Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B. Tech. CE Semester – VI

Subject: SYSTEM DESIGN PRACTICE (MINI PROJECT)

Project Title: English Accent Detection

Guided by:

Prof. Jignesh K. Shah
Assistant Professor

Dharmsinh Desai University

Faculty of Technology, College Road, Nadiad – 387001, Gujarat

CERTIFICATE

This is to certify that the term work carried out in the subject of **SYSTEM DESIGN PRACTICE (MINI PROJECT)** and submitted is the bonafide work of **Dhameliya Divyang R.**, Roll No.: **CE098**, Identity No.: **22CEUOS147**, and **Dobariya Vrund G.**, Roll No.: **CE101**, Identity No.: **22CEUOS146**, and **Gajera Rushang M.**, Roll No.: **CE107**, Identity No.: **22CEUOS068**, of B.Tech. Semester VI in the branch of Computer Engineering during the academic year **2024-2025**.

Prof. Jignesh Shah

Assistant Professor

CE Department

Dr. C. K. Bhensdadia

Head of Department

CE Department

English Accent Detection

March 2025

Contents

Contact Information

Name	Email Address
Dhameliya Divyang R.	dhameliyadivyang@gmail.com
Dobariya Vrund G.	vrund3626@gmail.com
Gajera Rushang M.	gajerarushang17@gmail.com

1 Introduction

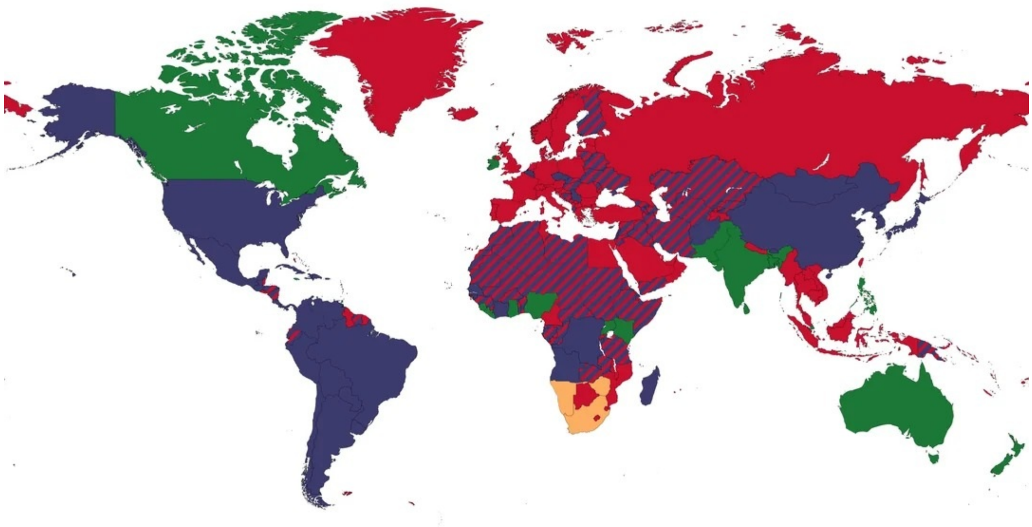


Figure 1. **Variation of English Accents Across Different Regions**

1.1 Project Overview

Our project aims to solve the problem of **accent detection** for the English language. English has overtaken other languages in terms of the number of speakers, both native and non-native. This widespread adoption of a single language on a global scale has led to numerous variations in pronunciation.

Speakers from different regions develop distinct accents for the same language, which can be either unique or a mixture of multiple influences. The determining factors for an accent include mother tongue, personal speech habits, historical background, and social influences. These factors shape how individuals pronounce words, forming recognizable patterns of speech that differentiate one accent from another.

1.2 Problem Statement

This project aims to develop a Deep-learning-based model for **English accent classification**.

So, given a set of audio samples:

$X = \{x_1, x_2, ..., x_n\}, \quad x_i \in R^d$ (d-dimensional vector of features representing an audio wave.)

The objective is to learn a function:

$$f : X \to Y, \quad Y = \{y_1, y_2, ..., y_m\} \text{ (correct accent labels)}$$

such that the classification accuracy is maximized:

$$\max_f \quad P(y_i|x_i), \quad \forall i \in \{1, 2, ..., n\}$$

where f is a deep learning model mapping speech features to accent labels.

1.3 Use-case

In today's world, text-to-speech (tts) systems have become an integral part of our daily lives. Modern voice assistants are capable of understanding a wide range of accents, as they are built upon complex NLP-based models trained on vast datasets. However, the level of personalization can still be significantly improved.

By training several tts models on accent-specific datasets, we can generate speech outputs that match a user's accent more naturally. This creates a more personalized experience. To achieve this, **an accurate accent detection model is essential**. Such a model can effectively identify the user's accent, enabling the system to select and use the most appropriate accent-specific model for generating response audio.

2 Literature Review

Audio classification is a fundamental task in speech and sound processing. Applications such as music genre classification, environmental sound recognition, and accent detection are widely used. Several approaches have evolved over time for this type of problem.

2.1 Sound waves and Audio pre-processing [4]

In any approach (ML-based or DL-based), some steps are common to build speech-related applications.

2.1.1 Sound Waves and Their Properties

Sound waves are mechanical and require a medium to propagate. They cause air molecules to oscillate, creating pressure variations. Some properties of sound waves:

- **Amplitude:** Determines loudness of sound.
- **Frequency:** The number of oscillations per second.
- **Pitch:** The perceived highness or lowness of a sound.
- **Intensity:** Measures the energy per unit area per unit time.

2.1.2 Continuous-Time Representation of Audio Signals

An audio signal in the continuous domain is represented as:

$$x(t) = \sum_{k=1}^{N} A_k \cos(2\pi f_k t + \phi_k) \tag{1}$$

$x(t)$: The audio signal as a function of time t.
N: Number of frequency components.
A_k: Amplitude of the k-th frequency, representing its loudness.
f_k: The frequency of the k-th component
ϕ_k: Phase of the k-th component, determining the starting point of the wave.

2.1.3 Time vs. Frequency Domain Analysis

- **Time domain:** Observing the amplitude of the sound wave w.r.t. time for a given frequency (keeping frequency fixed).
- **Frequency domain:** Observing the amplitude w.r.t. frequency at a given time.

One thing to keep in mind is that sound generated by a source consists of multiple frequency components rather than a single frequency. [4]

2.1.4 Analog to Digital Conversion

As computers work on digital data, there is a need to convert the analog wave to its digital format. For that, two steps are required:

- **Sampling:** Converts a continuous-time signal into a discrete-time signal by measuring amplitudes at regular intervals.
- **Quantization:** Maps continuous amplitude values to a finite set of discrete values rather than considering amplitude as a continuous variable.

A higher sampling rate retains the signal quality but increases the memory usage, so it requires a trade-off.

2.1.5 Loading the Data

Several libraries exist for audio processing in Python, including Librosa (which we used). These libraries return the audio signal represented as an array representing amplitude over time.

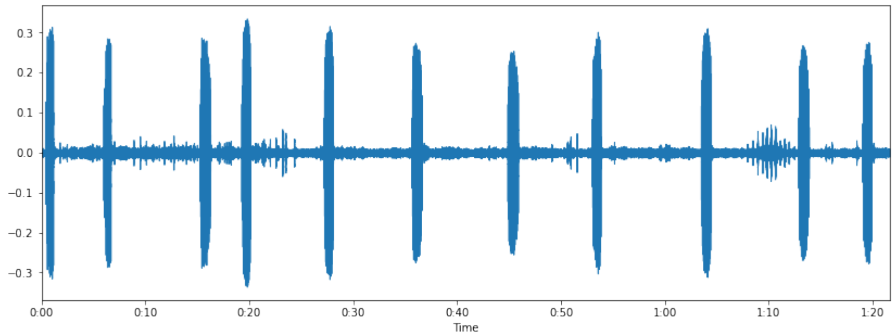


Figure 2: Example waveform representation of an audio signal (amplitude vs time)

The array length for storing the audio is:

$$N = \text{duration} \times \text{sampling rate}$$

ex. To store just 1 second at a sampling rate of 16000, we need an array with 16000 entries, hence pretty memory-consuming.

2.1.6 Problem with Recorded Audio Data

Suppose some audio produced by a source is recorded by a microphone. Figure 2 represents amplitude (vertical axis) over time (horizontal axis), known as a time-domain representation.

If we think in a reverse manner, this captured audio is the resultant waveform generated by the interference of different frequency components. (Figure 3). Analyzing the wave in this form is very difficult, so we need to separate out this frequency components from this recorded audio. This task can be done by applying a Fourier transform on audio signal.

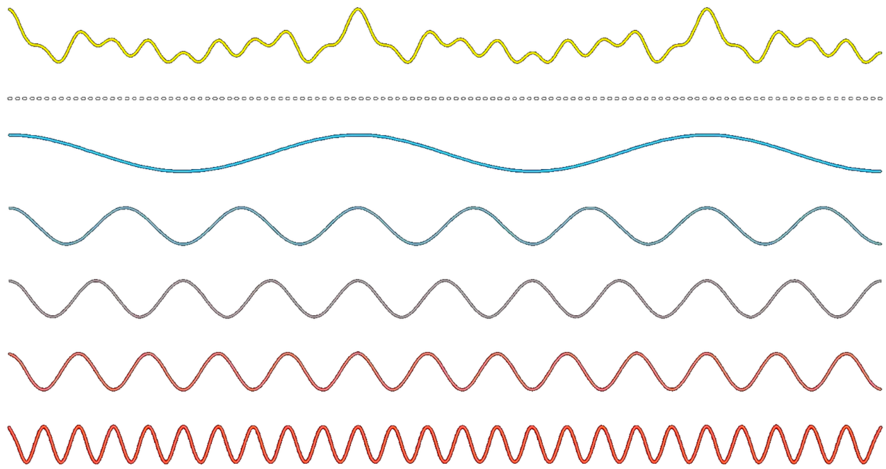


Figure 3: Resultant wave composed of multiple frequencies

Time-frequency domain Observing signals purely in the time or frequency domain provides limited insights. A better approach is a three-dimensional representation with time, frequency, and amplitude.

2.2 Fourier Transform and Spectrograms

Note: A detailed explanation of this topic is given in this article [6].

As per the mathematical equation for a sound wave (Equation 1), we know that a sound wave is a summation sinusoidals. where each of the sinusoidal terms represents a different frequency component. The Fourier Transform (FT) allows us to decompose a signal into its constituent sinusoidal components. This is essential in analyzing audio signals, which are typically composed of multiple frequencies.

2.2.1 Fourier Series

The Fourier series expresses a periodic function in time with period T as a summation of sinusoidal terms:

$$f(t) = \sum C_n e^{j\omega_n t} \tag{2}$$

where $f(t)$ is the signal, C_n are the Fourier coefficients representing the amplitude of each frequency component, and ω_n represents the angular frequencies.

The Fourier coefficients c_n are determined by:

$$C_n = \frac{1}{T} \int_0^T f(t) e^{-j\omega_n t} dt \tag{3}$$

which ensures that each coefficient captures the contribution of a particular frequency.

Using Euler's formula, we rewrite the Fourier series:

$$f(t) = \sum_{n=-\infty}^{\infty} C_n e^{j2\pi nt/T} \tag{4}$$

where C_n represents the amplitude of each frequency component.

2.2.2 Fourier Transform

For a function with a very wide frequency spectrum, we extend the limits of integration, leading to the Fourier Transform:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \tag{5}$$

where $F(\omega)$ represents the frequency domain representation of $f(t)$.

The inverse Fourier Transform is given by:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}d\omega \tag{6}$$

which converts a frequency-domain representation back into the time-domain signal.

2.2.3 Discrete Fourier Transform (DFT)

For digital signals, the Discrete Fourier Transform (DFT) is used:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \tag{7}$$

where $X(k)$ represents the frequency components, $x(n)$ is the discrete signal (audio signal in our case), N is the total number of samples, and k is the frequency index.

The inverse DFT (IDFT) reconstructs the time-domain signal:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N} \tag{8}$$

2.2.4 Short-Time Fourier Transform (STFT)

To get the audio wave in terms of both frequency and time (for the ease of analysis), we apply the Short-Time Fourier Transform (STFT), which at the end applies the DFT to **short**, **overlapping** segments of the signal:

$$STFT(t, f) = \sum x(n)w(n-t)e^{-j2\pi fn} \tag{9}$$

where $STFT(t, f)$ represents the time-frequency representation, $w(n)$ is a window function to limit the segment size and reduce **spectral leakage** [7], and $x(n)$ is the input signal.

2.2.5 Spectrograms

After performing STFT, we get our signal as a function of both time and frequency. Spectrograms are a way of visualizing this multivariate discrete function. (Figure 4).

These spectrograms can also be used as inputs to machine learning models like CNNs but require further processing for building more accurate sytems.

2.3 Mel Frequencies and MFCCs

For a detailed explanation: [8].

Human Perception of Sound [12] Amplitude primarily influences loudness perception. A 70 dB sound appears louder than a 50 dB sound. However, frequency also plays a role. humans perceive lower frequencies more sensitively than higher ones given a constant amplitude.

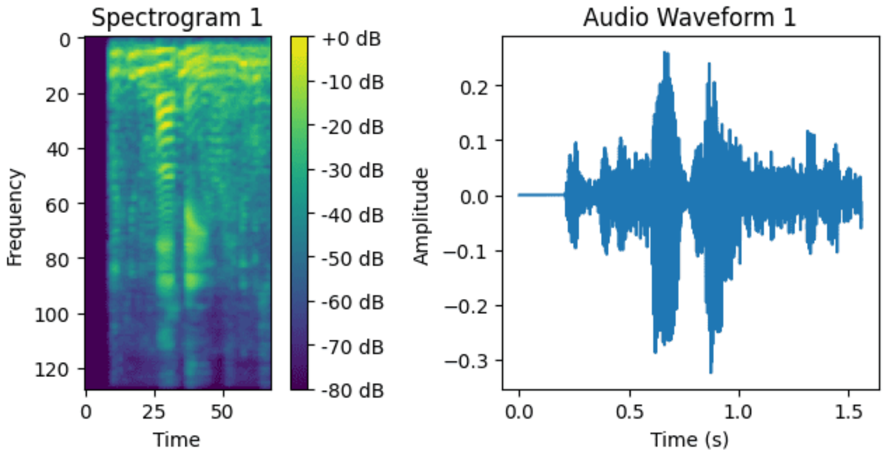


Figure 4: Spectrogram of an audio signal

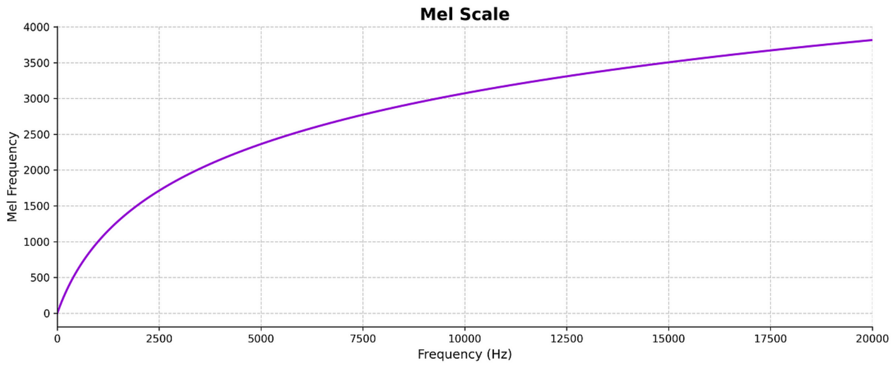


Figure 5: mel-scale

Mel Scale and Mel Spectrograms Humans perceive frequencies logarithmically, which is more accurately captured by the Mel scale. The relation between frequency and Mel scale is (Figure: 5)

$$M(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{10}$$

$$f = 700\left(10^{\frac{M}{2595}} - 1\right) \tag{11}$$

Though the mel-spectrograms work fine, there is a pitfall. The features represented by mel-spectrograms are highly correlated. The deep learning-based model may work fine, but the algorithms sensitive to multicollinear features would stuck. And keeping unnecessary features increases computational cost. There is a **requirement for dimensionality reduction**.

2.3.1 MFCCs (Mel-Frequency Cepstral Coefficients)

MFCCs can be obtained by performing the Discrete cosine transform (DCT) of the mel-spectrograms.

$$C_n = \sum_m \log S_m \cos\left[n\frac{\pi}{M}(m + 0.5)\right] \tag{12}$$

- C_n: Co-efficients.

- S_m: Function representing mel-spectrogram of the wave.
- m: Index representing discrete spectral bins in the summation.
- n: Index representing the output coefficients in the transformation.
- M: Total number of frequency bins (term **bin** is used, as the signal is a discrete function).

These MFCCs serve as features for audio tasks and can be directly used as features for training the model.

3 Analysis

3.1 Exploratory Data Analysis

3.1.1 Dataset Description

Name: Speech Accent Archive
Source: Kaggle (OpenSLR)
Total number of Samples: 2,138
Languages: English (native and non-native speakers)
Accent Labels: 171 different accents (out of which some don't have a sufficient amount of recordings)
Data Format: .mp3 audio + Transcriptions

*Every speaker speaks the same text.

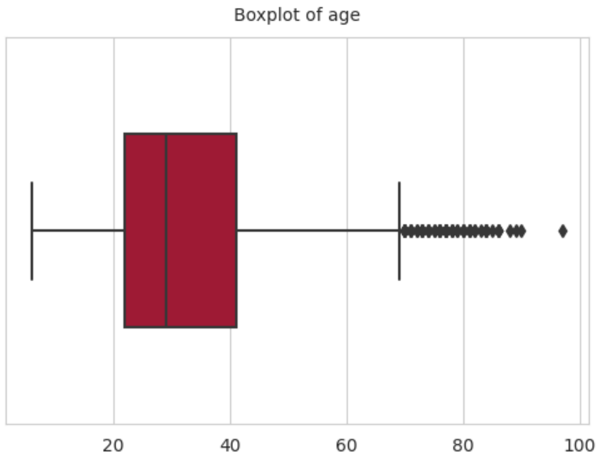


Figure 6: Age distribution of speakers

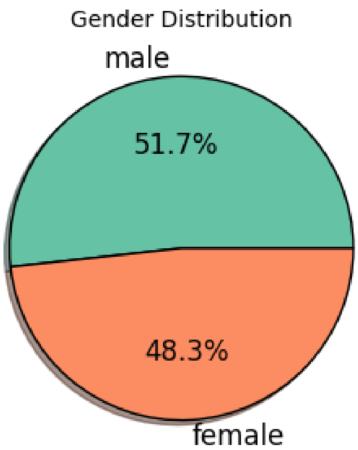


Figure 7: Gender distribution

Analyzing the distribution of gender makes sense, as men tend to have thicker and longer vocal cords, resulting in lower pitches, while women have shorter and thinner vocal cords, producing higher pitches. Age contributes for changes in voice as well.

3.1.2 Accent Distribution

The frequency distribution of different accent labels is illustrated in Figure 8. The dataset has an imbalanced distribution, with certain accents being more represented than others.

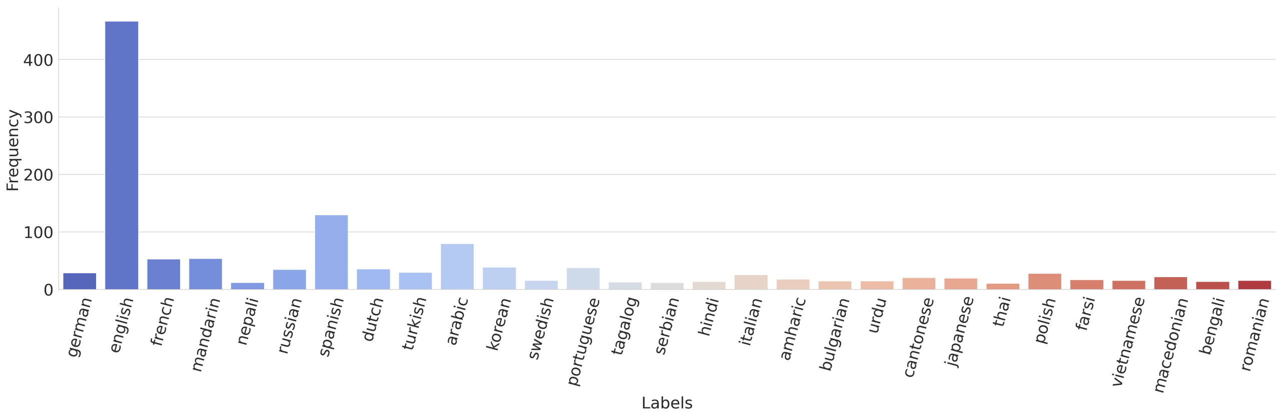


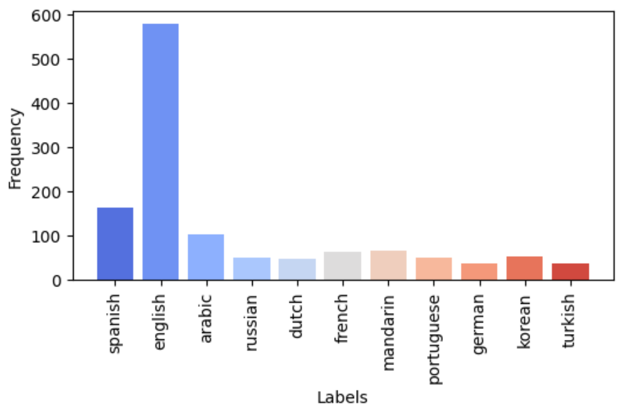
Figure 8: Frequency distribution of accents in the dataset. (If the number of recordings is more than 10)

3.1.3 Data Preprocessing

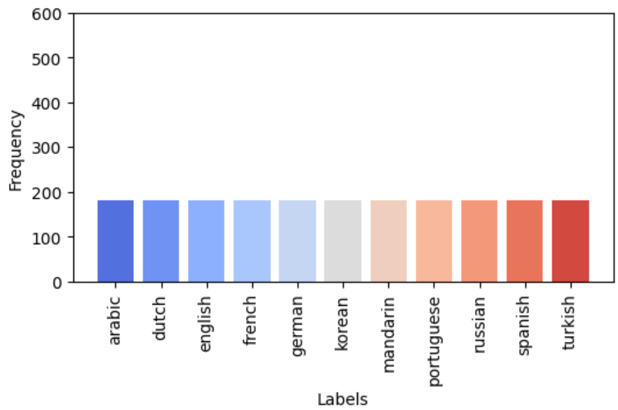
Hypothesis: As the speakers are speaking the same text, it will be easier for the model to capture the variations of pronunciations; hence, a small number of recordings is also sufficient.

Moreover, for handling this high class imbalance, the following preprocessing steps were applied:

- **Label Filtering:** Accent labels with fewer than 36 samples were removed to ensure sufficient data for each category.
- **Oversampling:** To mitigate class imbalance, the Undersampled majority class and then underrepresented accents were oversampled(with simple duplication).



(a) After applying label filtering



(b) After handling class-imbalance

Figure 9: Dataset after preprocessing

3.2 Our approaches

3.2.1 Approach-1

In our first approach, we extracted **Mel-Frequency Cepstral Coefficients (MFCCs)** from the audio files using **Librosa** and trained a **Convolutional Neural Network (CNN)** model from scratch. While training a sequence model like an **RNN** may seem more intuitive, our task is a **classification problem**.

Hypothesis: When a speaker pronounces a set of words, identifying variations in pronunciation is more crucial for accent classification than capturing temporal dependencies.

We experimented with both **13** and **40 MFCCs** per recording and observed that using more MFCCs requires either a higher number of convolutional layers or a more complex model to achieve similar accuracy.

3.2.2 Approach-2

This approach is based on **self-supervised learning**, meaning that the model is pre-trained on only unlabeled data, and **transfer learning** is required for using it for a downstream task. which is **different from semi-supervised learning** [9], where we need some amount of labeled data during the pretraining stage itself.
We utilized Facebook's **wav2vec2 base** model, which consists of approximately **110 million parameters**, with all the parameters trainable. The model has been pre-trained on around **960 hours** of **unlabeled speech data** (no labeled data is used) and requires fine-tuning (A transfer learning approach) for downstream tasks such as audio classification or speech-to-text applications [1]. The architecture is described below:

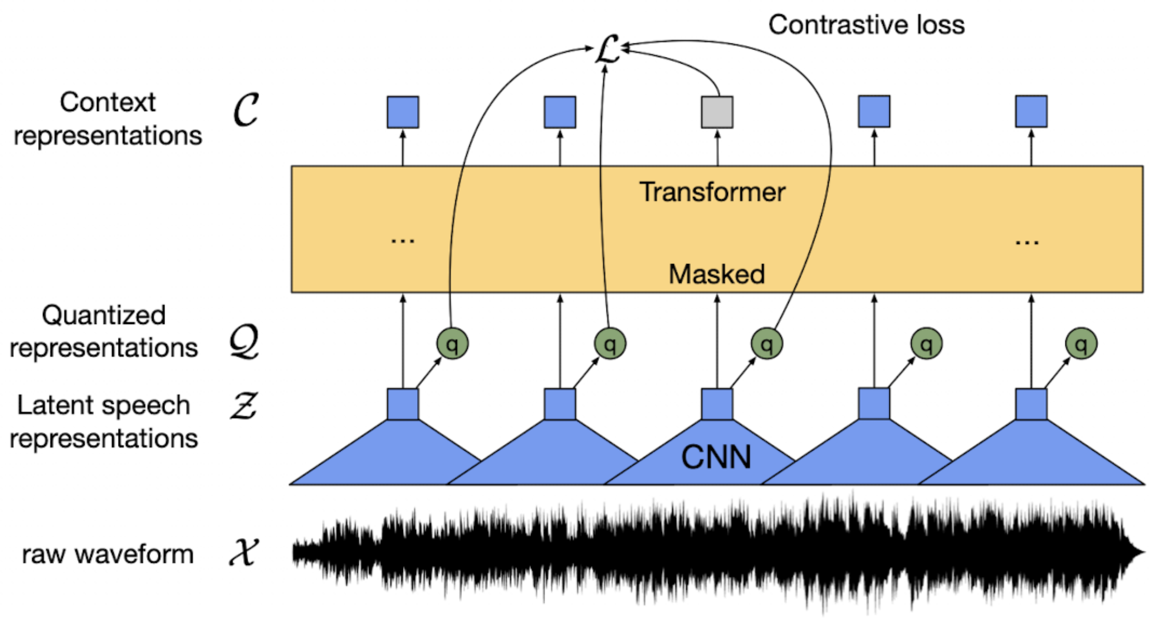


Figure 10: Architecture of wav2vec2-base [2]

[1]Pre-trained model is available at `https://huggingface.co/facebook/wav2vec2-base`

4 Experiments

This section presents the results of two experiments conducted to evaluate the performance of different models: CNN trained from scratch and wav2wec2-base's fine-tuning. Each experiment provides insights into accuracy, loss, and other metrics.

After performing pre-processing, the total number of samples reached **1980**. We trained the models with a train-test split ratio of 85/15.

The experiments were conducted in Google Colab, utilizing an NVIDIA Tesla T4 GPU.

4.1 Experiment 1: CNN trained from scratch

The experiment with 40 MFFCs per sample performed better. We tried several configurations. Details of the best one are provided below.

4.1.1 Model Summary and Configuration

Parameter	**Value**
Optimizer	Adam
Loss Function	Categorical Crossentropy
Learning Rate (lr)	0.001 (Initial)
Batch Size	32
Epochs	15
Activation Function	ReLU (Hidden), Softmax (Output)

Table 1: Training Configuration

Layer	**Output Shape**	**Params**
Conv1D (32, 3, relu)	(None, 38, 32)	128
MaxPooling1D (2)	(None, 19, 32)	0
Conv1D (64, 3, relu)	(None, 17, 64)	6,208
MaxPooling1D (2)	(None, 8, 64)	0
Flatten	(None, 512)	0
Dense (64, relu)	(None, 64)	32,832
Dense (softmax)	(None, 11)	715

Table 2: CNN Model Summary

4.1.2 Training and Testing Results

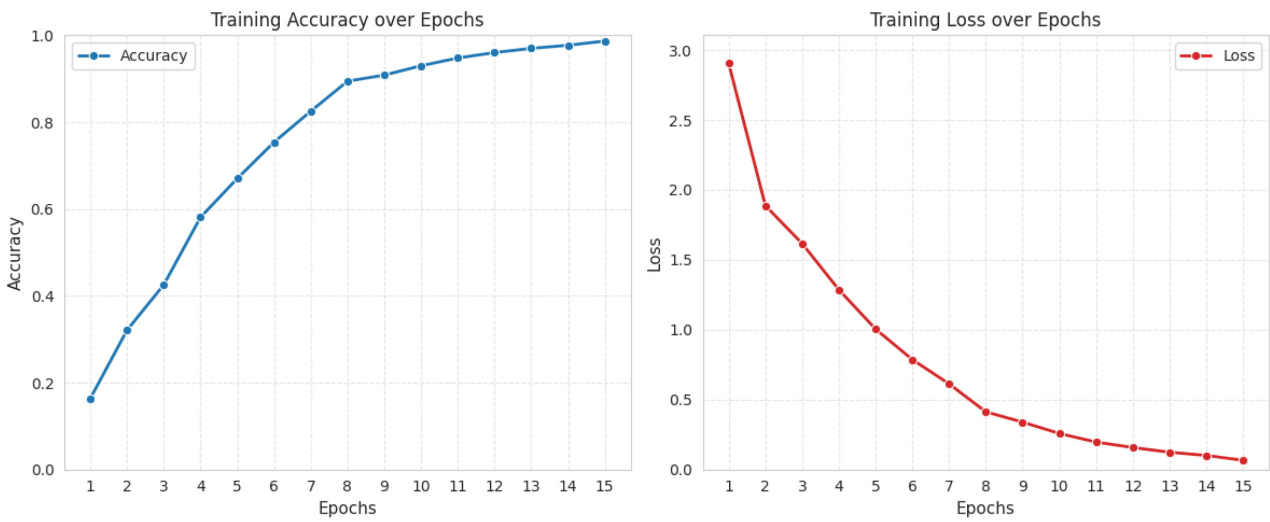


Figure 11: Training Accuracy and Loss over Epochs

Metric	Value
Accuracy	91.07%
Loss	0.2930

Table 3: Test Results for CNN-Based Model

4.1.3 Observations

Although the model performed well on the train and test sets, it failed badly on unseen data (we tested with some audio samples). Increasing the number of layers led to overfitting, while decreasing them caused significant underfitting. This led us to conclude that the model does not generalize well.

There were two possible approaches to address this issue:

1. Train the same model on a larger dataset (potentially 10k to 20k recordings).
2. Take advantage of pre-trained models through transfer learning. (The training part is resource intensive.)

We did not choose the first approach because the data preprocessing appeared highly resource-intensive. The experiment for the approach taken is described below.

4.2 Experiment 4.2: Fine-Tuning of facebook/wav2vec2-base

The paper[2] suggests freezing the feature encoder layers to prevent the weights tuned during pre-training from changing.

We experimented with three different configurations, referred to as **config_1**, **config_2**, and **config_3**, each progressively increasing the number of trainable layers:

- **Configuration 1:** All **12** transformer layers, along with the feature encoder layers, were frozen, meaning only the classification head was trained.
- **Configuration 2:** The number of frozen transformer layers was reduced to **10**, allowing partial fine-tuning of the transformer model.
- **Configuration 3:** Only the feature encoder layers were frozen, enabling **full fine-tuning of the transformer layers**.

This structured approach helped us analyze the impact of different levels of fine-tuning on model performance.

4.2.1 Model Summary and Configuration

Parameter	Configuration 1	Configuration 2	Configuration 3
Base Model	wav2vec2-base		
Optimizer	AdamW		
Learning Rate (lr)	1e-5		
Batch Size	1		
Epochs	5	6	14

Table 4: Training Configurations

Layer	Output Shape	Parameters
Feature Extractor (CNN)	(None, 1024)	95.2M (Frozen)
Transformer Encoder (12 Layers)	(None, 768)	7.06M
Fully Connected Layer	(None, 512)	393,728
Classification Head (Dense)	(None, 11)	7,979

Table 5: wav2vec2-base Model Summary

4.2.2 Training and Testing Results

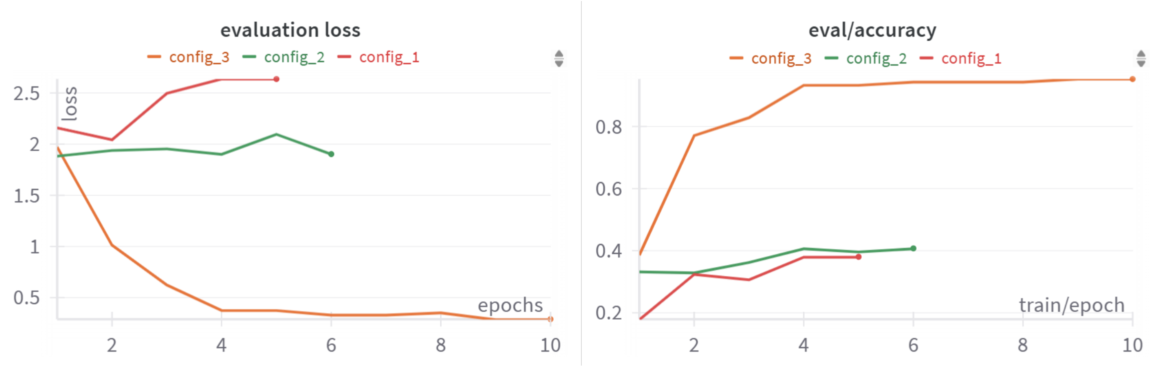


Figure 12: Evaluation Loss and Accuracy over Epochs

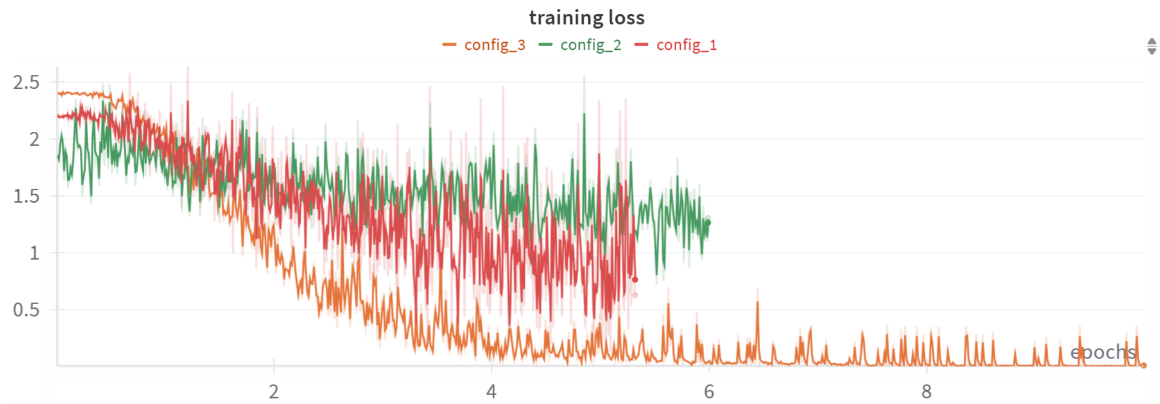


Figure 13: Training Loss over Epochs

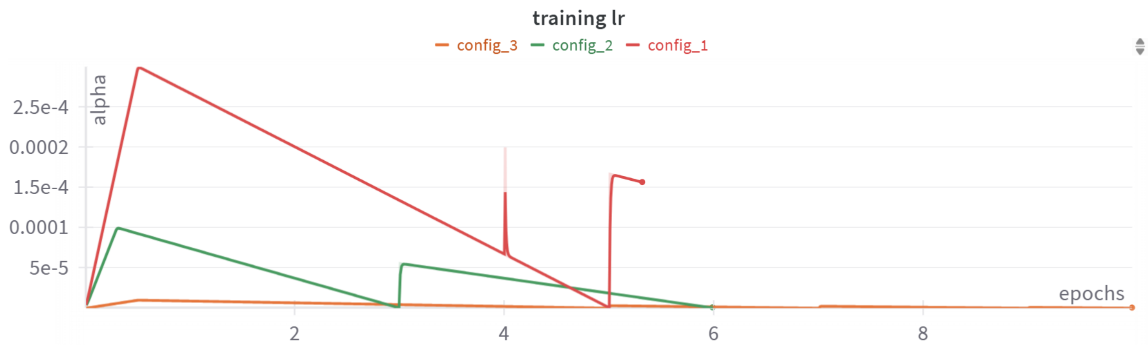


Figure 14: α over epochs

As Figure 13 depicts, the training loss fluctuates a lot. This is due to a small batch size. Due to GPU memory constraints, we kept a batch size of only **1**, but techniques like gradient accumulation can be used in this case.
Moreover, the learning rate was stable (Figure 14) for configuration 3.

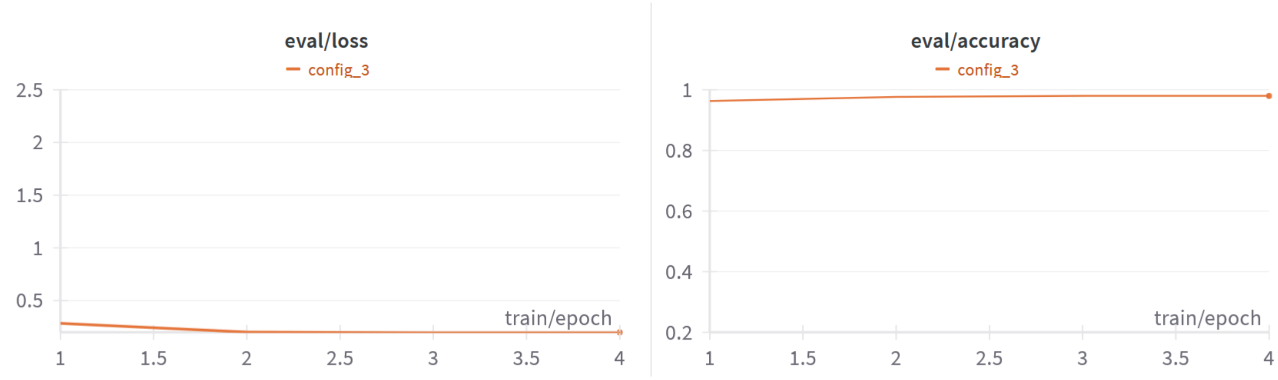


Figure 15: Loss and Accuracy for last 4 epochs of configuration 3

Loss	0.19885	**Accuracy**	97.98%

Table 6: Test Accuracy for wav2vec2-base Model

4.2.3 Observations

Configuration 1 (Only training classification head) caused a very slow increase in accuracy. We concluded that the model id underfitting. Very little improvement was observed in **Configuration 2** (2 unfreezed transformer layers). Then, we decided to train all 12 transformer layers along with the classification head in **third configuration**, and it worked well. It gave a smooth learning curve and achieved around 98% testing accuracy, generalized well. The fine-tuned model is available for inference at HuggingFace[2].

[2]Fine-tuned model is available at `https://huggingface.co/vrund1346/wav2vec2_accent_classification_v2`

5 User Interface and Architecture

5.1 Design Pattern

The system follows a modular design combining microservices and MVC.

5.1.1 Microservices Architecture

The system consists of:

- **React (Frontend):** Handles UI and API interactions.
- **Spring Boot (Backend):** Manages authentication, business logic, and database transactions.
- **Flask API:** Executes the deep learning model and returns predictions.

5.1.2 Model-View-Controller (MVC) in Spring Boot

Spring Boot follows the MVC structure:

- **Model:** Represents application data.
- **View:** React UI for data display and interaction.
- **Controller:** Handles requests, processes data, and manages responses via:
  - **Service:** Processes data and logic.
  - **Repository:** Manages database operations.

5.1.3 RESTful API Communication

System components interact via REST APIs:

- React requests predictions from Flask.
- React handles authentication and data via Spring Boot.
- Responses update the UI.

5.2 System Architecture Flow

1. **User Interaction:** React UI sends requests.
2. **API Processing:** Flask handles model execution; Spring Boot manages business logic and storage.
3. **Response Handling:** Results update the UI.

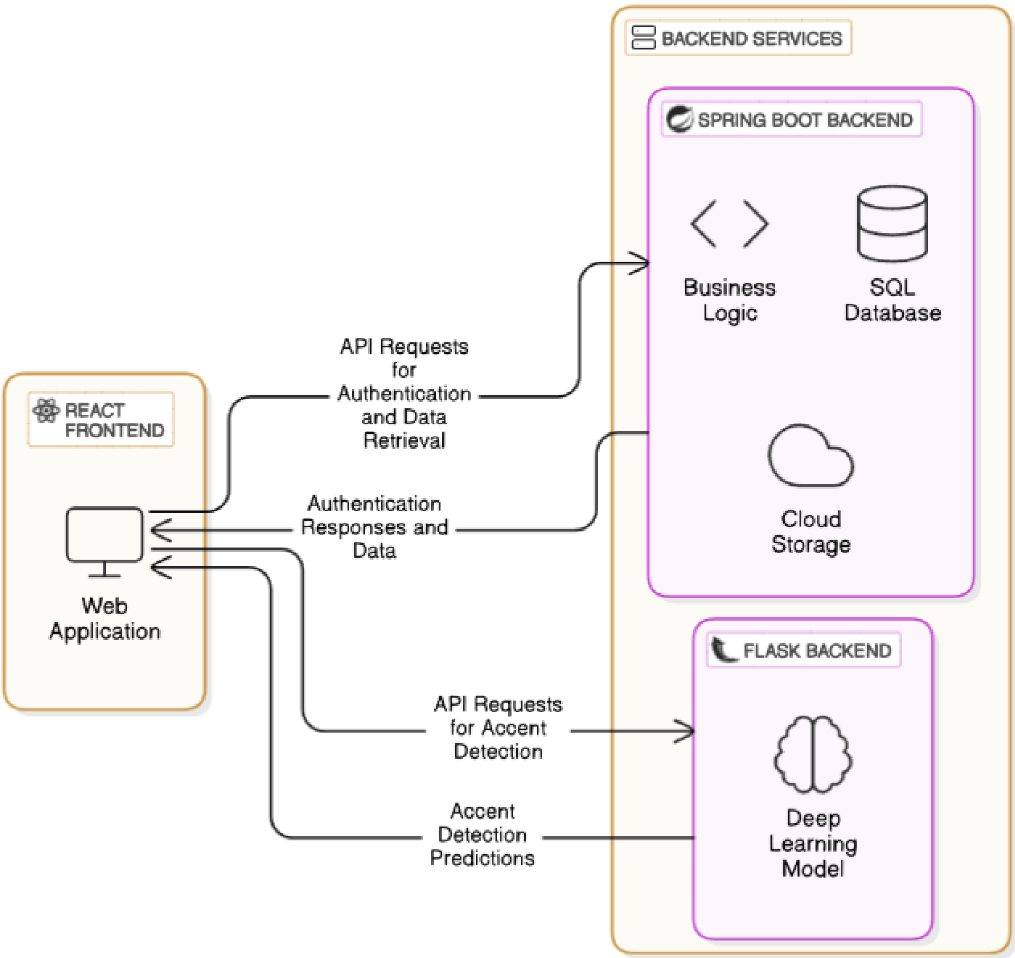


Figure 16: **System Architecture Diagram**

6 Implementation Details

This section describes how the system is implemented, covering model deployment, backend development, frontend integration, and overall system architecture.

6.1 Frontend Development

The user interface is built with **React** and integrates with the backend.

- **Tech Stack**: React with the MUI frontend framework used.
- **UI Design**: Key components and user interactions.
- **API Integration**: Handling responses from the backend and state management.
- **Dependencies**:
  - **React** - A JavaScript library for building dynamic and responsive user interfaces.
  - **React Router** - A library for handling navigation and routing in React applications.
  - **MUI** - A React UI framework providing pre-styled components for a modern design.
  - **Wavesurfer.js** - A customizable audio waveform visualization library for React applications.
  - **React Simple Maps** - A library for creating interactive and customizable maps in React.
  - **d3-geo** - A library for geographical computations, used with maps.

6.2 Backend Development

The backend is developed using **SpringBoot** and provides APIs for frontend communication.

- **Tech Stack**: SpringBoot & Flask.
- **Database**: Store and retrieve data from **Mysql Database** (Deployed on **Aiven**-a cloud provider).
- **API Development**: REST APIs.
- **Authentication & Authorization**: JWT authentication with role-based authorization.
- **Cloud Integration**: The system leverages **Microsoft Azure** for scalable cloud storage and computing resources, ensuring high availability and secure data management.
- **Dependencies:**
  - **SpringBoot** - A Java-based framework for backend development.
  - **Flask** - A lightweight Python framework used for serving the machine learning model.
  - **Spring Security** - Provides authentication and authorization mechanisms.
  - **JWT (JSON Web Tokens)** - For secure token-based authentication mechanism used for user verification and authorization.
  - **Spring Data JPA** - Enables interaction with the MySQL database.

- **MySQL** - A relational database management system, deployed on Aiven.
- **Jakarta Mail** - Java mail API for handling email communication.
- **Microsoft Azure Storage SDK** - Enables seamless integration with Azure cloud storage.

6.3 Screenshots of Web Interface

This section presents the key views of the implemented system, categorized into authentication, dashboard, user interactions, and result analysis.

6.3.1 Authentication Screens

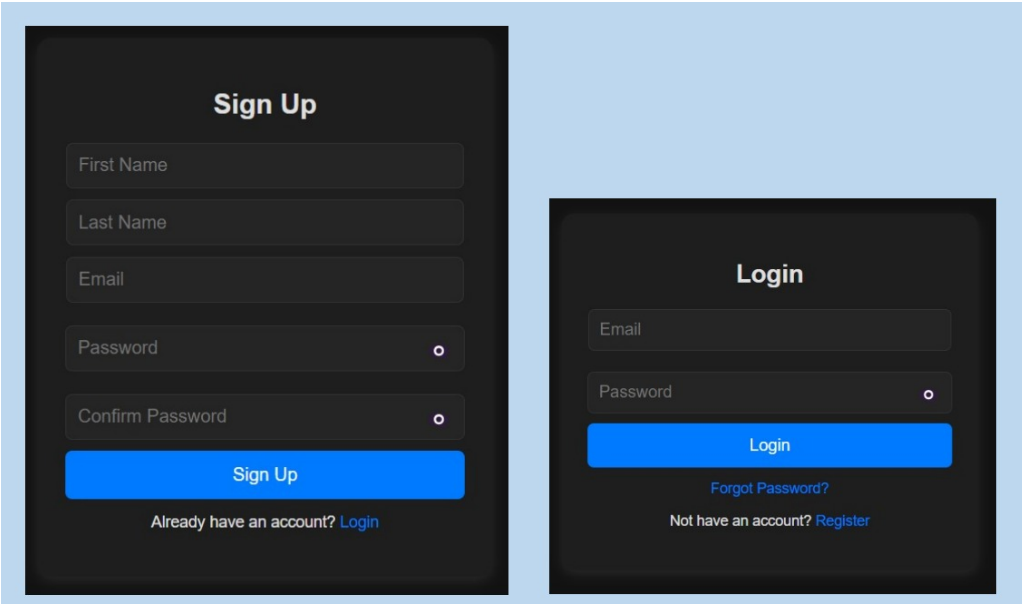


Figure 17: Login & Register View

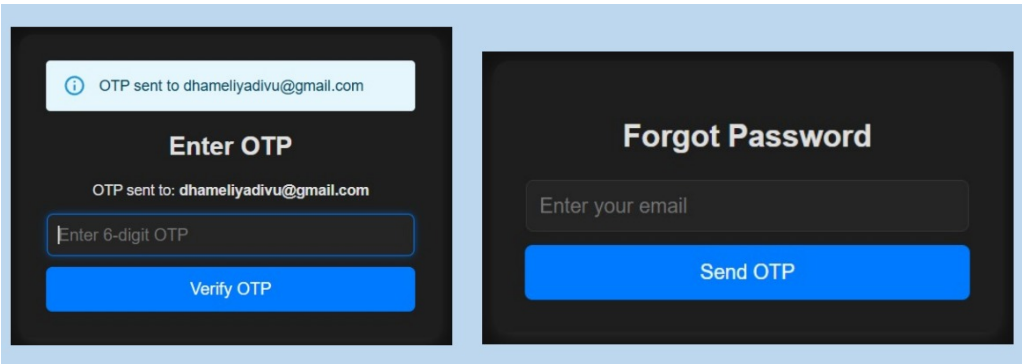


Figure 18: Forgot-Password View

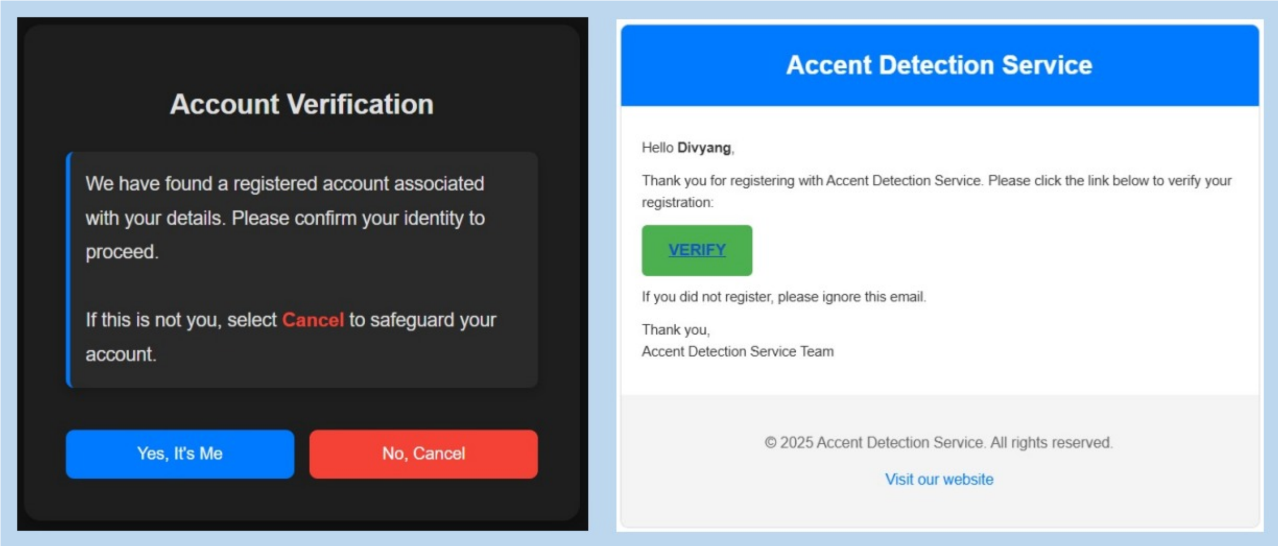


Figure 19: Account Verification View

6.3.2 Dashboard Views

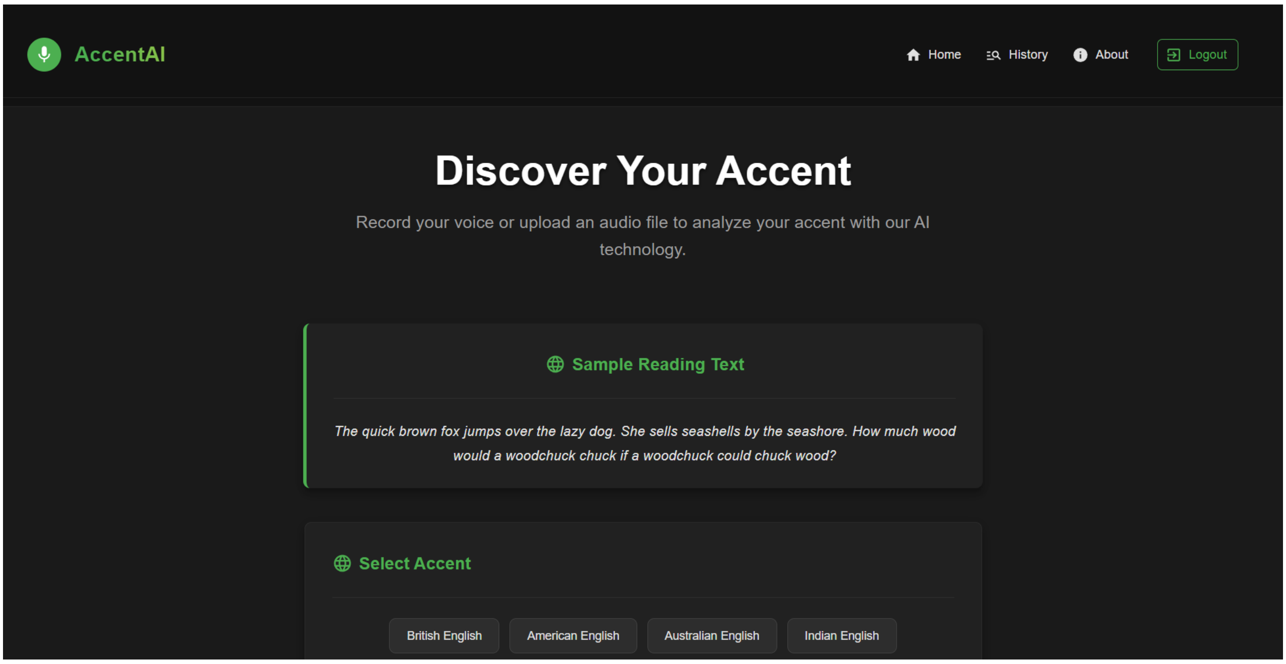


Figure 20: Dashboard View - Overview

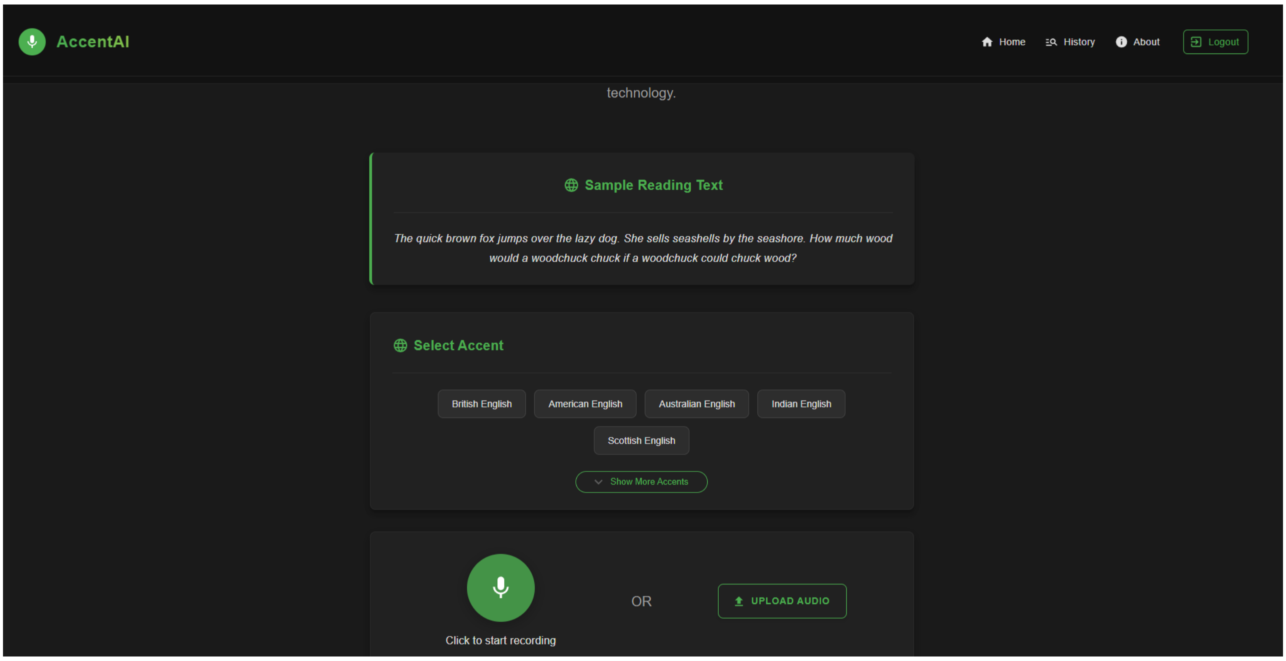


Figure 21: Dashboard View - Detailed Analysis

6.3.3 User Input and Analysis

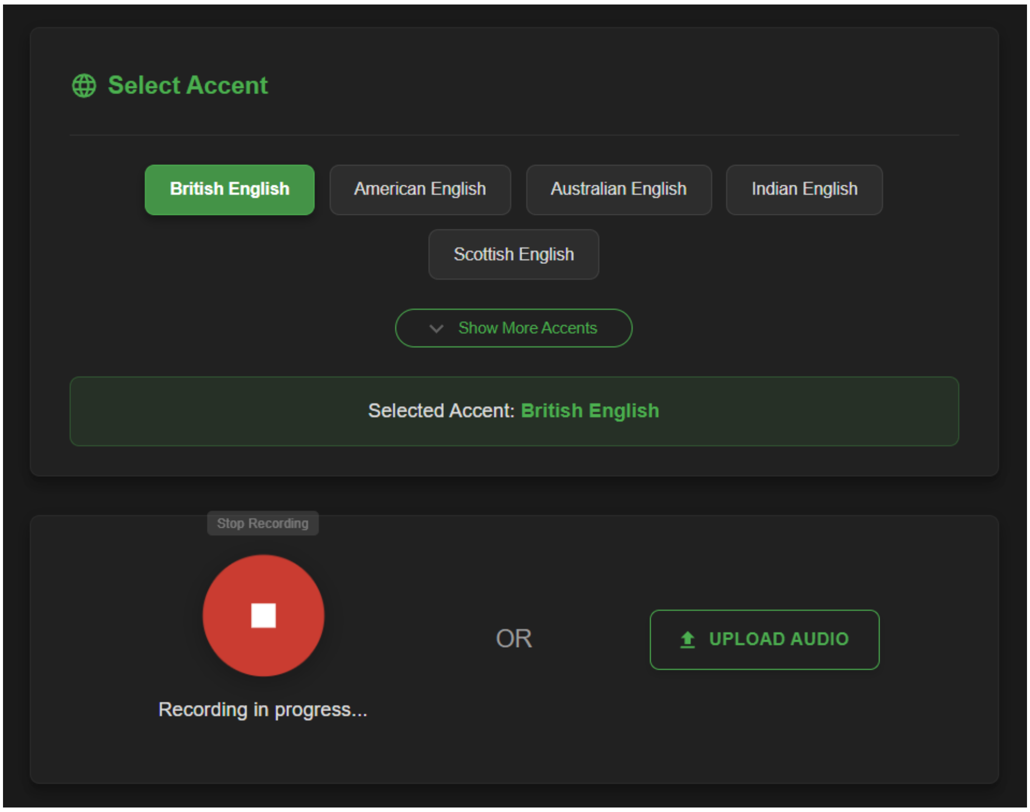


Figure 22: User Input Interface

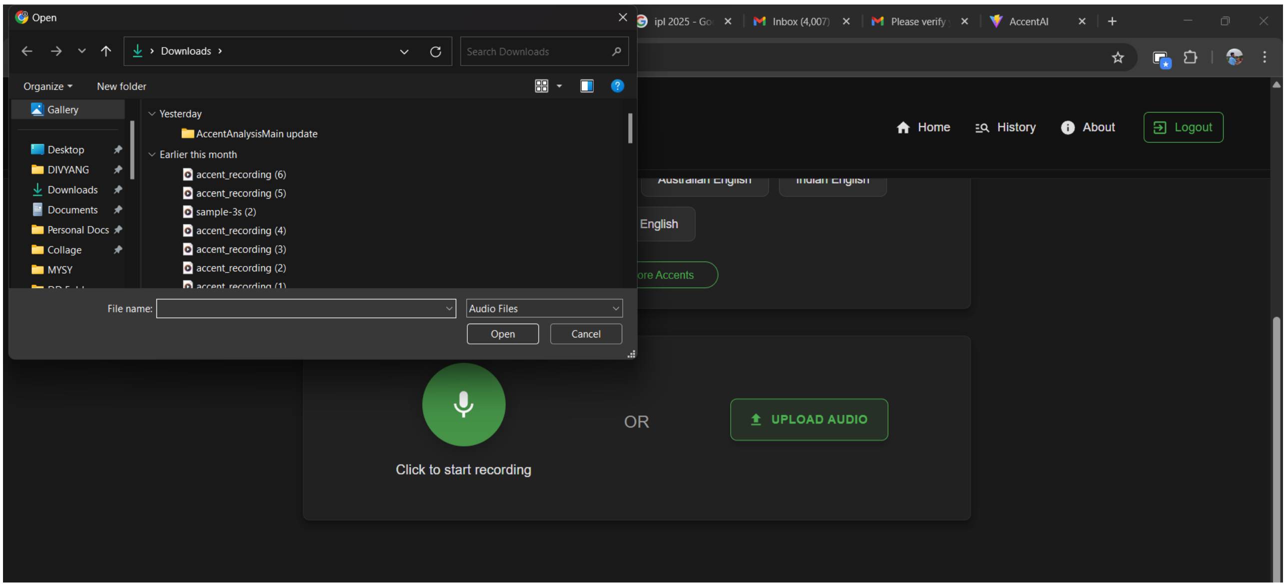


Figure 23: User Input with File Upload

6.3.4 Accent Analysis and Results

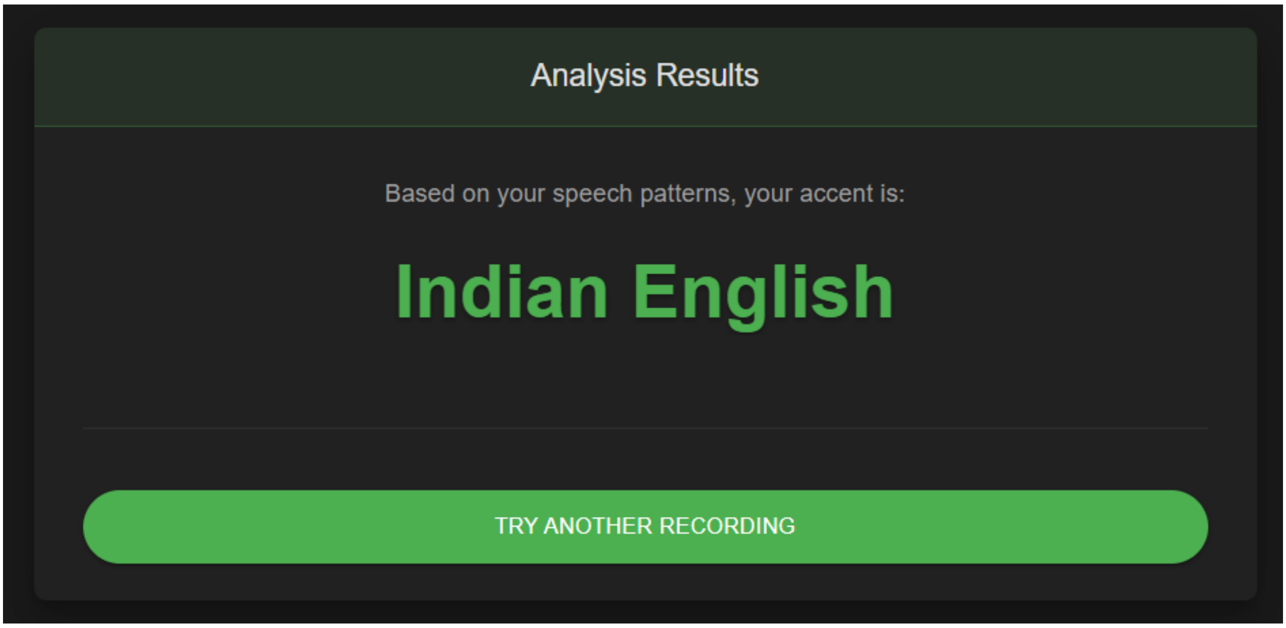


Figure 24: Model Prediction Results

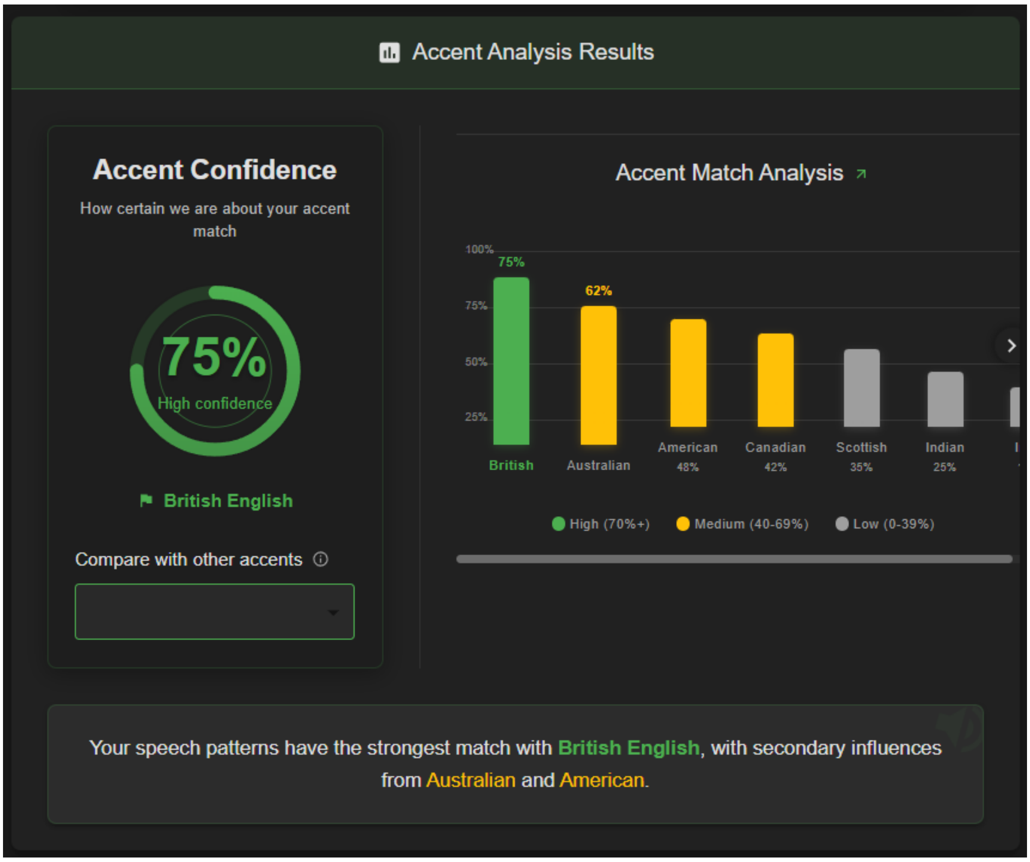


Figure 25: Accent Analysis Dashboard

6.3.5 Additional Views

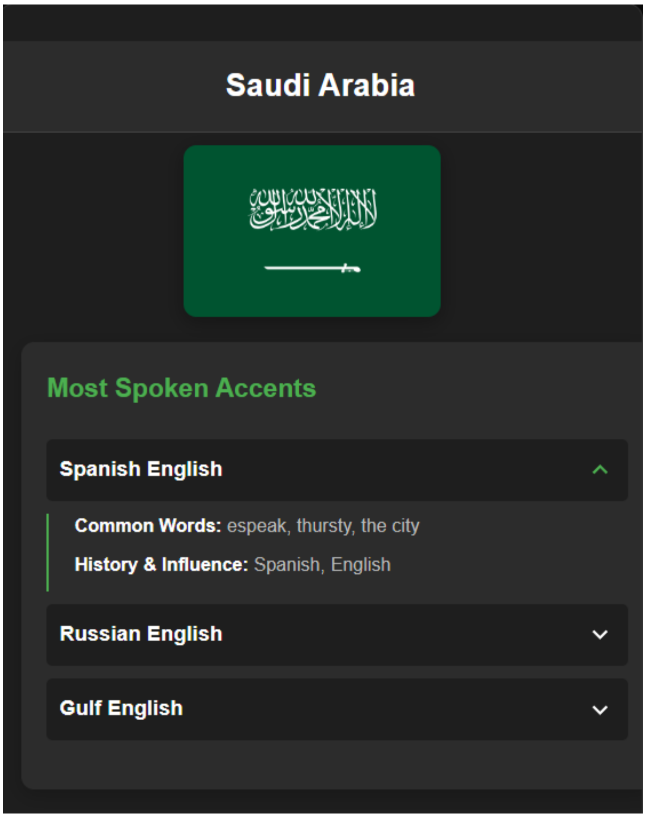


Figure 26: Country Details View

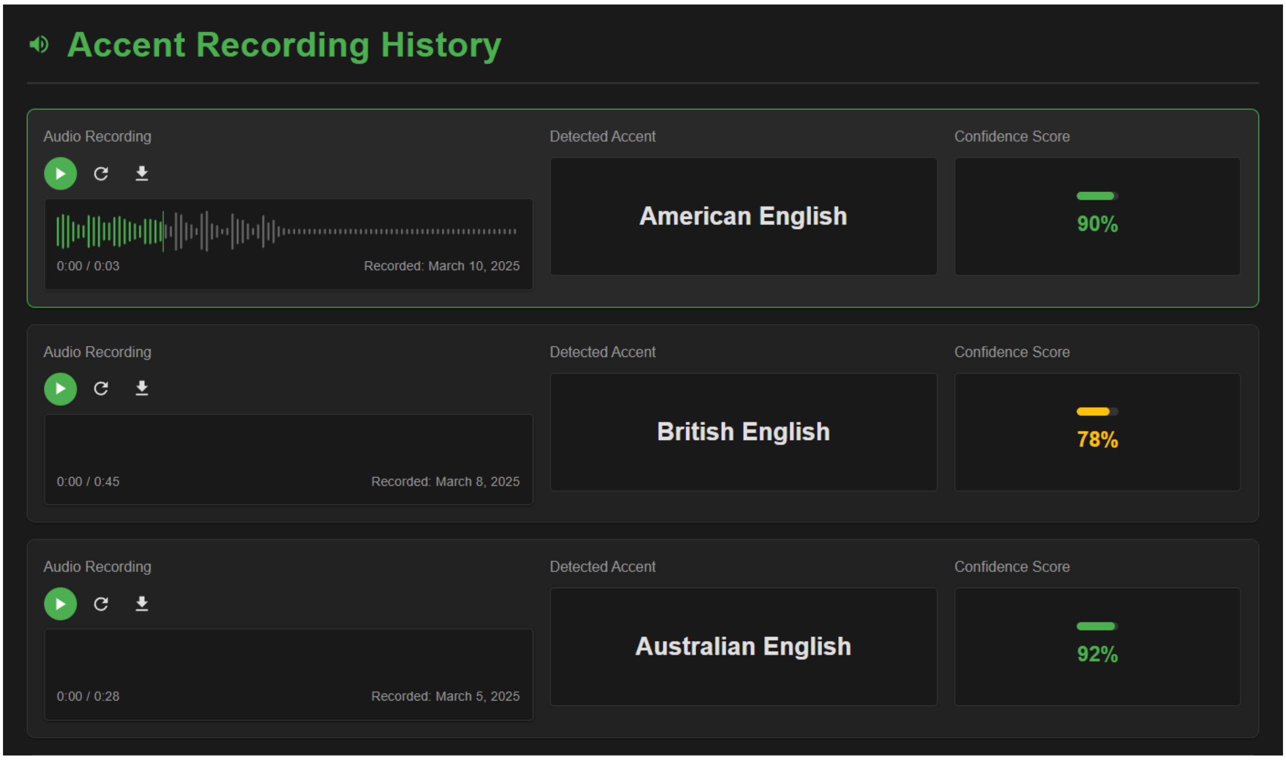


Figure 27: User History Page

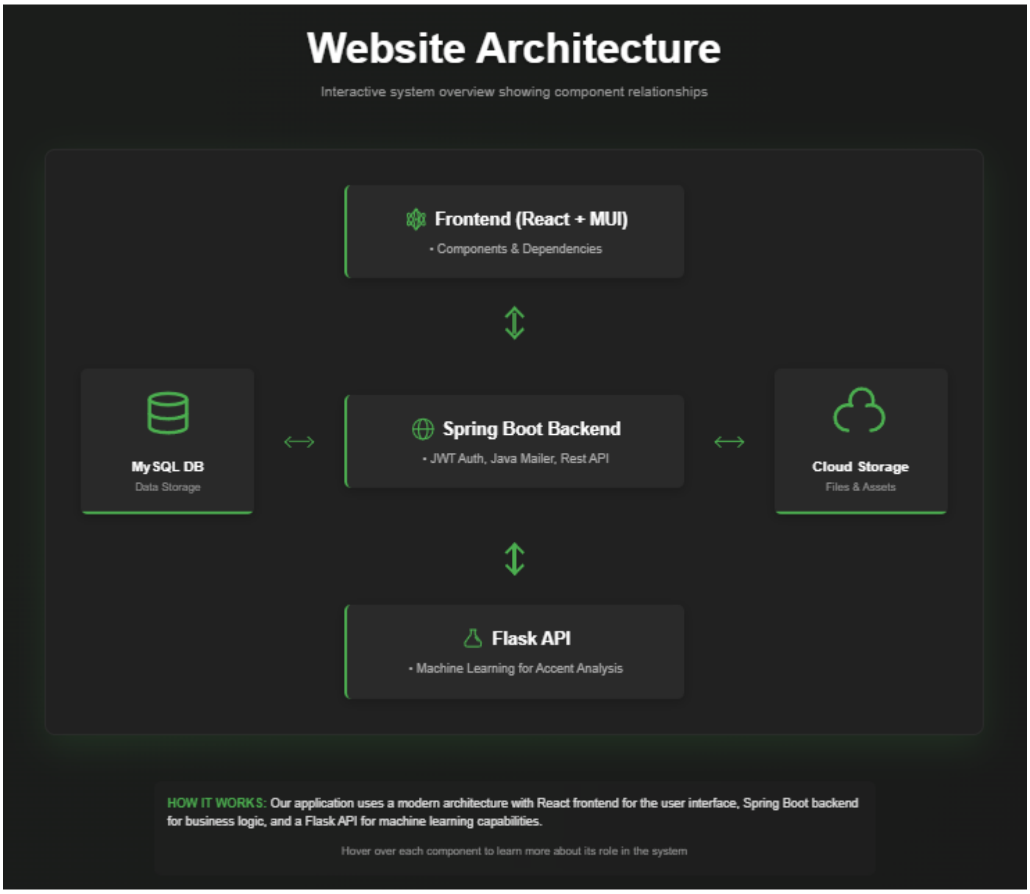


Figure 28: About Page

7 Results & Evaluation

7.1 Performance on unseen data (For best model)

The model correctly predicted accents for **9 out of 22** audio recordings.

Audio File	**Speaker Gender**	**Correct Accent**	**Predicted Accent**
Arabic(M)audio.mp3	Male	Arabic	German
Arabic(F)audio.mp3	Female	Arabic	Turkish
Dutch(M)audio.mp3	Male	Dutch	**Dutch**
Dutch(F)audio.mp3	Female	Dutch	**Dutch**
English(M)audio.mp3	Male	English	**English**
English(F)audio.mp3	Female	English	**English**
French(M)audio.mp3	Male	French	**French**
French(F)audio.mp3	Female	French	German
German(M)audio.mp3	Male	German	**German**
German(F)audio.mp3	Female	German	**German**
Korean(M)audio.mp3	Male	Korean	Turkish
Korean(F)audio.mp3	Female	Korean	Mandarin
Mandarin(M)audio.mp3	Male	Mandarin	Spanish
Mandarin(F)audio.mp3	Female	Mandarin	Korean
Portuguese(M)audio.mp3	Male	Portuguese	German
Portuguese(F)audio.mp3	Female	Portuguese	Mandarin
Russian(M)audio.mp3	Male	Russian	Spanish
Russian(F)audio.mp3	Female	Russian	Mandarin
Spanish(M)audio.mp3	Male	Spanish	**Spanish**
Spanish(F)audio.mp3	Female	Spanish	Turkish
Turkish(M)audio.mp3	Male	Turkish	German
Turkish(F)audio.mp3	Female	Turkish	**Turkish**

Table 7: Performance on Unseen Data for the Best Model

7.2 UI Test Cases

Test ID	Test Case Description	Action Performed	Expected Outcome	Actual Outcome	Pass/Fail
TC01	User registration with valid details	firstname:"Rushang", lastname:"Patel", email:"rushang@mail.com", password:"Pass@123", confirm password: "Pass@123", click submit	Verification email sent to email:"rushang@mail.com"	Email received successfully	Pass
TC02	User registration with invalid email	firstname:"Divayag", lastname:"Shah", email:"123user@mail.com", password:"Pass@123", click submit	System shows error:"Invalid email format"	Error displayed correctly	Pass
TC03	Email verification after registration	click:"Verification link" received in email:"rushang@mail.com"	Account status:"Verified"	Account verified successfully	Pass
TC04	Login with correct credentials	email:"rushang@mail.com", password:"Pass@123", click:"Login"	Redirected to page:"Home"	Redirected to home page	Pass
TC05	Forgot password functionality	click:"Forgot Password", email:"rushang@mail.com"	Email status:"Password reset email sent"	Email received successfully	Pass
TC06	Upload invalid file format	uploadfile:"divayagaudio.txt", click:"Detect"	System output:"Invalid file format"	Error displayed correctly	Pass
TC7	View audio history	navigate:"History Page"	history shows:"vrund speech.mp3" with detected accent	History displayed correctly	Pass

Table 8: Test Cases for Accent Detection System

8 Conclusion

This study successfully developed an English accent detection system utilizing deep learning models, achieving promising results. The system demonstrates a high ability to distinguish between various English accents. The model's performance was validated using accuracy.

8.1 Future Work

To enhance the performance and applicability of the proposed system, the following improvements are recommended:

- **Dataset Expansion:** Incorporating a more diverse range of accents and dialects to improve model robustness and minimize bias.
  - **African (Nigerian) Accent Speech Data (AASD):** A dataset of speech recordings from various Nigerian regions. [5].
  - **UK and Ireland Dialect Dataset:** A dataset featuring speech recordings from various UK and Irish dialects, including Irish English, Scottish English, and Welsh English [10].
  - **AccentDB:** A structured and labeled dataset containing speech samples from four non-native Indian English accents (8 speakers) and four native English accents (13 speakers from 4 countries). It also includes a metropolitan Indian accent (2 speakers), making it valuable for accent-based speech analysis [11].
- **Model Optimization:** Employing advanced techniques such as hyperparameter tuning and regularization strategies to refine accuracy and stability.
- **Web Application Enhancements:** Improving the frontend with React-based interactive visualizations and real-time feedback on accent detection results.

We are working on our third experiment to extend the classification for 15 accents with these datasets.

9 References

References

[1] Rachael Tatman. Speech Accent Archive. *Kaggle*. Available: https://www.kaggle.com/datasets/rtatman/speech-accent-archive

[2] Alexei Baevski et al. Wav2Vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *arXiv*, 2020. Available: https://arxiv.org/abs/2006.11477

[3] Audio signal processing for machine learning. *YouTube Playlist*. Available: https://youtube.com/playlist?list=PL-wATfeyAMNqIee7cH3q1bh4QJFAaeNv0&feature=shared

[4] Vrund Dobariya. Audio Preprocessing and Feature Extraction (Part 1). *Medium*. Available: https://medium.com/@vrund3626/audio-preprocessing-and-feature-extraction-part-1-72575035609a

[5] AI Olapo. African (Nigerian) Accent Speech Data AASD. *Kaggle*. Available: https://www.kaggle.com/datasets/aiolapo/africannigerian-accent-speech-data-aasd

[6] Vrund Dobariya. Audio Preprocessing and Feature Extraction (Part 2). *Medium*. Available: https://medium.com/@vrund3626/audio-preprocessing-and-feature-extraction-part-2-f79b0f769c10

[7] Wikipedia contributors. *Spectral Leakage*. Wikipedia, The Free Encyclopedia. Available at: https://en.wikipedia.org/wiki/Spectral_leakage

[8] Vrund Dobariya. Audio Preprocessing and Feature Extraction (Part 3). *Medium*. Available: https://medium.com/@vrund3626/audio-preprocessing-and-feature-extraction-part-3-3ccca4c34d14

[9] IBM. *Self-Supervised Learning*. Available at: https://www.ibm.com/think/topics/self-supervised-learning.

[10] Hema Bhushan. UK and Ireland English Dialect Speech. *Kaggle*. Available: https://www.kaggle.com/datasets/hemabhushan/uk-and-ireland-english-dialect-speech

[11] Sparsh Ims. AccentDB - Core & Extended. *Kaggle*. Available: https://www.kaggle.com/datasets/imsparsh/accentdb-core-extended

[12] *Frequency and Loudness Perception*. In: Sound and Audio. Pressbooks. Available at: https://pressbooks.pub/sound/chapter/frequency-and-loudness-perception/.

[13] *React Component Library - MUI* https://mui.com/material-ui/all-components/

[14] *NPM react-simple-maps* https://www.npmjs.com/package/react-simple-maps

[15] *Wavesurfer.js* https://wavesurfer.xyz/