VOICESCRIPT: TRANSFORMING INDIA'S ORAL TRADITION

A MINI PROJECT REPORT

Submitted by

DHIVYASHRI K K
23ITR037

HARIKARTHIK K V
23ITR053

HEMAVARSHINI V
23ITR060

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

KONGU ENGINEERING COLLEGE

(Autonomous)
PERUNDURAI, ERODE – 638 060

NOVEMBER 2024

DEPARTMENT OF INFORMATION TECHNOLOGY

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

NOVEMBER 2024

BONAFIDE CERTIFICATE

This is to certify that the Mini Project Report entitled ***VOICESCRIPT*: TRANSFORMING INDIA'S ORAL TRADITION** is the bonafide record of project work done by **DHIVYASHRI K K (23ITR037), HARIKARTHIK K V (23ITR053)** and **HEMAVARSHINI V(23ITR60).**

SUPERVISOR **HEAD OF THE DEPARTMENT**

(Signature with seal)

ABSTRACT

This project focuses on the development of a Speech-to-Text and Translation System, designed to address the linguistic diversity and communication challenges in India. With the ability to transcribe audio files in Indian languages such as Hindi, Bengali, Punjabi, Tamil, and Urdu, and translate them into English, the system bridges the gap between oral traditions and written communication. It leverages a robust combination of technologies to deliver efficient transcription and accurate translation services, catering to diverse user needs. The system is implemented using Python Flask for the backend and HTML, CSS, and JavaScript for the frontend, ensuring a seamless user experience.

The backend system processes audio files in formats like .mp3, .wav, and more, converting them into the .wav format for uniformity in audio processing and better accuracy in transcription. The transcribed text is then translated into English, ensuring both grammatical correctness and contextual accuracy. The translated text is saved and made available for download, providing users with a tangible output for future reference.

A significant feature of the project is its user-friendly web interface. Users can upload audio files, select the desired language for transcription, and view the transcription and translated output in real-time. The system also includes a live audio recording feature, which allows users to record their speech directly through their browsers. This eliminates the dependency on pre-recorded files and introduces real-time functionality, making the system highly versatile and suitable for dynamic applications. The results, including the transcription in the original language and the English translation, are displayed interactively, with the option to download the translated text for further use.

This project is particularly beneficial in addressing the challenges of cross-lingual communication. By enabling the transcription and translation of Indian languages, it supports linguistic inclusivity and accessibility. The system is ideal for various practical applications, including educational environments, where students and researchers can access content in different languages, and accessibility services, where individuals can interact across language barriers. It also has potential uses in digitizing oral traditions, helping preserve cultural heritage in written form.

To ensure reliability, the system incorporates error-handling mechanisms, such as providing meaningful error messages in cases of unsupported audio formats or transcription issues. This feature improves user experience and builds trust in the system's capabilities. Furthermore, the system's modular design allows for easy scalability and future enhancements.

Future enhancements for the system include the integration of advanced machine learning algorithms to improve transcription accuracy and translation quality. Expanding language support to include more Indian and global languages is another key objective. The incorporation of real-time transcription with native script output, text-to-speech conversion for translations, and deployment on cloud platforms for better scalability and accessibility are also being considered.

By focusing on inclusivity, accessibility, and innovation, this project aims to create a significant impact in fields like education, communication, and cultural preservation. Its ability to transform oral content into accessible written records while enabling seamless translation ensures it remains relevant across multiple domains. Through continuous improvement and expansion, this project aspires to contribute to a more connected, multilingual digital ecosystem.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Efficient and accurate voice-to-text transcription and translation systems have become essential in today's globalized world, where communication across languages is crucial. These systems help individuals and organizations bridge language barriers, improve accessibility, and enhance the overall user experience. In the healthcare, education, and customer service sectors, voice-to-text transcription can significantly improve efficiency and accessibility. This project aims to create an advanced voice-to-text transcription and translation system that can process speech in multiple languages, including Indian languages like Hindi, Bengali, Punjabi, Urdu, and Indian English, and provide accurate translations in English, all while offering seamless integration with a web interface.

1.1 OVERVIEW

1.1.1 Importance of Voice-to-Text Transcription and Translation

Voice-to-text transcription refers to the process of converting spoken language into written text. This technology has evolved significantly, powered by advanced machine learning algorithms and natural language processing (NLP). Voice-to-text systems are crucial for accessibility, especially for individuals with disabilities, and in situations where manual transcription is impractical.

Voice-to-text systems combined with translation capabilities allow users to convert speech from one language to another in real-time, enhancing communication across linguistic boundaries. This project aims to not only transcribe speech but also translate it to English, making it more accessible for non-native speakers. Furthermore, this system will cater to Indian languages and regional dialects, which are often underrepresented in mainstream speech-to-text and translation tools.

1.1.2 Addressing Multilingual Needs

India is home to over 20 official languages and hundreds of dialects, making it a linguistically diverse nation. Traditional transcription systems often fail to handle the intricacies of regional languages, particularly when converting speech into accurate written text or translating it to English. Our system will focus on overcoming these barriers by accurately transcribing and translating speech in languages like Hindi, Bengali, Punjabi, Tamil, Urdu, and Indian English.

1.1.3 Role of Technology in Improving Accessibility

By utilizing advanced technologies such as deep learning, speech recognition, and NLP, this project will enable users to convert their speech into text in real-time and translate it into English. It aims to simplify communication for users who speak different languages, thus fostering inclusion and providing access to services that might otherwise be limited by language. The integration of AI-driven features such as automated speech recognition (ASR) and machine translation (MT) will ensure a seamless user experience.

Key Features of the Project

The primary features of this voice-to-text transcription and translation system include:

- **Speech Recognition**: The ability to transcribe spoken words into text with high accuracy.
- **Multilingual Support**: Support for multiple languages and dialects, including Hindi, Bengali, Punjabi, Urdu, and Indian English.
- **Real-Time Translation**: The system translates transcribed text into English, making it accessible to a wider audience.
- **Cloud Integration**: Leveraging cloud services to handle large volumes of data and improve processing speed.
- **User-Friendly Interface**: A simple and intuitive web-based interface that allows users to access the service without technical expertise.
- **Voice-to-Text and Translation Synchronization**: Real-time transcription and translation, ensuring a seamless experience for users.

1.2 CHALLENGES

1.2.1 Data Security and Privacy Concerns

Since the system processes sensitive audio data, privacy and security are significant challenges. Ensuring the confidentiality of users' voice data, especially in healthcare and legal contexts, requires stringent security measures. Complying with data protection regulations, such as GDPR, and maintaining trust will be crucial for the system's adoption.

1.2.2 Multilingual and Accents Variability

India's linguistic diversity presents a challenge in accurately transcribing and translating speech, especially with regional accents. Variations in pronunciation, dialects, and slang can reduce the accuracy of transcription systems. The project must address these variations by training the system to recognize and adapt to different accents and linguistic nuances.

1.2.3 Handling Noisy Environments

Real-world environments often present challenges for voice recognition, such as background noise or overlapping speech. The system must be robust enough to handle such conditions and maintain high accuracy in transcription and translation.

1.2.4 Latency in Real-Time Processing

Real-time transcription and translation require efficient processing to ensure minimal delay. The project must address latency issues by optimizing the system's backend, ensuring that transcription and translation occur without significant delays, even with large audio files.

1.2.5 Scalability and Resource Constraints

Handling large volumes of audio data for real-time processing requires substantial computing power. Efficient scaling to accommodate multiple users, especially in real-time scenarios, presents a challenge, particularly in cloud-based implementations.

1.2.6 Limited Availability of Pre-trained Models for Indian Languages

While models for English and other major languages are widely available, resources for Indian languages are often limited. Training models to accurately transcribe and translate Indian languages, especially for languages with complex scripts like Hindi and Urdu, requires significant computational resources and expertise.

1.3 OBJECTIVE

This project aims to create an effective, accurate, and user-friendly voice-to-text transcription and translation system that caters to multiple languages, with a focus on Indian languages. The key objectives include:

1. **Accurate Transcription of Speech**

   Develop a highly accurate speech-to-text engine capable of transcribing spoken words into text for multiple languages, especially Indian languages like Hindi, Bengali, Punjabi, and Urdu.

2. **Real-Time Translation**

   Enable seamless real-time translation of transcribed text into English to improve communication between speakers of different languages.

3. **Multilingual Support**

   Provide support for a range of Indian languages and dialects, ensuring that users from different linguistic backgrounds can access the system.

4. **Accessible User Interface**

   Design an easy-to-use, web-based interface for both end-users and developers, ensuring that the system is accessible to a wide audience with varying levels of technical expertise.

5. **Integration with Existing Technologies**

   Integrate with existing technologies, such as cloud services, to manage large data volumes and ensure efficient processing without latency.

6. **Improve User Experience**

   Create a smooth and responsive experience, particularly in noisy or real-world environments, by utilizing advanced noise cancellation and error correction techniques.

7. **Scalable and Robust System**

   Ensure that the system can scale efficiently to accommodate a large number of users and large audio files without compromising performance or accuracy.

CHAPTER 2

PROBLEM STATEMENT

➢ Challenge of multilingual communication in india

- India's linguistic diversity poses a significant challenge in achieving seamless communication across its many languages. Accurately transcribing speech into native scripts and translating it into English is essential for fostering inclusivity and improving accessibility.
- With widely spoken languages like Hindi, Indian English, Urdu, Bengali, Punjabi, and Tamil, the need for an efficient system tailored to handle such diversity is more critical than ever.

➢ Complexity of accents and pronunciations

- Indian accents vary significantly, not only between states but also within regions, creating a challenge for traditional transcription systems. These systems often fail to process unique pronunciations, intonations, and linguistic nuances, resulting in inaccuracies.
- Additionally, specialized handling is required for languages with distinct scripts, such as Urdu and Tamil, to ensure proper representation during transcription.

➢ Limitations in translation systems

- Accurately translating transcriptions into English requires preserving contextual and cultural nuances. Standard systems often fall short in this aspect, leading to loss of meaning and ineffective communication.
- Addressing this challenge demands advanced solutions capable of handling the complexities of multilingual translation.

➢ Technology-driven solution approach

- This project proposes a system designed to transcribe speech from Indian languages into their native scripts while translating them accurately into English. By leveraging advanced NLP , the system adapts to regional linguistic patterns and ensures high accuracy.
- Noise-canceling features further enhance real-time transcription accuracy, making the solution suitable for diverse environments.

➢ Socioeconomic impact of the solution

- By bridging the communication gap between regional languages and English, this system addresses critical barriers in education, healthcare, governance, and digital inclusivity. It empowers individuals by enabling effective cross-lingual communication and supports India's multilingual ecosystem.
- Ultimately, this project promotes unity in India's linguistically rich society, fostering inclusivity and digital integration for individuals with diverse linguistic backgrounds.

CHAPTER 3

METHODOLOGY

3.1 System design and architecture

The system will be designed with a focus on ensuring accuracy in transcription and translation while catering to the diverse linguistic needs of Indian users. The front-end will be developed using HTML, CSS, and JavaScript to provide an intuitive interface where users can upload audio files, record live speech, and view results. The back-end, built with Python (Flask), will handle the core functionalities, such as audio processing, transcription, and translation. It will integrate APIs for speech recognition and translation, ensuring seamless communication between the user interface and the processing modules

3.2 Speech recognition and accent adaptation

The speech recognition module will utilize Python's SpeechRecognition library, integrated with Google Speech-to-Text API for improved accuracy. The system will handle multiple Indian languages, including Hindi, Tamil, Urdu, and Bengali, adapting to the wide variety of Indian accents. Custom language models and accent-specific tuning will be employed to minimize errors caused by regional pronunciation differences. Additionally, noise cancellation techniques will ensure robust performance in real-world, noisy environments.

3.3 Text translation module

The text translation module will leverage advanced AI-based translation APIs, such as Google Translate, to convert transcribed native language text into English. The system will focus on preserving the contextual and cultural nuances of the source language during translation. To achieve this, language models will be fine-tuned for common Indian phrases, idioms, and grammatical structures, ensuring that the translated text remains accurate and meaningful.

3.4 Live captioning and real-time processing

The live captioning feature will enable users to view transcriptions and translations in real-time while speaking. This will be achieved through asynchronous processing, where the system continuously processes audio chunks and displays results on the user interface. Noise reduction algorithms will be applied to enhance the clarity of live transcriptions, making the feature ideal for both individual and group settings, such as classrooms or meetings.

3.5 Downloadable text functionality

After transcription and translation, users will have the option to download the results as text files in formats such as `.txt` . The back-end will handle file generation using Python’s built-in libraries, ensuring that the

downloaded text retains proper formatting and alignment. This functionality provides users with a convenient way to save and share their transcription and translation outputs for further use.

3.6 Multilingual accessibility and user experience

To cater to users across India, the system will offer a user-friendly interface with options to select preferred languages for transcription and translation. The interface will feature clear navigation, dark mode, and support for multiple scripts, ensuring accessibility for all users, including those unfamiliar with English. The design will prioritize inclusivity, ensuring that the system is as intuitive and efficient as possible.

CHAPTER 4
IMPLEMENTATION

4.1 FRONT-END DEVELOPMENT

The front-end is implemented using HTML, CSS, JavaScript, and Bootstrap to create an interactive and visually appealing user interface. Key components include:

- **Audio Upload and Recording**: Users can upload audio files or record live speech directly through the browser.
- **Language Selection**: Dropdowns enable users to choose the input language and the desired output format (native script or translated English text).
- **Results Display**: Transcriptions and translations are displayed dynamically, with formatting to ensure readability.
- **Responsive Design**: Implemented with Bootstrap to ensure compatibility across devices, including desktops and smartphones.
  JavaScript is used for managing real-time updates, such as displaying live captions or showing progress during uploads and processing.

4.2 BACK-END IMPLEMENTATION

The back-end is developed using Flask to manage API endpoints and core processing workflows. Major features include:

- **Audio Preprocessing**: The `pydub` library is used to normalize audio and convert it into a format suitable for speech recognition.
- **API Integration**: Google's Speech-to-Text API is called for transcribing speech into text, while Google Translate API handles text translation into English.
- **Process Handling**: Flask handles the flow of data between the front-end, transcription, and translation modules. Multi-threading is used to improve performance for real-time features.
  The back-end ensures secure communication and data processing with error-handling mechanisms for invalid inputs or failed API responses.

4.3 SPEECH RECOGNITION MODULE

Speech recognition is achieved using Python's SpeechRecognition library and Google's Speech-to-Text API. Steps include:

- **Audio Conversion**:

  Audio files are converted to `.wav` format if necessary, ensuring compatibility.

- **Accent Optimization**:

  The Google API is configured to handle Indian accents and supports multiple regional languages like Hindi, Tamil, Urdu, and Bengali.

- **Real-Time Updates**:

  For live captions, audio chunks are sent for recognition in near real-time, ensuring minimal latency.

4.4 TEXT TRANSLATION MODULE

The translation module relies on Google Translate API to convert transcriptions from native scripts into English. Implementation highlights:

- **Native Script Handling**: Proper Unicode support is ensured for languages like Tamil and Urdu, preserving their specific script formats during processing.
- **Contextual Translation**: Special attention is given to preserving the meaning and cultural context of the text during translation.

4.5 LIVE CAPTIONING

Live captioning is implemented to provide real-time transcription of audio inputs. Key aspects include:

- **Chunk Processing**: Audio is split into smaller, manageable segments that are processed sequentially to reduce delays.
- **Dynamic Updates**: Captions are displayed in the front-end as they are generated, allowing users to follow speech instantaneously.
- **Noise Filtering**: Background noise is minimized using preprocessing, ensuring that captions remain accurate even in noisy environments.

4.6 DOWNLOAD FUNCTIONALITY

The download functionality is implemented using **Flask's `send_file` utility** to provide users with formatted output files.

- **Supported Formats**: Transcriptions and translations can be downloaded as `.txt` or `.pdf` files.
- **Dynamic File Creation**: Files are generated on the fly after processing, ensuring updated and accurate content.
- **Secure File Access**: Temporary file paths are used to manage downloads, ensuring data privacy.

CHAPTER 5

RESULTS AND DISCUSSION

The voice-to-text transcription and translation system was successfully implemented and demonstrated effective real-time speech-to-text conversion for a variety of Indian languages, including Hindi, Urdu, Tamil, Punjabi, and Bengali. The system's ability to handle diverse accents and regional dialects was impressive, but some minor inaccuracies were observed, especially with non-standard or fast speech. In terms of transcription accuracy, the system worked well in clear speech scenarios, with minor transcription errors arising in cases of heavy accents or noisy environments. The translation from native scripts into English was performed efficiently; however, some cultural nuances and idiomatic expressions did not fully translate, affecting the overall meaning in certain instances.

The integration of natural language processing allowed for smooth functioning, but challenges still arose with complex sentences and phrases that did not align well with the translation algorithms. The system's ability to provide downloadable text files and offer real-time transcription with minimal delay enhanced the overall user experience. The frontend interface was designed to be user-friendly, allowing for easy navigation and interaction with the system. Despite the positive outcomes, challenges remained in addressing edge cases such as mixed language use and regional slang, which required further refinement of the models.

While the system demonstrated significant potential for bridging linguistic gaps, the results highlighted areas where further improvements are necessary. The system performed well with standard speech, accurately transcribing and translating text, but its accuracy diminished in handling regional accents, colloquialisms, and fast speech. In particular, the ability to preserve contextual meaning during translation was sometimes compromised, as the system occasionally failed to capture cultural nuances that are integral to communication in languages like Urdu and Tamil. Despite these limitations, the system’s core functionality, including real-time transcription and downloadable text output, was highly effective and responsive. The AI-based language models used for both transcription and translation showed good adaptability to different languages but need more refinement to handle complex sentence structures and mixed-language inputs, which are common in Indian multilingual contexts.

Future enhancements could include more robust noise-canceling algorithms, better training for specific dialects, and advanced contextual understanding to improve translation quality. The user interface performed well, though there is room for improvement in responsiveness, especially in regions with less stable internet connections. Overall, the project showed the potential for improving cross-lingual communication in India but needs further refinement to handle the complexity and diversity of Indian languages effectively.

CHAPTER 6

CONCLUSION

The voice-to-text transcription and translation system developed for Indian languages has shown promising results in addressing the challenges of multilingual communication in India. The system successfully transcribes speech from various Indian languages into their respective native scripts and translates the transcribed text into English, offering a significant improvement in accessibility. Although the system performs well under standard conditions, challenges persist in handling regional accents, dialects, and cultural nuances during translation. The integration of AI and natural language processing has proven effective in simplifying the transcription process, but further refinements are required to address edge cases such as mixed-language inputs and complex sentence structures. Despite these challenges, the system's ability to provide real-time transcription, enable seamless downloads, and improve communication across linguistic barriers holds great potential. As the system evolves and incorporates more training data and enhanced algorithms, it can further contribute to bridging language gaps, improving digital inclusivity, and fostering better communication in India's diverse multilingual landscape.

APPENDIX 1

CODING

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Voice Script - transforming india's oral tradition</title>
  <link rel="stylesheet" href="static/style.css">
  <link rel="icon" href="https://th.bing.com/th/id/OIP.rL_rED1XSv-
5uSqxR7_ZUgHaHa?w=178&h=180&c=7&r=0&o=5&dpr=1.3&pid=1.7">
</head>
<body>
  <div class="container">
    <h1>Speech to Text and Translation</h1>

    <form id="uploadForm" enctype="multipart/form-data">
      <label for="audioFile">Upload an Audio File:</label>
      <input type="file" id="audioFile" name="audio" accept="audio/*" required>

      <label for="language">Select Language:</label>
      <select id="language" name="language" required>
        <option value="hi-IN">Hindi</option>
        <option value="en-IN">Indian English</option>
        <option value="ur-IN">Urdu</option>
        <option value="bn-IN">Bengali</option>
        <option value="pa-IN">Punjabi</option>
        <option value="ta-IN">Tamil</option>
      </select>
```

```
    <button type="button" id="uploadBtn">Upload File</button>
  </form>

  <div id="recordingSection">
    <button type="button" id="startRecord">Start Recording</button>
    <button type="button" id="stopRecord" disabled>Stop Recording</button>
  </div>

  <div id="results" style="display: none;">
    <h2>Results</h2>
    <p><strong>Transcription:</strong> <span id="transcription"></span></p>
    <p><strong>Translation:</strong> <span id="translation"></span></p>
    <a id="downloadLink" href="#" style="display: none;" download>Download Translated
Text</a>
  </div>
 </div>

 <script>
  let mediaRecorder;
  let audioChunks = [];

  document.getElementById("uploadBtn").onclick = async function() {
   const file = document.getElementById("audioFile").files[0];
   const language = document.getElementById("language").value;
   if (!file) {
    alert("Please select an audio file.");
    return;
   }

   const formData = new FormData();
   formData.append("audio", file);
   formData.append("language", language);
```

```
  try {
    const response = await fetch("/upload", {
      method: "POST",
      body: formData,
    });

    if (!response.ok) {
      const errorData = await response.json();
      alert("Error: " + (errorData.error || "Something went wrong"));
      return;
    }

    const result = await response.json();
    document.getElementById("transcription").textContent = result.transcription;
    document.getElementById("translation").textContent = result.translation;

    document.getElementById("results").style.display = "block";
    const downloadLink = document.getElementById("downloadLink");
    downloadLink.href = result.download_link;
    downloadLink.style.display = "inline-block";

  } catch (error) {
    alert("Error: Unable to process the file. Please try again.");
  }
};
document.getElementById("startRecord").onclick = async function() {
  try {
    const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
    mediaRecorder = new MediaRecorder(stream);

    mediaRecorder.ondataavailable = function(event) {
      audioChunks.push(event.data);
```

```
    };

    mediaRecorder.onstop = async function() {
      const audioBlob = new Blob(audioChunks, { type: 'audio/wav' });
      const audioFile = new File([audioBlob], 'audio.wav');

      const formData = new FormData();
      const language = document.getElementById("language").value;
      formData.append("audio", audioFile);
      formData.append("language", language);

      const response = await fetch("/upload", {
        method: "POST",
        body: formData,
      });

      const result = await response.json();
      document.getElementById("transcription").textContent = result.transcription;
      document.getElementById("translation").textContent = result.translation;

      document.getElementById("results").style.display = "block";
      const downloadLink = document.getElementById("downloadLink");
      downloadLink.href = result.download_link;
      downloadLink.style.display = "inline-block"; // Show the download link
    };

    mediaRecorder.start();
    document.getElementById("startRecord").disabled = true;
    document.getElementById("stopRecord").disabled = false;
  } catch (err) {
    alert("Error: " + err);
  }
};
```

```
    document.getElementById("stopRecord").onclick = function() {
      mediaRecorder.stop();
      document.getElementById("startRecord").disabled = false;
      document.getElementById("stopRecord").disabled = true;
    };
  </script>
</body>
</html>
```

Style.css

```
{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Arial', sans-serif;
  background-color: #f8f8f8;
  color: #333;
  padding: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
}

.container {
  background: linear-gradient(135deg, #000000, #ffffff);
  padding: 30px;
  border-radius: 15px;
  box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
  width: 100%;
```

```
  max-width: 600px;
  text-align: center;
}

h1 {
  font-size: 2.5rem;
  margin-bottom: 20px;
  color: white;
}

form {
  background-color: white;
  padding: 25px;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
  display: flex;
  flex-direction: column;
  align-items: center;
}

form label {
  font-size: 1rem;
  color: #333;
  margin-bottom: 8px;
  text-align: left;
  width: 100%;
}

form input[type="file"],
form select,
form button {
  width: 100%;
  padding: 10px;
```

```
  margin: 10px 0;
  border: 1px solid #ccc;
  border-radius: 5px;
  font-size: 1rem;
  color: #333;
}

form input[type="file"]:hover,
form select:hover,
form  button:hover  {
  border-color:  #000;
  transition: 0.3s ease;
}

form button {
  background-color: #000;
  color: white;
  font-weight: bold;
  cursor: pointer;
}

form button:hover {
  background-color: #444;
}

#results {
  margin-top: 20px;
  padding: 20px;
  background-color: white;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}
```

```
#results h2 {
  font-size: 1.5rem;
  margin-bottom: 10px;
}

#results p {
  font-size: 1rem;
  margin-bottom: 15px;
  color: #555;
}

#downloadLink {
  display: block;
  margin-top: 10px;
  text-align: center;
  font-size: 1rem;
  color: black;
  text-decoration: none;
  padding: 10px;
  border-radius: 5px;
  background-color: #fff;
  border: 1px solid #000;
}

#downloadLink:hover {
  background-color: #000;
  color: white;
}

@media screen and (max-width: 768px) {
  .container {
    width: 90%;
  }
```

```
  h1 {
    font-size: 2rem;
  }
}
```

App.py

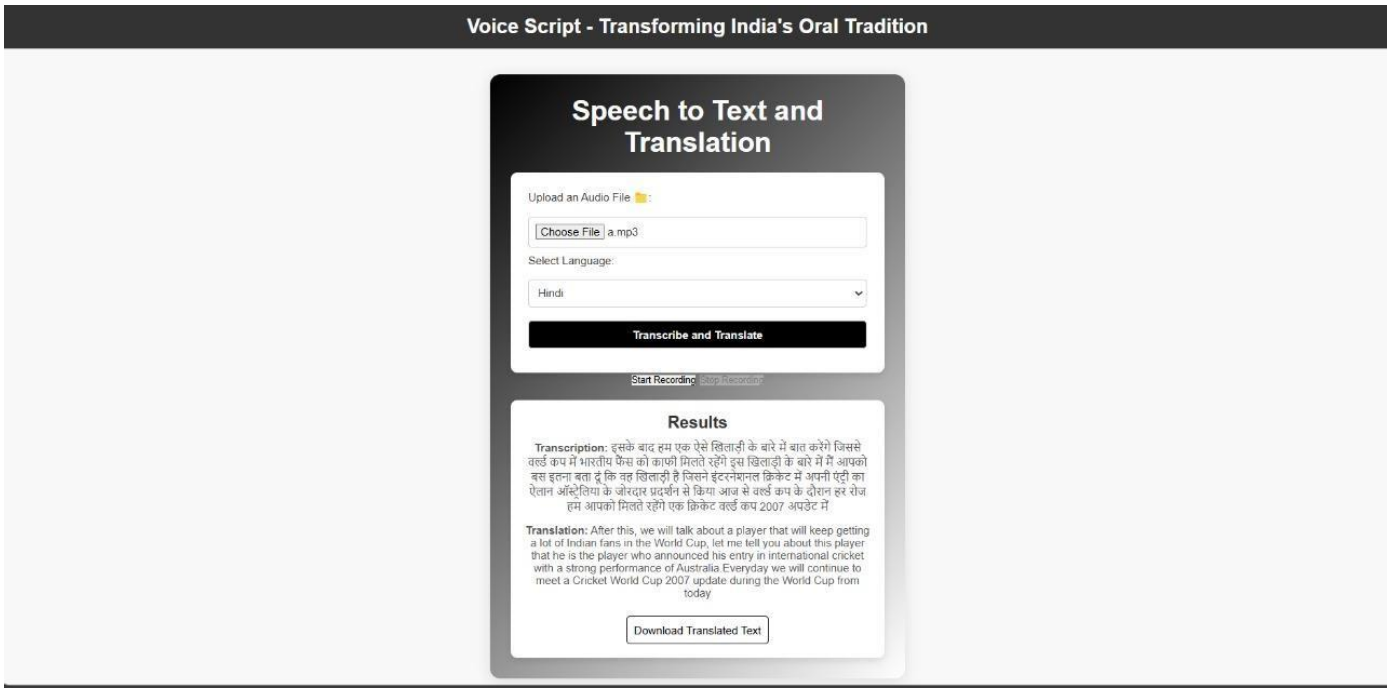
```
import os
from flask import Flask, render_template, request, jsonify, send_from_directory
from pydub import AudioSegment
import speech_recognition as sr
from googletrans import Translator

app = Flask(_name_)
UPLOAD_FOLDER = 'uploads'
TRANSLATED_FOLDER = 'translated_files'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['TRANSLATED_FOLDER'] = TRANSLATED_FOLDER

# Ensure folders exist
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

if not os.path.exists(TRANSLATED_FOLDER):
    os.makedirs(TRANSLATED_FOLDER)

def prepare_voice_file(path: str) -> str:
    audio_file = AudioSegment.from_file(path)
    wav_file = os.path.splitext(path)[0] + '_converted.wav'
    audio_file = audio_file.set_frame_rate(16000).set_channels(1)
    audio_file.export(wav_file, format='wav')
    return wav_file
```

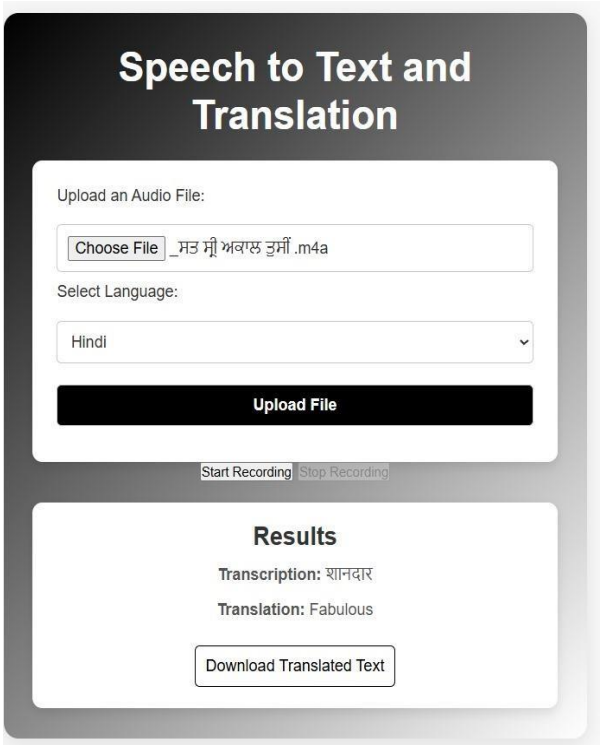
```
def transcribe_audio(audio_file_path, language) -> str:
    recognizer = sr.Recognizer()
    with sr.AudioFile(audio_file_path) as source:
        audio_data = recognizer.record(source)
    text = recognizer.recognize_google(audio_data, language=language)
    return text

def translate_to_english(text: str, source_lang: str) -> str:
    translator = Translator()
    translation = translator.translate(text, src=source_lang, dest='en')
    return translation.text

def save_translation_to_file(translated_text: str, filename: str) -> str:
    file_path = os.path.join(app.config['TRANSLATED_FOLDER'], filename)
    with open(file_path, 'w', encoding='utf-8') as f:
        f.write(translated_text)
    return file_path

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'audio' not in request.files:
        return jsonify({'error': 'No audio file provided'}), 400

    file = request.files['audio']
    language = request.form.get('language')
    if file.filename == '' or language == '':
        return jsonify({'error': 'File or language not provided'}), 400

    filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
```

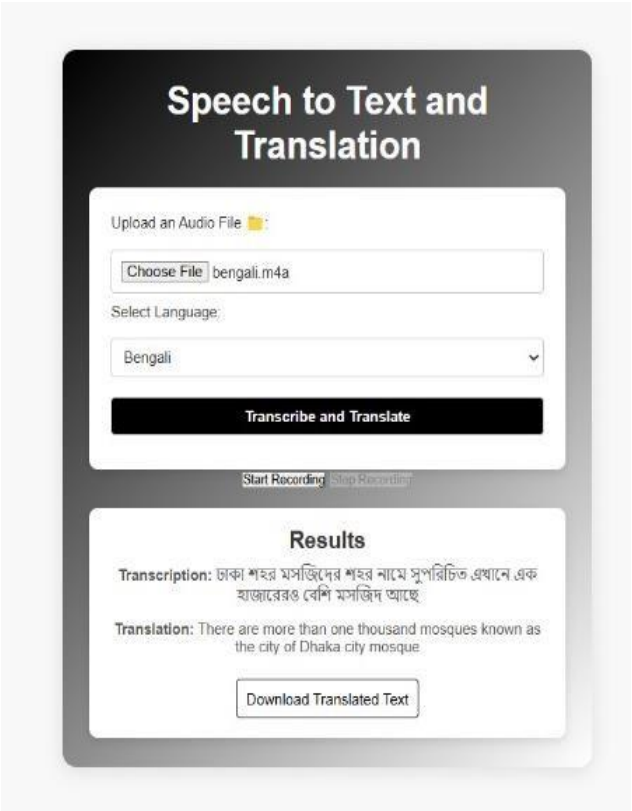
```
    file.save(filepath)

    try:
        wav_file = prepare_voice_file(filepath)
        transcription = transcribe_audio(wav_file, language)
        translation = translate_to_english(transcription, language[:2])
        translated_file_path = save_translation_to_file(translation, file.filename.split('.')[0] +
'_translated.txt')
    except Exception as e:
        return jsonify({'error': str(e)}), 500

    return jsonify({
        'transcription': transcription,
        'translation': translation,
        'download_link': f"/download/{os.path.basename(translated_file_path)}"
    })

@app.route('/download/<filename>')
def download_file(filename):
    return send_from_directory(app.config['TRANSLATED_FOLDER'], filename)

if __name__ == '__main__':
    app.run(debug=True)
```

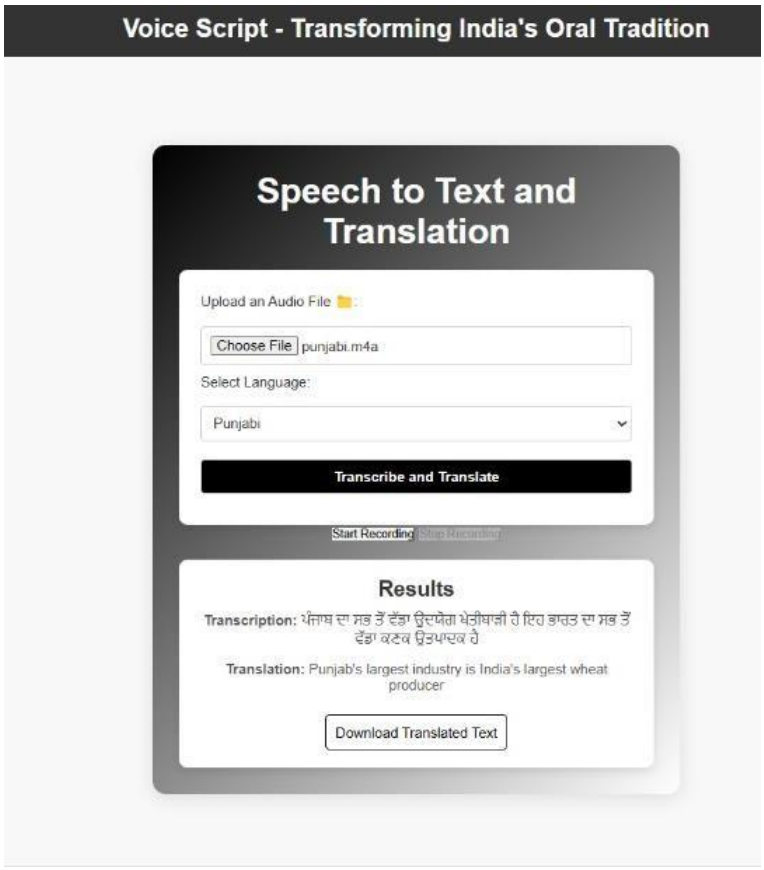
APPENDIX 2

SNAPSHOTS



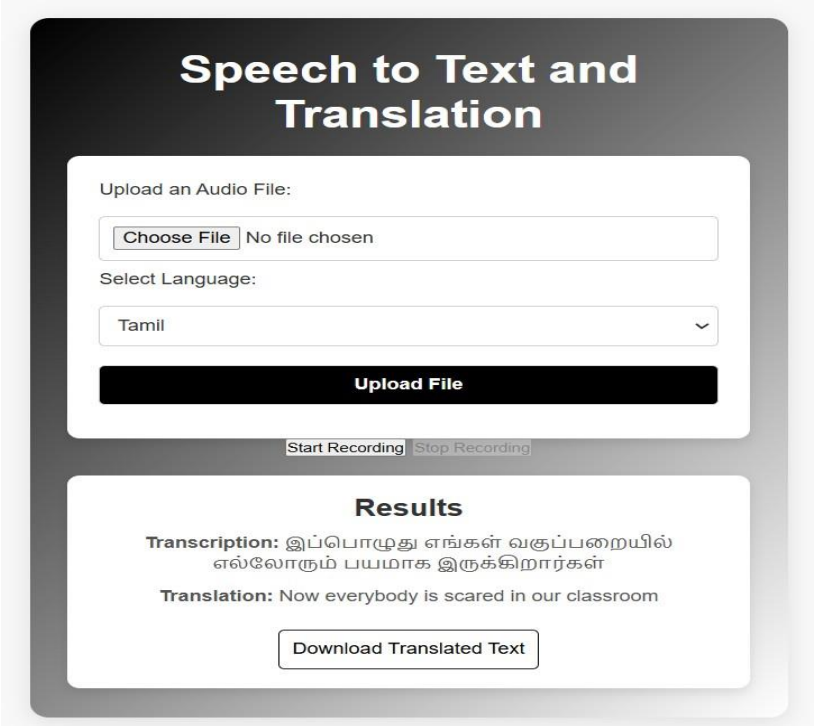


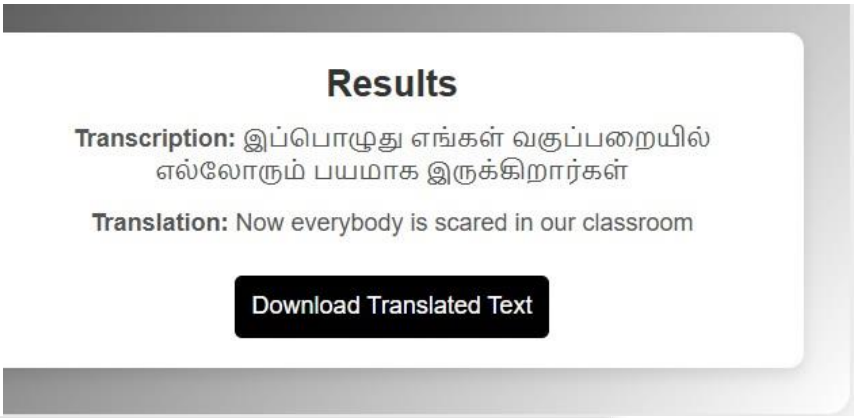


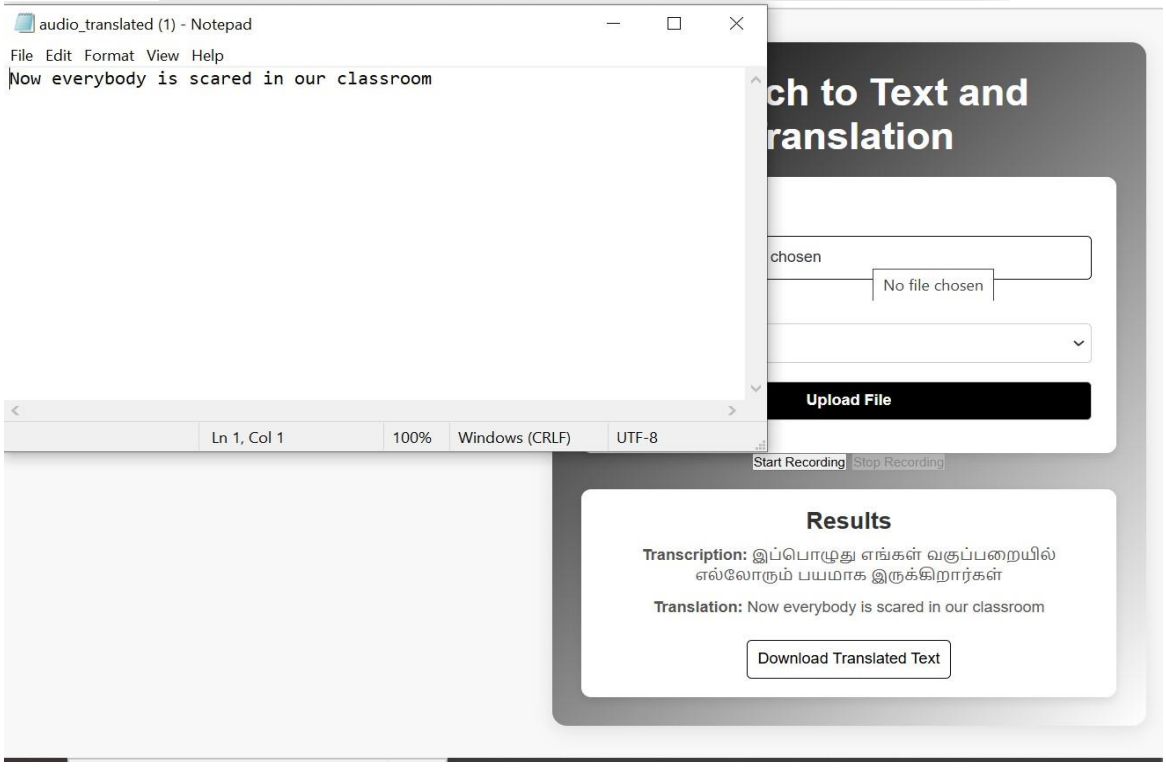

Voice Script - Transforming India's Oral Tradition
Speech to Text and Translation
Upload an Audio File :
Choose File punjabi.m4a
Select Language:
Punjabi
Transcribe and Translate
Start Recording Stop Recording
Results
Transcription: ਪੰਜਾਬ ਦਾ ਸਭ ਤੋਂ ਵੱਡਾ ਉਦਯੋਗ ਖੇਤੀਬਾੜੀ ਹੈ ਇਹ ਭਾਰਤ ਦਾ ਸਭ ਤੋਂ ਵੱਡਾ ਕਣਕ ਉਤਪਾਦਕ ਹੈ
Translation: Punjab's largest industry is India's largest wheat producer
Download Translated Text


Speech to Text and Translation
Upload an Audio File:
Choose File _வணக்கம் நீங்கள் எப்.m4a
Select Language:
Tamil
Upload File
Start Recording Stop Recording
Results
Transcription: வணக்கம் நீங்கள் எப்படி இருக்கிறீர்கள் தமிழ் மொழி மற்றும் கலாச்சாரத்தை அறிந்து கொள்வது ஒரு அற்புதமான விஷயம் யூ
Translation: Hello how are you a wonderful thing to know the Tamil language and culture
Download Translated Text


Voice Script - Transforming India's Oral Tradition
Speech to Text and Translation
Upload an Audio File :
Choose File urdu.m4a
Select Language:
Urdu
Transcribe and Translate
Start Recording Stop Recording
Results
Transcription:
Translation: TV has good and bad effects on the booth. Watching TV time wasting time. This baby is unable to work on time. They are unable to eat on time. Amy and Abu are worried.Is
Download Translated Text


Speech to Text and Translation
Upload an Audio File:
Choose File No file chosen
Select Language:
Tamil
Upload File
Start Recording Stop Recording
Results
Transcription: இப்பொழுது எங்கள் வகுப்பறையில் எல்லோரும் பயமாக இருக்கிறார்கள்
Translation: Now everybody is scared in our classroom
Download Translated Text


Results
Transcription: இப்பொழுது எங்கள் வகுப்பறையில் எல்லோரும் பயமாக இருக்கிறார்கள்
Translation: Now everybody is scared in our classroom
Download Translated Text


audio_translated (1) - Notepad
File Edit Format View Help
Now everybody is scared in our classroom
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
ch to Text and
ranslation
chosen
No file chosen
Upload File
Start Recording Stop Recording
Results
Transcription: இப்பொழுது எங்கள் வகுப்பறையில் எல்லோரும் பயமாக இருக்கிறார்கள்
Translation: Now everybody is scared in our classroom
Download Translated Text

REFERENCES

[1] https://www.veed.io/tools/audio-to-text

[2] https://www.speech-to-text.cloud/

[3] https://github.com/harikarthik12/voice-script-report