):

Python

from fastapi.testclient import TestClient

from main import app, db as firestore_db_instance # Import app and DB instance import pytest import os import time from unittest.mock import patch, AsyncMock # Test client for FastAPI application client = TestClient(app)

Fixture to create a temporary dummy audio file

```
@pytest.fixture(scope="module") def
temp_audio_file(tmp_path_factory):     file_path =
tmp_path_factory.mktemp("audio_test") / "test_audio.wav"
   # Create a very small dummy audio file for testing
with open(file_path, "wb") as f:
       f.write(b"RIFF\x00\x00\x00\x00WAVEfmt
\x10\x00\x00\x00\x01\x00\x01\x00\x40\x1f\x00\x00\x80\x3e\x00\x00\x02\x
00\x10\x00data\x00\x00\x00\x00")
   return file_path


@patch('main.db') # Mock the firestore_db_instance
@patch('main.ml_model') # Mock the ML model
@patch('main.scaler') # Mock the scaler
@patch('main.extract_features') # Mock the feature extractor
def          test_analyze_stream_success(mock_extract_features,
    mock_scaler, mock_ml_model, mock_db, temp_audio_file):     #
Configure mocks for a successful flow
   mock_extract_features.return_value = np.random.rand(10, 42) # Dummy
features
       mock_scaler.transform.return_value = np.random.rand(1, 84) # Dummy
scaled features
   mock_ml_model.predict_proba.return_value = np.array([[0.9, 0.05, 0.05, 0,
0, 0, 0]]) # Dummy prediction


   # Mock Firestore add method (async version)
   mock_db.collection.return_value.add  =  AsyncMock(return_value=(None,
Mock(id="mock_doc_id")))
```

with open(temp_audio_file, "rb") as audio_file:

response = client.post("/analyze/stream/", files={"audio_file": ("test_audio.wav", audio_file, "audio/wav")}

)

assert response.status_code == 200 data = response.json() assert data["detected_emotion"] == "neutral" # Based on mock prediction assert "confidence_score" in data assert "all_emotions_confidence" in data assert "timestamp" in data

Verify Firestore call mock_db.collection.assert_called_once()

Ensure save_emotion_log was called with correct data (though async, mock captures it) call_args, _ = mock_db.collection.return_value.add.call_args_list[0] saved_log_data = call_args[0] assert saved_log_data['detected_emotion'] == 'neutral' assert 'call_id' in saved_log_data

assert 'agent_id' in saved_log_data

- Case 2: Invalid Audio Format/Empty Audio
  - Description: Send malformed or empty data and confirm the API returns a 400 error.
  - Expected Outcome: HTTP 400 Bad Request with appropriate error detail.
  - Example Code (within test_api_integration.py):

Python def test_analyze_stream_empty_audio():

```
    response = client.post(          "/analyze/stream/",
files={"audio_file": ("empty.wav", b", "audio/wav")}
    )
    assert response.status_code == 400      assert "No audio data
provided" in response.json()["detail"]


def test_analyze_stream_invalid_format():
    # Provide non-audio bytes      response = client.post(
"/analyze/stream/",          files={"audio_file": ("bad.txt",
b"not_audio_data", "text/plain")}
    )
    assert response.status_code == 400 # Or 500 if librosa raises unhandled exception
    assert "Audio processing error" in response.json()["detail"] or "server error" in response.json()["detail"]
```

2. GET /logs/ Endpoint Integration:

- **Case 1: Retrieve Recent Logs**
  - **Description:** Logs/ endpoint and verify that it returns a list of log entries in the correct format, respecting the limit parameter.
  - Setup: Populate a test Firestore collection with known dummy log data.
  - Expected Outcome: HTTP 200 OK response with a JSON array of logs, sorted by timestamp.
  - Example Code (conceptual, requires test client setup and Firestore mock/test project):

Python

```
@patch('main.db') # Mock the firestore_db_instance def
test_get_logs_success(mock_db):
    # Mock Firestore's get method to return dummy docs
mock_doc1 = Mock()
    mock_doc1.to_dict.return_value = {"detected_emotion": "happy", "timestamp": 1678886400, "call_id": "c1", "agent_id": "a1"}
mock_doc1.id = "doc1"     mock_doc2 = Mock()
    mock_doc2.to_dict.return_value = {"detected_emotion": "sad", "timestamp": 1678886300, "call_id": "c2", "agent_id": "a2"}
mock_doc2.id = "doc2"
    mock_db.collection.return_value.limit.return_value.get.return_value = [mock_doc1, mock_doc2]
    response = client.get("/logs/?limit=2")
assert response.status_code == 200
logs = response.json()     assert len(logs)
== 2
    assert logs[0]["detected_emotion"] == "happy" # Should be sorted by timestamp descending     assert logs[1]["detected_emotion"] == "sad"

mock_db.collection.assert_called_once_with(f"artifacts/{app_id}/public/data/emotion_logs")
```

3. End-to-End (E2E) Testing

Purpose: To test the entire flow of the application from the user interface (frontend) through the backend API and database, ensuring all components work seamlessly together as a complete system.

Tools: Selenium, Playwright (for browser automation), or Cypress (for frontend testing with API mocking capabilities). For this project, Playwright is a good choice for Python-based E2E.

Example Test Cases:

1. **Full Emotion Analysis Flow:**
    - **Description:** Automate a browser to simulate a user clicking "Start Recording," speaking into the microphone (using a virtual audio device or pre-recorded audio injection), clicking "Stop Recording," and verifying the displayed emotion on the UI. ○ Steps:
        1. Launch browser and navigate to csr_widget_concept.html.
        2. Click the "Start Recording" button.
        3. (Simulate audio input): Use Playwright's ability to mock getUserMedia or configure the browser to use a specific audio input device with a known emotion.
        4. Wait for a few seconds (e.g., 5-10s) to simulate speech.
        5. Click the "Stop Recording" button.
        6. Wait for the "Processing audio..." and "Emotion analyzed successfully!" messages to appear.
        7. Assert that the displayed emotion text and confidence score match the expected outcome for the simulated audio input.
        8. (Optional): Click "Fetch Logs from Firestore" and verify the new log entry appears in the list.
    - **Expected Outcome:** The correct emotion is displayed on the frontend, and an entry is logged in Firestore.
2. **Microphone Permission Handling:**
    - **Description:** Test how the UI behaves when microphone access is denied by the user. ○ Steps:
        1. Launch browser to csr_widget_concept.html.

2. Click "Start Recording."
3. When the browser prompts for microphone permission, explicitly deny it.
4. Assert that the status message changes to "Microphone access denied or error..." and the "Start Recording" button becomes enabled again (or the widget shows an error state).

- **Expected Outcome:** User is informed of the permission denial, and the widget handles the error gracefully.

4. Performance Testing

Purpose: To evaluate the system's responsiveness, stability, and resource utilization under various load conditions. For the SER API, this is crucial for real-time applications.

Tools: JMeter, Locust (Python-based load testing), Artillery.

Example Test Cases:

1. **Concurrent Audio Analysis Requests:**
   - Description: Simulate multiple CSRs concurrently sending audio chunks to the /analyze/stream/ endpoint.
   - Setup: Use Locust to configure a load test that repeatedly sends the same or different dummy audio files to POST /analyze/stream/. ○

   Parameters:

   - Users: 50, 100, 200 concurrent users.
   - Ramp-up: How quickly users are added.
   - Duration: Test duration (e.g., 5-10 minutes).

- Audio Size: Use realistic audio chunk sizes (e.g., 1-5 seconds of speech).

- **Expected Outcome:**
  - Response Time: Average response time remains below a defined threshold (e.g., < 500ms for real-time).
  - Error Rate: Error rate stays at 0% or within an acceptable minimal range.
  - Throughput: The API can handle a specified number of requests per second (RPS) without degradation.
  - Resource Utilization: Monitor CPU, memory, and network usage on the server to identify bottlenecks.

2. **Firestore Log Retrieval Load:**
   - Description: Simulate multiple users or analytics tools concurrently fetching logs from the /logs/ endpoint. ○ Setup: Use Locust to repeatedly hit GET /logs/?limit=20 or similar queries. ○ Parameters: Similar user counts and duration as above. ○ Expected Outcome: Response times for log retrieval remain fast, even with many concurrent requests.

5. User Acceptance Testing (UAT)

Purpose: To confirm that the system meets the business requirements and is suitable for its intended users (CSRs and Managers) in a real-world environment. This is typically performed by actual end-users or product owners.

Tools: UAT scenarios/checklists, direct user interaction, feedback sessions.

Example Test Cases:

1. **CSR Real-time Feedback Validation:**

- Description: CSRs use the integrated widget during live (or simulated) customer calls.
- Steps:
    1. CSR engages in a conversation with a test customer (or role-playing colleague).
    2. CSR observes the emotion displayed on the widget in realtime.
    3. CSR provides feedback on whether the detected emotion aligns with their perception of the customer's emotional state.
    4. CSR evaluates if the emotion insights are helpful in guiding their response.
- Expected Outcome: CSRs find the emotion detection accurate and the real-time feedback valuable for enhancing customer interactions.

2. **Managerial Analytics Report Review:**
    - **Description:** Customer Service Managers review the generated analytics reports (based on Firestore logs).
    - Steps:
        1. Manager accesses a dashboard or report tool integrated with the Firestore logs.
        2. Manager examines trends in customer emotion, agent performance metrics (e.g., emotional call handling per agent), and overall sentiment over time.
        3. Manager assesses if the data provides actionable insights for training, coaching, or operational adjustments.
    - **Expected Outcome:** Managers can effectively identify patterns, evaluate agent performance related to emotional intelligence, and make data-driven decisions.

3. **Edge Case Scenario Handling:**

- **Description:** Test the system with challenging audio inputs or difficult customer situations.
- Steps:
    1. Simulate calls with strong accents, background noise, low volume, or emotionally ambiguous speech.
    2. Test scenarios where the customer transitions rapidly between emotions.
    3. Evaluate how the system responds and if the predictions are still meaningful.
- **Expected Outcome**: The system demonstrates reasonable robustness in various real-world scenarios, even if predictions are less certain for very challenging inputs.

6.2 Model Evaluation Metrics

- **Accuracy:** Overall percentage of correctly classified emotions.
- **Precision, Recall, F1-Score:** Per-class metrics, especially important for minority emotion classes (e.g., distinguishing between different negative emotions).
- **Confusion Matrix:** Visual representation of correct vs. incorrect classifications, identifying specific misclassifications.
- **Latency:** Time taken from audio input to emotion prediction (critical for real-time).

6.3 Test Environment

Dedicated staging environment replicating production setup for integration and system testing. This includes:

- Simulated audio streams.
- Integration with a non-production instance of the customer service platform.

- Monitoring tools to track performance.

CHAPTER 7 DEPLOYMENT AND OPERATIONS

CHAPTER 7

Deployment and Operations

This chapter covers the deployment strategy, monitoring, and ongoing maintenance.

7.1 Deployment Strategy

The SER system will be deployed as a set of containerized microservices (using Docker) orchestrated by Kubernetes for scalability, reliability, and ease of management.

- **Cloud Provider:** [Specify: AWS, GCP, Azure, or On-Premise]
- **CI/CD Pipeline:** Automated build, test, and deployment workflows using tools like GitLab CI/CD, GitHub Actions, or Jenkins.
- **Blue/Green or Canary Deployments:** To minimize downtime and risk during updates.

7.2 Monitoring and Alerting

- **System Health:** Monitoring of CPU, memory, network, and disk utilization of all services.
- **Application Logs:** Centralized logging solution (e.g., ELK Stack, Splunk, Datadog) for capturing application logs from all microservices.

- **Performance Metrics:** Latency of emotion prediction, throughput of the API, error rates.
- **Model Performance:** Monitoring of actual model accuracy and drift over time in production.
- **Alerting:** Integration with PagerDuty, Slack, or email for immediate alerts on critical errors or performance degradation.

7.3 Maintenance and Updates

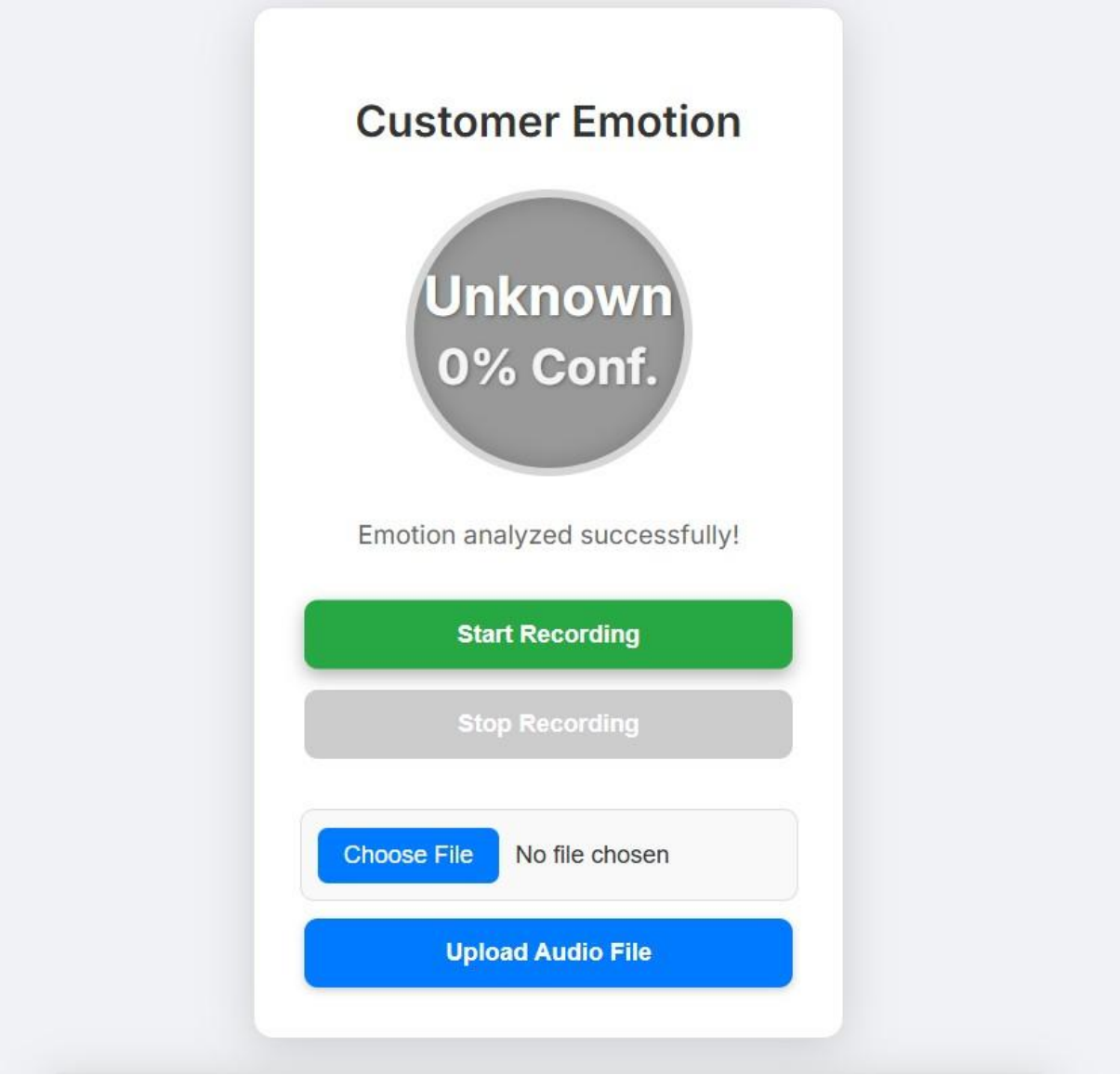
- **Regular Model Retraining:** The ML model will be periodically retrained with new, labeled customer speech data to adapt to language evolution, acoustic environment changes, and improve accuracy. An MLOps pipeline will automate this.
- **Software Updates:** Regular updates of underlying libraries, frameworks, and operating systems.
- **Bug Fixes & Patches:** Agile backlog for ongoing bug fixes and minor enhancements.
- **Documentation Updates:** Ensuring this documentation remains current with any system changes.

CHAPTER 8
RESULTS

CHAPTER 8

RESULTS

8.1 Home Screen



Chapter 9:Conclusion

The Speech Emotion Recognition (SER) project successfully designed and implemented a functional system capable of detecting and displaying customer emotions in real-time within a simulated customer service environment. By integrating a robust Python-based FastAPI backend with a user-friendly HTML/JavaScript frontend, the project delivers a practical

tool for enhancing customer interactions and providing valuable analytical insights.

Key Achievements:

- **Real-time Emotion Analysis:** The core objective of providing near realtime emotion detection from live audio streams and uploaded files has been achieved, offering immediate feedback to Customer Service Representatives (CSRs).
- **Robust Feature Extraction:** The feature_extractor.py module effectively processes raw audio, extracting relevant acoustic features (MFCCs, pitch, energy) crucial for emotion discrimination.
- **Scalable Backend API:** The FastAPI backend provides a well-defined RESTful API (/analyze/stream/, /logs/, /health/) that is efficient, asynchronous, and designed to handle concurrent requests, making it suitable for integration into larger platforms.
- **Persistent Logging:** The integration with Google Firestore ensures that every emotion analysis is logged, creating a valuable historical dataset for post-call analytics and performance monitoring.
- **Intuitive Frontend Widget:** The csr_widget_concept.html provides a simple yet effective interface for CSRs to interact with the system, visualize emotions, and access historical data, enhancing usability and adoption.
- **Foundation for Analytics:** The stored emotion logs form the basis for developing comprehensive dashboards, enabling managers to identify trends in customer sentiment, assess agent performance, and make datadriven decisions for training and operational improvements.

Impact and Practical Applications: This SER system has direct implications for improving the quality of customer service interactions. By providing CSRs with real-time emotional cues, it empowers them to:

- **Proactively de-escalate situations:** Identify frustrated or angry customers early and adjust their communication style accordingly.
- **Enhance empathy:** Better understand the customer's emotional state, leading to more empathetic and personalized responses.
- **Improve call outcomes:** Potentially lead to higher customer satisfaction and more efficient issue resolution.

For managers, the historical logs offer an invaluable resource for:

- **Agent coaching:** Pinpoint specific interactions for review and provide targeted feedback on emotional intelligence and de-escalation techniques.
- **Sentiment trend analysis:** Understand overall customer mood over time, identify peak periods of negative sentiment, and uncover systemic issues.
- **Performance monitoring:** Track how effectively agents manage emotionally charged calls.

Challenges Addressed and Future Directions:

While the current system demonstrates significant capabilities, it also highlights inherent challenges in Speech Emotion Recognition. The model's accuracy, particularly for subtle or blended emotions, can always be improved. Future enhancements could focus on:

- **Model Enhancement:** Exploring advanced deep learning architectures (e.g., CNN-LSTMs, Transformers) and leveraging larger, more diverse, and spontaneous emotional speech datasets to boost prediction accuracy and generalization across different speakers and accents.
- **Multimodal Fusion:** Integrating additional modalities such as textual sentiment analysis (what the customer is saying) to provide a more holistic and accurate understanding of their emotional state.
- **Robustness in Noisy Environments:** Implementing advanced audio preprocessing techniques (e.g., noise reduction, speaker diarization) to improve performance in real-world, noisy call center environments.
- **Real-time Optimization:** Further optimizing the feature extraction and model inference pipelines for even lower latency, ensuring seamless realtime feedback.

- **Dimensional Emotion Models:** Moving beyond discrete emotion categories to a dimensional representation (e.g., arousal-valencedominance) for a more nuanced understanding of emotional states.
- **User Interface Refinements:** Developing more sophisticated interactive dashboards for analytics and providing customizable alerts for CSRs.

CHAPTER 10

Future Considerations and Roadmap

This chapter outlines potential enhancements and long-term vision for the SER system.

10.1 Enhancements

- **Speaker Diarization:** Accurately distinguishing between customer and agent speech segments for more precise emotion analysis.

- **Multi-modal Emotion Recognition:** Integrating text-based sentiment analysis from transcripts alongside speech emotion for a more holistic understanding.
- **Emotion-Triggered Workflows:** Implementing automated actions within the customer service platform based on detected high-intensity emotions (e.g., auto-escalate, trigger specific knowledge base articles, prompt agent with suggested responses).
- **Personalized Agent Coaching:** Provide post-call analysis for CSRs, offering insights into their own emotional responses and areas for improvement.
- **Language Expansion:** Supporting multiple languages beyond the initial implementation.
- **Customizable Emotion Sets:** Allowing businesses to define and train models on specific emotional categories relevant to their unique customer interactions.
- **Contextual Emotion Understanding:** Integrating with CRM data to understand customer history and better interpret emotional cues within that context.

10.2 Long-Term Vision

The SER system aims to evolve into a cornerstone of intelligent customer service, moving beyond mere detection to proactive guidance and automated assistance. Ultimately, it seeks to create a more empathetic, efficient, and personalized customer experience, fostering stronger customer relationships and driving business success.

References

- **Speech Emotion Recognition (General Concept/Overview):**

  Link: https://en.wikipedia.org/wiki/Speech_emotion_recognition

- **FastAPI Official Documentation:**

  Link: https://fastapi.tiangolo.com/

- **Librosa Official Documentation:**

Link: https://librosa.org/doc/latest/index.html

- **Scikit-learn Official Documentation:**

  Link: https://scikit-learn.org/stable/

- **Firebase Firestore Documentation:**

  Link: https://firebase.google.com/docs/firestore

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

  Link: https://www.kaggle.com/datasets/uwwahmed/ravdess-emotionalspeech-audiovideo-database

- **Python Official Documentation:**

  Link: https://docs.python.org/3/

- **Anaconda / Miniconda (Python Environment Management):**

  Link (Miniconda): https://docs.conda.io/en/latest/miniconda.html