**AI-Powered Public Speaking Training Platform**

**Context:**

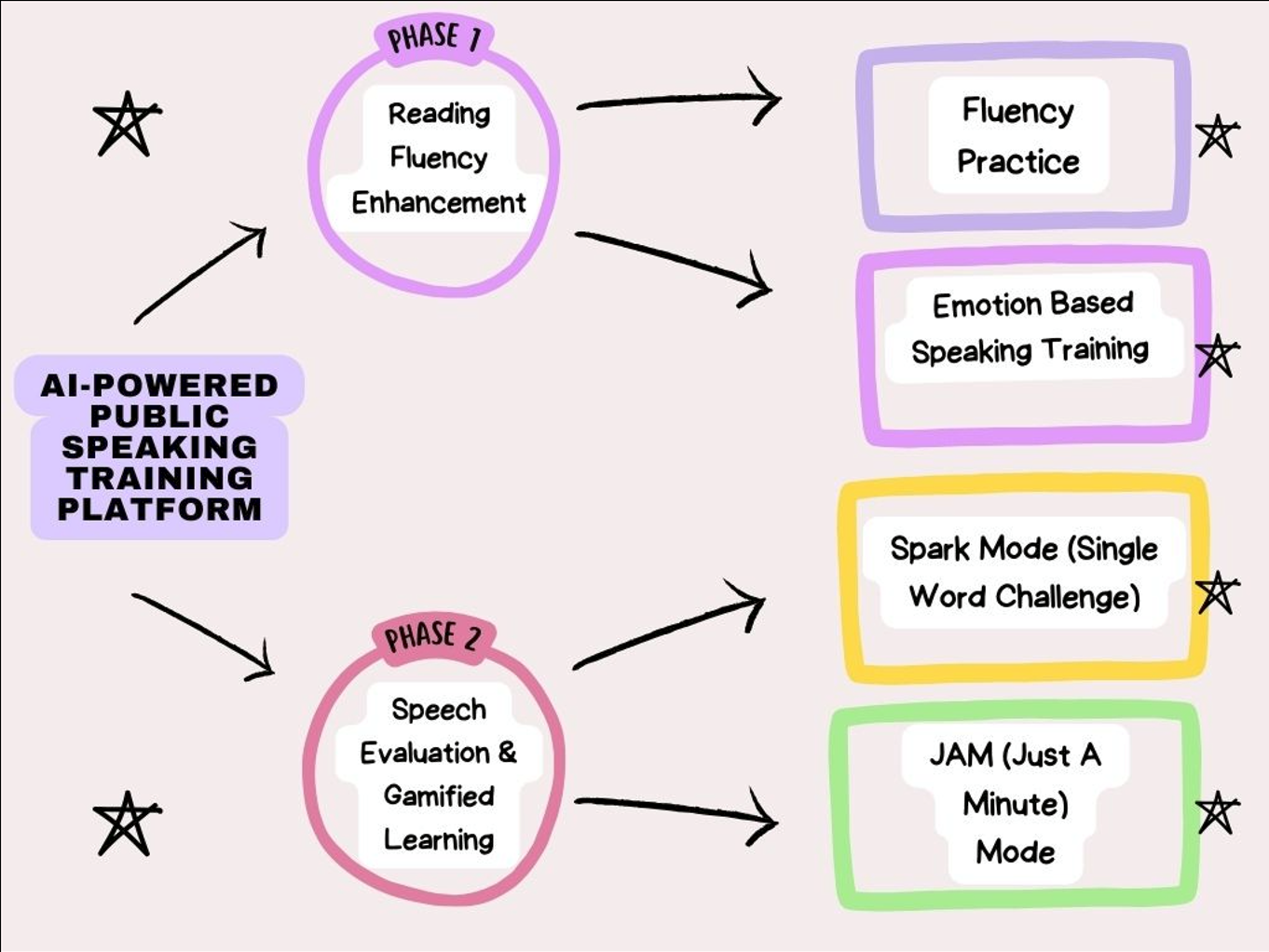
Public speaking is a crucial skill for professional growth. Your challenge is to build an interactive, AI-powered training platform that helps users enhance their speaking abilities through gamified, real-time feedback-driven exercises.

**Project Overview:**

The Speech Training & Gamified Learning App is a Streamlit-based web application designed to enhance speech fluency and expression through interactive exercises. The app is structured into multiple pages, each focusing on different aspects of speech training. It also includes external resources to support hearing-impaired individuals by converting speech to sign language.

**Technologies Used:**

* Streamlit: For building the web-based interactive interface.
* Streamlit Pages API: For modular navigation between different training exercises.
* Subprocess: (Mentioned in the request but not in the provided code; can be used for executing external scripts if needed.)
* Web Integration: External modules linked via sidebar navigation.



**Multi-Page Navigation with Streamlit Pages:**

The application consists of four interactive speech training modules:

* Fluency Practice (🗣️) - Helps users practice smooth and continuous speech.
* Emotion-Based Speaking (🎭) - Enhances speech delivery by incorporating emotional expressions.
* Single Word Spark (💡) - Focuses on articulation and pronunciation of individual words.
* Sentence Speech Challenge (📝) - Encourages structured sentence formation and verbal delivery.

**Source Code:**

**Streamlit Pages:**

# Define pages

fluency\_practice = st.Page("pages/fluency\_practice.py", title="Fluency Practice", icon="🗣️")

emotion\_based\_speaking = st.Page("pages/emotion\_based\_speaking.py", title="Emotion-Based Speaking", icon="🎭")

single\_word\_spark = st.Page("pages/single\_word\_spark.py", title="Single Word Spark", icon="💡")

sentence\_speech\_challenge = st.Page("pages/sentence\_speech\_challenge.py", title="Sentence Speech Challenge", icon="📝")

# Configure navigation

pg = st.navigation([

fluency\_practice,

emotion\_based\_speaking,

single\_word\_spark,

sentence\_speech\_challenge ])

**Model Evaluation Streamlit project follows this structure:**

Model Evaluation Streamlit/

│── pages/

│ ├── emotion\_based\_speaking.py

│ ├── fluency\_practice.py

│ ├── sentence\_speech\_challenge.py

│ ├── single\_word\_spark.py

│── app.py

│── requirements.txt

1. **Fluency Practice (🗣️):**

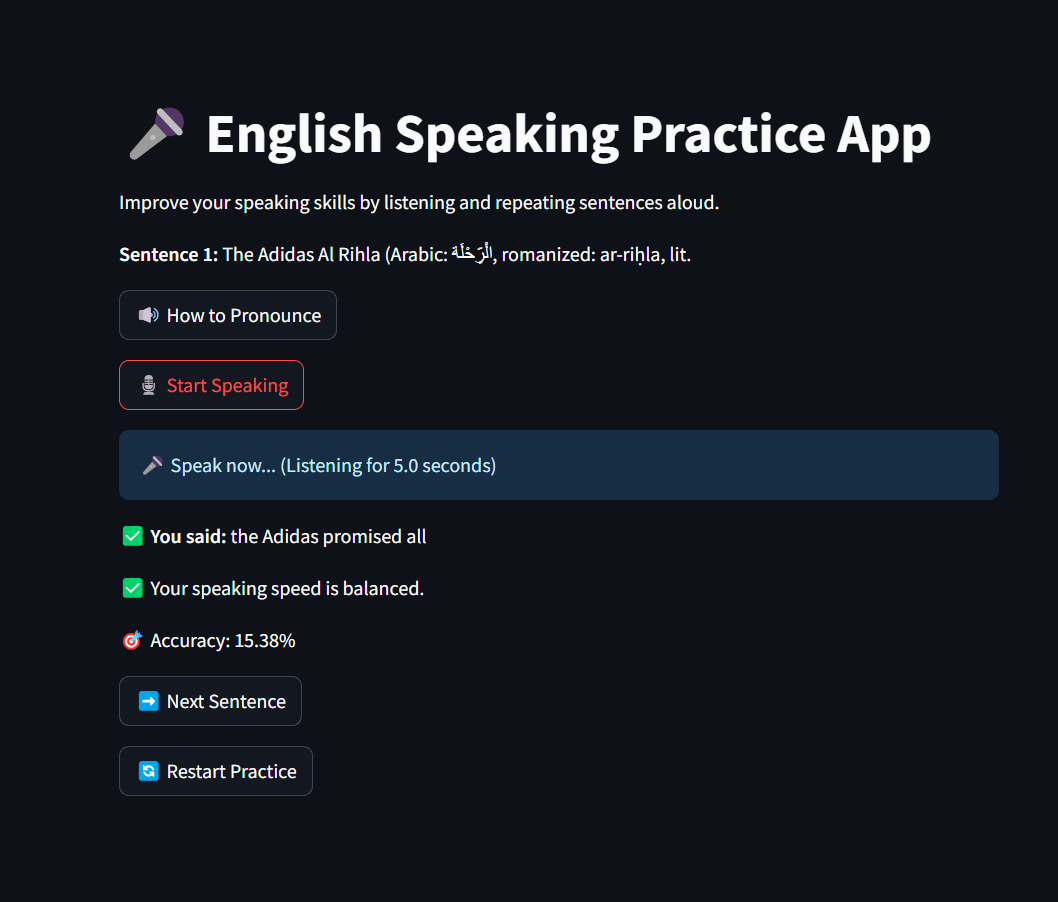
The English Speaking Practice App is a Streamlit-based web application designed to help users improve their English pronunciation, fluency, and accuracy through interactive speech recognition and feedback. The app fetches random sentences from Wikipedia, allows users to listen to the correct pronunciation, and provides real-time feedback on spoken sentences.

**Features:**

* 🎤 **Speech Recognition**: Converts user speech to text and provides feedback.
* 🔊 **Text-to-Speech (TTS)**: Pronounces sentences for accurate learning.
* 📖 **Wikipedia Sentence Fetching**: Generates random sentences for practice.
* 📈 **Accuracy & Speed Analysis**: Compares spoken words with original text and gives performance metrics.
* ⏳ **Time Calculation**: Sets expected speaking time based on sentence length.
* 🔄 **Restart & Next Sentence Options**: Allows structured practice sessions.

**User Flow:**

1. **User selects the number of sentences to practice**.
2. **Random sentences are fetched from Wikipedia**.
3. **User listens to correct pronunciation**.
4. **User speaks the sentence aloud**.
5. **App analyzes accuracy and speed**:
   * Checks pronunciation accuracy.
   * Measures speaking speed.
   * Provides feedback on fluency.
6. **User can proceed to the next sentence or restart practice**.



1. **Emotion-Based Speaking (🎭):**

The **Speech Training & Emotion Recognition App** is an interactive tool designed to help users improve their spoken English skills by practicing pronunciation, emotion expression, and timing while reading dynamically generated sentences. The app provides feedback on speech clarity and speed, and recognizes user emotions based on sentiment analysis.

**Features:**

* **Dynamic Sentence Generation**: Uses words from the NLTK Gutenberg corpus to generate practice sentences.
* **Emotion Detection**: Analyzes the sentiment of sentences and assigns an emotion (Happy, Motivated, Neutral, Sad, or Angry).
* **Speech Recognition**: Captures and transcribes user speech using the Google Speech Recognition API.
* **Speaking Time Analysis**: Calculates the optimal time needed for speaking and provides feedback on speed.
* **Text-to-Speech (TTS) Pronunciation**: Converts text to speech using Google Text-to-Speech (gTTS) for pronunciation assistance

**Application Flow**:

1. **User Inputs:**

* Enter the number of sentences for practice.
* Click **Start Practice Session**.

1. **Sentence Generation:**

* The app generates random sentences and displays the emotion associated with each.
* The allowed speaking time is calculated based on word count.

1. **Speech Pronunciation Assistance:**

* Users can click **Pronounce Sentence** to hear the sentence read aloud.

1. **Speech Recognition:**

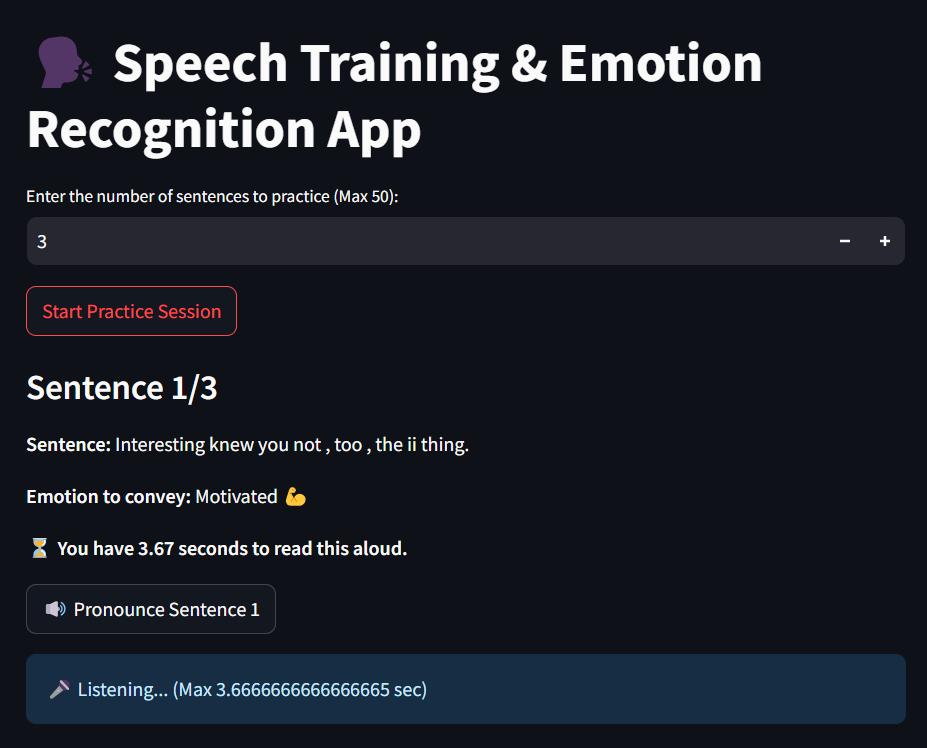
* The app listens to user speech and transcribes it.
* It records actual speaking time.

1. **Feedback:**

* Provides feedback if the user speaks too fast or too slow.

**Analyzes sentence sentiment and categorizes emotions:**

* Happy 😊
* Motivated 💪
* Neutral 😐
* Sad 😔
* Angry 😠



1. **Single Word Spark (💡):**

The **Speech Training & Emotion Recognition App** is an interactive quiz-based application designed to help users enhance their vocabulary and pronunciation by speaking synonyms aloud. The app leverages **speech recognition**, **natural language processing (NLP)**, and **semantic similarity** using **Sentence-BERT** to evaluate users' spoken responses.

**Features:**

* 🎙️ **Voice-based quiz interaction**
* 📚 **Randomized synonym-based questions** using the Datamuse API
* 🤖 **Speech recognition and automatic assessment** using Google Speech Recognition API
* 📊 **Sentence-BERT similarity scoring** to compare user responses with correct answers
* 🔄 **Automated quiz progression** for a smooth experience
* 🏆 **Real-time scoring system** based on similarity relevance

**Application Flow:**

1. **User starts the quiz**: Selects the number of questions.
2. **Question Generation**: Random words and their synonyms are fetched from the Datamuse API.
3. **Speech Input**: The user speaks their answer into the microphone.
4. **Speech Recognition & Processing**: The system captures the response and converts it into text.
5. **Answer Evaluation**:
   * The response is compared with the correct answer using **Sentence-BERT similarity scoring**.
   * A similarity score is computed and translated into points.
6. **Score Calculation**:
   * Based on the similarity percentage, points are awarded.
7. **Quiz Completion & Review**:
   * The final score is displayed along with a detailed review of the user's responses.

**Source Code:**

Fetch Synonym-Based Questions

def get\_datamuse\_synonym\_question():

base\_url = "https://api.datamuse.com/words"

letter = random.choice(string.ascii\_lowercase)

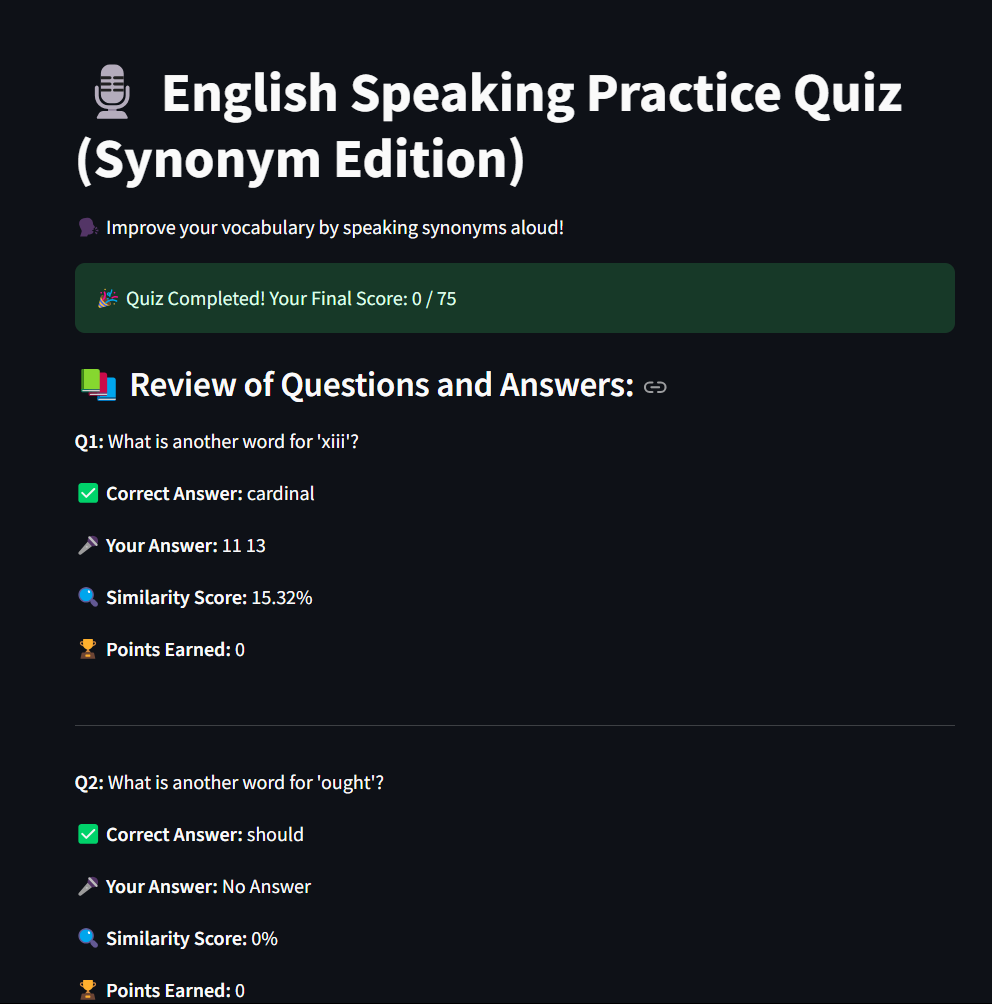
resp = requests.get(base\_url, params={"sp": f"{letter}\*", "max": 50})

word = random.choice(resp.json())["word"]

syn\_resp = requests.get(base\_url, params={"rel\_syn": word, "max": 10})

synonym = random.choice(syn\_resp.json())["word"]

return {"question": f"What is another word for '{word}'?", "answer": synonym.lower()}



1. **Sentence Speech Challenge (📝):**

This application helps users improve their English-speaking skills by providing real-time speech recognition, AI-driven feedback, and Wikipedia-based content for practice. The app analyzes spoken words, compares them with reference content, and provides suggestions for improvement.

**Features:**

* **Speech Recognition**: Converts spoken words into text using Google Speech Recognition.
* **Wikipedia Content Extraction**: Fetches a summary of any topic from Wikipedia to serve as a reference paragraph.
* **Keyword Analysis**: Extracts and compares key terms from both the reference text and the user’s speech.
* **AI-Based Feedback**: Uses NLP techniques to provide structured feedback, highlighting missing or extra keywords.
* **Speech Similarity Scoring**: Computes a similarity score between the user’s speech and the reference text using Sentence-BERT embeddings.
* **Speech Enhancement**: Suggests improvements for better speech delivery.
* **Custom Speaking Duration**: Allows users to set a time limit for their speech practice.

**Core Components:**

### 1. fetch\_wikipedia\_summary(topic)

Fetches a short summary from Wikipedia for the given topic.

### 2. extract\_keywords(text)

Extracts key terms from the text while removing stopwords and punctuation.

### 3. recognize\_speech(duration)

Records speech from the microphone and converts it into text.

### 4. average\_embedding(text)

Computes Sentence-BERT embeddings for the given text.

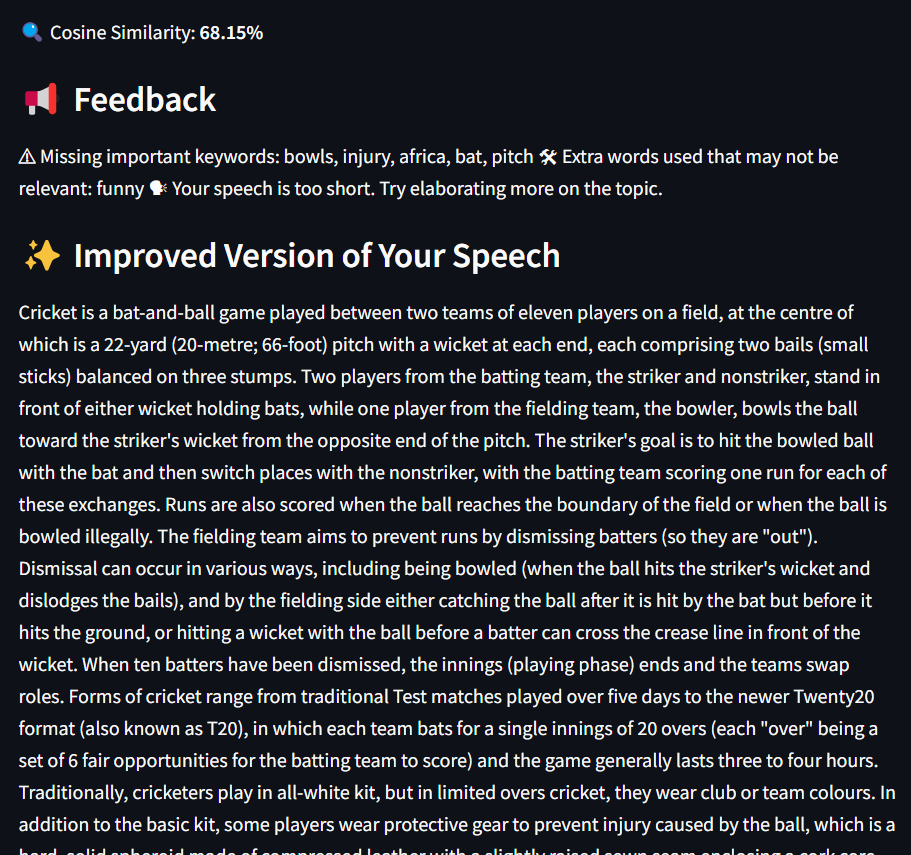
### 5. generate\_feedback(user\_speech, reference\_text)

Compares user speech with the reference text and provides feedback.

### 6. improve\_speech(user\_speech, reference\_text)

Suggests a revised version of the user's speech based on the reference text.





1. **Tamil Speech to Sign Language:**

The **Tamil Speech to Sign Language** system is designed to convert spoken Tamil words into corresponding sign language representations. This is particularly useful for communication between hearing and speech-impaired individuals and Tamil speakers.

**Features:**

* Converts **Tamil speech** into **Tamil text**.
* Translates the recognized Tamil text into **Tamil Sign Language hand gestures**.
* Displays the sign language representations using images.
* Interactive UI with a **speech recording** button.

**Workflow:**

1. **Speech Input**: User clicks "Record Speech" and speaks in Tamil.
2. **Speech-to-Text Conversion**: The system converts the speech into Tamil text.
3. **Text Processing**: Extracts individual Tamil characters.
4. **Sign Language Mapping**: Maps each Tamil letter to its corresponding sign language image.
5. **Display**: The images of hand gestures are displayed sequentially.

