# A

**Project Based Assignment**

**On**

**Project PyDataStruct**

**Submitted**

**By**

RANGDAL PAVANSAI 22QM1A6748

WAGMARE SANJANA 22QM1A6760

U SAI HRUTHVIN 21N81A67C5

POLUDASU NAGA PRABHAS 22QM1A67A white rectangular sign with purple and yellow logo

Description automatically generated46

Under the Guidance of

**Mrs. AFIYA PARVEEN**

Asst. Prof

**Department of Computer Science and Engineering –Data Science**

III-B.Tech II-Semester

Academic Year 2024-25

**KG REDDY COLLEGE OF ENGINEERING &**

**TECHNOLOGY**

**Chilkur(V), Moniabad(M). Ranga Reddy(D), Telangana – 501504**

**A white rectangular sign with purple and yellow logo

Description automatically generated**

# CERTIFICATE

This is to certify that the Dissertation Report entitled, **“Project Xara”** submitted by Mr/Ms.“ **RANGDAL PAVANSAI (22QM1A6748), WAGMARE SANJANA (22QM1A6760), U. SAI HRUTHVIN (21N81A67C5) , POLUDASU NAGA PRABHAS (22QM1A6746)**, “of III-B.Tech II-Semester, Department of Computer Science and Engineering-Data Science from KG Reddy College of Engineering & Technology, Moniabad, India, is a record of Bonafede, Project Based Assignment completed by him/her under my/ours supervision.

|  |  |
| --- | --- |
| **Internal Guide**  Mrs. Afiya Parveen  (Asst. Prof.)  Dept. of CSE-Data Science  KG Reddy College of Engineering & Tech. | **PBA Coordinator**  Mrs. P. Haindavi  (Asst. Prof.)  Dept. of CSE-Data Science  KG Reddy College of Engineering & Tech. |
| **PRC Coordinator**  Mrs. Afiya Parveen  (Asst. Prof.)  Dept. of CSE-Data Science  KG Reddy College of Engineering & Tech. | **Head of the Dept.**  Mr. L. Govardhan  (Asst. Prof. & HoD)  Dept. of CSE-Data Science  KG Reddy College of Engineering & Tech. |

# DECLARATION

We Certify that

1. The work contained in this report is original and has been done by me under the guidance of my Guide.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. We have followed the guidelines provided by the Institute for preparing the report.
4. We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
5. Whenever we have used materials (Data, Theoretical analysis, figures and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in reference. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

|  |  |  |
| --- | --- | --- |
| **NAME OF THE STUDENT** | **ROLL NO.** | **Sign** |
| RANGDAL PAVASAI | 22QM1A6748 |  |
| WAGMARE SANJANA | 22QM1A6760 |  |
| U. SAI HRUTHIVN | 21N81A67C5 |  |
| POLUDASU NAGA PRABHAS | 22QM1A6746 |  |

# ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Mrs. P. Haindavi**, Assistant Professor, CSE- Data Science and Project Supervisor, Department of Computer Science and Engineering - Data Science, KG Reddy College of Engineering & Technology, for his able guidance and useful suggestions, which helped us in completing the project in time.

We are particularly thankful to **Mr. L. Govardhan**, the Head of the Department, Department of Computer Science and Engineering - Data Science, his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honorable Principal **Dr. Sai Satyanarayana Reddy**, for providing all facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartful thanks **to Dr. K. Krishna Reddy Garu**, Chairman and **Dr. Rohit Kandakatla**, Director of KG Reddy College of Engineering & Technology, for providing a congenial atmosphere to complete this project successfully.

We also thank all the staff members of Department of Computer Science and Engineering - Data Science, KG Reddy College of Engineering & Technology for their valuable support and generous advice. Finally, thanks to all our friends and family members for their continuous support and enthusiastic help.

|  |
| --- |
| RANGDAL PAVANSAI |
| WAGMARE SANJANA |
| U. SAI HRUTHVIN |
| POLUDASU NAGA PRABHAS |

# ABSTRACT

The "Xara! Text-To-Speech Project by Tech T-Rex" is a simple and interactive web application designed to convert text into speech in various languages. Built using Python and Streamlit, the app provides a platform for users to upload different types of files, extract text, translate it into their preferred language, and listen to the translated content as speech. The application supports text extraction from plain text files, PDFs, Word documents, and even images. For images, it uses Optical Character Recognition (OCR) to detect and extract the text. Once the text is extracted, users can review or edit it directly in the app. A key feature of Xara is its ability to translate text into multiple languages. Users can choose from a list of Indian and international languages through easy-to-use dropdown menus. To avoid confusion, the app ensures that only one type of language (Indian or international) can be selected at a time. If the user uploads a long text, the app splits it into smaller, manageable chunks to ensure smooth translation and text-to-speech conversion. It uses the Google Translator service to translate the text, automatically detecting the source language and converting it into the selected language. Once the text is ready, the app uses gTTS (Google Text-to-Speech) to generate audio from the translated text. Users can listen to the audio directly within the app and enjoy the results of their input and selections. The interface is clean and user-friendly, featuring file upload options, text editing areas, and buttons for performing translation and text-to-speech operations. Additionally, a link to the project's GitHub repository is provided for users who want to explore the code behind the application. The app is ideal for anyone looking for a simple way to convert text from various sources into speech in their preferred language. It combines multiple tools and functionalities into one platform, making it a versatile and accessible solution for text-to-speech and translation tasks.

# LIST OF FIGURES

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Fig No.** | **Description of the image** | **Pg. No.** |
| 1 | 5.1 | Flow chart of Xara | 14 |
| 2 | 8.1 | Front look of Xara | 19 |
| 3 | 8.2 | Working of Xara | 19 |

# TABLE OF CONTENT

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Chapter No.** | **Topic** | **Pg. No.** |
| 1 | Chapter – 0 | Certificate  Declaration  Acknowledgement  Abstract  List of Figures  List of Tables  Table of Content | 2  3  4  5  6  6  7 |
| 2 | Chapter – 1 | Introduction | 8 |
| 3 | Chapter – 2 | Background Research | 9 |
| 4 | Chapter – 3 | Objectives | 11 |
| 5 | Chapter – 4 | Methodology | 12 |
| 6 | Chapter – 5 | Design | 14 |
| 7 | Chapter – 6 | Execution Procedure | 16 |
| 8 | Chapter – 7 | Outputs | 19 |
| 9 | Chapter – 8 | Future Scope | 20 |
| 10 | Chapter – 9 | Appendix Code | 23 |
| 11 | Chapter – 10 | Conclusion | 28 |
| 11 | Chapter – 11 | References | 39 |

# CHAPTER – 1

## INTRODUCTION

The application of this app is "Xara! Text-To-Speech Project by Tech T-Rex." This application will transform any text from any format-text file, PDF, Word, or even images to speech. Furthermore, it translates multilingually, thereby catering to both Indian and international languages, hence addressing the needs of the entire world.

In the age of information and digital world, accessibility and effective processing of information become utmost necessities. Xara relies on cutting-edge applications including Optical Character Recognition, machine translation, and text-to-speech synthesis in offering an integrated solution. With the OCR functionality available, retrieval of text from scanned documents, images, or any non-editable formats makes it extremely helpful for a wide variety of documents. This international translation feature of the application shall break the language as the text converter to a various number of international languages from Indian, such as Hindi, Tamil, Telugu languages, while the international ones would be French, Spanish and Japanese etc. in this consideration and is covered by context relevance due to advanced Translation API.

It produces natural speaking audio from the translated text so that it is helpful for blind users and aurally oriented learners. The software, as a rule, tends to have good pronunciation and offers interruption-free user experience.

It has an intuitive interface developed using Streamlit. Users can upload files, view the extracted text, select target languages for translation, and listen to the generated audio-all in an interactive and intuitive interface. Users are always able to edit the extracted text before translation or speech generation for extra control and customization.

Xara lies at the heart of accessibility. Features like playback within the application for audio directly make it highly useful for educators, professionals, and students, making it easy to process text in an inclusive manner. It is especially valuable in multilingual societies; it smoothes over language and document format barriers.

The program, "Xara! Text-To-Speech", makes text processing more inclusive, effective, and relevant for the global world by bringing together OCR, machine translation, and text-to-speech synthesis all in one in the way of redefining accessibility and efficiency in forms of presentation or communication through.

# CHAPTER – 2

## BACKGROUND RESEARCH

### 2.1 The Need for Text-to-Speech Applications

#### Accessibility Challenges:

* + Visually impaired individuals or those with reading disabilities often struggle to access written content.
  + Text-to-speech applications provide an alternative, enabling them to "listen" to the content.

#### Language Barriers:

* + In a multilingual society, people often encounter written text in languages they do not understand.
  + Translation and text-to-speech tools help bridge these gaps.

#### Convenience in Information Consumption:

* + Auditory learning has become a preferred mode for many individuals.
  + Applications like audiobooks and virtual assistants rely heavily on text-to-speech technologies.

### 2.2 Existing Technologies

#### OCR (Optical Character Recognition):

* + OCR has evolved from basic text extraction systems to advanced tools like EasyOCR, capable of recognizing multilingual and complex layouts with high accuracy. Initially used for digitizing printed text, OCR now plays a crucial role in extracting information from images, scanned documents, and PDFs, enabling digital workflows and accessibility.

#### Translation Services:

* + Tools like Google Translate have revolutionized communication by breaking language barriers, supporting hundreds of languages. While they excel in literal translations, challenges persist in contextual accuracy and handling regional dialects, which are crucial for nuanced communication.

#### Text-to-Speech Systems:

* + Early TTS systems produced robotic sounds, but modern engines like gTTS generate natural, human-like speech. These systems are integrated into smartphones, IoT devices, and web apps, enhancing user accessibility, especially for visually impaired individuals and multilingual applications.

### 2.3 Current Limitations in Available Tools

#### Fragmented Functionality:

* + Most existing tools focus on one aspect: text extraction, translation, or speech synthesis.
  + Users often have to rely on multiple tools to achieve their goals.

#### Language Support Gaps:

* + Limited support for Indian languages in global applications.
  + Poor pronunciation and lack of regional dialect support in many TTS systems.

#### User Experience Challenges:

* + Complex interfaces and technical barriers can deter non-technical users.

### 2.4 Research Insights Driving Xara

#### Integrated Solution:

* + There’s a gap in the market for an all-in-one platform that handles text extraction, translation, and speech synthesis seamlessly.

#### Focus on Multilingual Accessibility:

* + The need to support Indian languages alongside major international languages.
  + Enable accurate pronunciation and translation of complex scripts.

#### User-Centric Design:

* + Simplify the process with an interactive interface that even non-technical users can navigate.

#### Application in Real-World Scenarios:

* + Use cases such as education, document accessibility, content creation, and language learning.

# CHAPTER – 3

## OBJECTIVES

### 3.1 Ensure Data Privacy and Security:

Implement robust security protocols to protect user data and comply with privacy regulations.

### 3.2 Optimize Performance and Speed:

Ensure fast processing times for text extraction, translation, and speech synthesis for a seamless user experience.

### 3.3 Support Customization and Personalization:

Allow users to adjust settings such as voice pitch, speed, and language preferences to suit individual needs.

### 3.4 Facilitate Offline Functionality:

Develop offline capabilities for essential features to support users in low-connectivity environments.

### 3.5 Integrate Multi-Device Compatibility:

Ensure the platform is accessible and fully functional across different devices (smartphones, tablets, desktops, etc.)

### 3.6 Enable Scalability:

Design the platform to handle increased demand and scale as user needs grow over time.

### 3.7 Incorporate a Feedback Mechanism:

Include a system for users to provide feedback and suggest improvements for continuous development.

### 3.8 Support Content Extraction from Diverse Formats:

Expand OCR capabilities to handle various document types, such as PDFs, images, and scanned handwritten documents.

### 3.9 Provide Real-Time Translation and Speech Synthesis:

Enable instant translation and speech synthesis for real-time communication needs.

### 3.10 Ensure Cross-Platform Synchronization:

Allow users to access and synchronize their data and progress seamlessly across devices.

### 3.11 Promote User Engagement and Education:

Offer tutorials, help sections, and interactive guides to educate users on maximizing the platform's features.

# CHAPTER – 4

## METHODOLOGY

### 4.1 Features

* **File Support:** Upload text files, PDFs, Word documents, or images.
* **Language Options:** 
  + - Indian languages like Hindi, Tamil, Bengali, etc.
    - International languages like French, German, Korean, etc.
* **Text Extraction:** Extracts text from uploaded files, including OCR for images.
* **Translation:** Translates text into the selected language.
* **Speech Synthesis:** Converts translated text into speech using Google Text-to-Speech (gTTS).

### 4.2 Libraries Used

* **Streamlit (streamlit):** For creating an interactive web application.
* **Google Text-to-Speech (gTTS):** To generate speech from text.
* **Deep Translator (deep\_translator):** For translating text into different languages.
* **EasyOCR (easyocr):** To extract text from images.
* **Pillow (PIL):** For handling image processing.
* **NumPy (numpy):** To manipulate image data.
* **PyPDF2 (PyPDF2):** To read PDF content.
* **python-docx (docx):** To read content from Word documents.
* **chardet (chardet):** For detecting the encoding of text files.

### Application Structure

### User Interface

#### Title and Description:

* + - Displayed using st.title() and st.write().

#### Language Selection:

* + - Indian and international languages are provided in separate dropdown menus (st.selectbox()).

### File Upload

* + - Allows users to upload files of type .txt, .pdf, .docx, .jpg, .jpeg, .png.
    - Handles each file type using conditional checks on uploaded\_file.type.

### Text Handling

* + Extracts text from the uploaded file.
  + **For text files:** Decodes content using chardet.
  + **For PDFs:** Extracts content page by page using PyPDF2.
  + **For Word documents:** Reads paragraphs using python-docx.
  + **For images:** Extracts text using EasyOCR.

### Translation and Text-to-Speech

#### Translation:

* + - * + Uses Google Translator (deep\_translator.GoogleTranslator) to translate the text into the selected language.
        + Splits large texts into chunks (<5000 characters) to meet translation API limits.

#### Text-to-Speech:

* + - * + Converts translated text into audio using gTTS.
        + Saves and plays the audio file.

### User Feedback

* + - * Displays extracted text and translated text for user verification using st.text\_area().
      * Provides warnings and errors for invalid inputs or selections.

# CHAPTER – 5

## DESIGN

### 5.1 FLOW CHART

A screenshot of a computer code

Description automatically generated

**Fig 5.1:** Flow chart of Xara

### FLOW DIAGRAM (Textual Representation)

[Title and Welcome Message]

↓

[Language Selection Dropdowns]

↓

[File Uploader] → [Extract Text Based on File Type]

↓

[Display Extracted Text in Text Area]

↓

[User Edits or Inputs Text in Text Area]

↓

[Translate Button Pressed] → [Split Text into Chunks]

↓

[Translate Chunks to Selected Language]

↓

[Convert Translated Text to Speech]

↓

[Play Audio in the App]

↓

[Display Translated Text in Text Area]

↓

[Code Button] → [Redirect to GitHub]

↓

[Footer Section]

# CHAPTER – 6

## EXECUTION PROCEDURE

**Step 1: Set Up Environment**

1. **Install Python:**
   * Ensure Python 3.7 or above is installed on your system. You can download it from [python.org](https://www.python.org/).
2. **Create a Virtual Environment:**
   * Open a terminal or command prompt and navigate to your project directory.
   * Create a virtual environment:

python -m venv myenv

* + Activate the virtual environment:
    - **On Windows:**

myenv\Scripts\activate

* + - **On macOS/Linux:**

source myenv/bin/activate

**Step 2: Install Required Libraries**

1. Install the required Python libraries by running:

pip install streamlit gtts deep-translator easyocr pillow numpy PyPDF2 python-docx chardet

1. **Install Tesseract-OCR for EasyOCR:**
   * **On Windows:**
     + Download and install Tesseract from [Tesseract OCR GitHub](https://github.com/tesseract-ocr/tesseract).
     + Add the Tesseract installation path to the system environment variables.
   * **On macOS/Linux:**
     + Install via a package manager:

sudo apt install tesseract-ocr

**Step 3: Save the Code**

1. Copy the provided Python code.
2. Save it as a file named app.py in your project directory.

**Step 4: Run the Streamlit App**

1. Open a terminal or command prompt in the directory containing app.py.
2. Run the Streamlit server:

streamlit run app.py

1. The app will open in your default web browser. If not, copy the URL displayed in the terminal and paste it into your browser.

**Step 5: Interact with the App**

1. **Select a Language:**
   * Choose an Indian or International language from the dropdown menus.
   * Ensure only one language is selected at a time.
2. **Upload a File:**
   * Upload a text file, PDF, Word document, or image.
   * The app will extract and display text from the uploaded file.
3. **Translate and Speak:**
   * Edit the extracted text if needed in the text area.
   * Click the "Translate and speak" button to translate the text and generate speech.
   * Listen to the audio output directly in the app.
4. **Access GitHub Repository:**
   * Click the "Code!" button to navigate to the GitHub repository for this project.

**Step 6: Manage Output Files**

* Audio files generated by the app (translated\_output.mp3) will be saved in the same directory as app.py.

**Step 7: Stop the Server**

* To stop the Streamlit server, press Ctrl+C in the terminal or command prompt.

# CHAPTER – 7

## OUTPUTS

|  |
| --- |
| A screenshot of a computer  Description automatically generated **Fig 8.1:** Front look of Xara |
| A screenshot of a computer  Description automatically generated **Fig 8.2:** Working of Xara |

# CHAPTER – 8

## FUTURE SCOPE

**1. Advanced Language Understanding and Processing**

* **Integration of Advanced NLP Techniques:**
  + Implement state-of-the-art Natural Language Processing models like GPT-series for better context understanding and dynamic responses.
* **Multilingual Support:**
  + Expand Xara’s capability to handle more regional and international languages with real-time translation.

**2. Personalization and Customization**

* **User Preferences:**
  + Allow users to customize Xara’s personality, tone, and response style.
* **Learning User Behaviour:**
  + Implement machine learning algorithms to adapt to user preferences over time.

**3. Integration with IoT and Smart Devices**

* **Smart Home Control:**
  + Enable Xara to control smart home devices like lights, thermostats, and appliances via voice commands.
* **Wearable Technology Integration:**
  + Develop Xara’s interface for smartwatches and other wearables to provide on-the-go assistance.

**4. Enhanced Multimedia Capabilities**

* **Image and Video Processing:**
  + Extend Xara’s ability to understand and process multimedia inputs for applications like object recognition, scene understanding, or video analysis.
* **Augmented Reality (AR):**
  + Integrate AR features for real-world applications like virtual tours, education, and gaming.

**5. Domain-Specific Applications**

* **Healthcare:**
  + Provide teleconsultation, appointment scheduling, and health monitoring.
* **Education:**
  + Act as a personalized tutor for students, helping with studies, career guidance, and skill development.
* **Finance:**
  + Offer financial advice, track expenses, and provide market insights.

**6. Integration with Third-Party Platforms**

* **Enterprise Use:**
  + Deploy Xara as a virtual assistant for businesses, handling customer support and internal task automation.
* **Social Media Integration:**
  + Enable Xara to interact with users on platforms like WhatsApp, Facebook Messenger, and Instagram.

**7. Ethical and Responsible AI**

* **Data Privacy and Security:**
  + Ensure robust measures to protect user data and comply with global data protection regulations.
* **Bias Mitigation:**
  + Develop mechanisms to eliminate biases in Xara’s responses to ensure fairness and inclusivity.

**8. Offline Functionality**

* **Edge Computing:**
  + Enable Xara to perform key functions offline, increasing accessibility in areas with limited internet connectivity.

**9. Gamification and Entertainment**

* **Interactive Games:**
  + Include conversational games and quizzes for entertainment and learning.
* **Storytelling:**
  + Allow Xara to narrate stories or deliver content dynamically based on user input.

**10. Continuous Learning and Feedback**

* **User Feedback Mechanism:**
  + Provide users with the ability to give feedback to continuously improve Xara’s performance.
* **Automatic Updates:**
  + Implement a system for seamless updates to integrate new features and improvements regularly.

# CHAPTER – 9

## APPENDIX CODE

import streamlit as st

from gtts import gTTS

from deep\_translator import GoogleTranslator

import easyocr

from PIL import Image

import numpy as np

import PyPDF2

from docx import Document

import chardet

st.title("Xara! Text-To-Speech Project by Tech T-Rex")

st.write("### Welcome to Xara! Text-To-Speech Bot")

# Define the language options

indian\_languages = {

    "Bengali": 'bn',

    "Gujarati": 'gu',

    "Hindi": 'hi',

    "Kannada": 'kn',

    "Malayalam": 'ml',

    "Marathi": 'mr',

    "Punjabi": 'pa',

    "Tamil": 'ta',

    "Telugu": 'te',

    "Urdu": 'ur',

    "Nepali": 'ne',

}

international\_languages = {

    "Arabic": 'ar',

    "Chinese": 'zh-Cn',

    "Dutch": 'nl',

    "English": 'en',

    "French": 'fr',

    "German": 'de',

    "Italian": 'it',

    "Japanese": 'ja',

    "Korean": 'ko',

    "Portuguese": 'pt',

    "Russian": 'ru',

    "Spanish": 'es',

    "Swedish": 'sv',

    "Turkish": 'tr',

    "Thai": 'th',

    "Vietnamese": 'vi',

    "Persian": 'fa',

    "Swahili": 'sw',

    "Filipino": 'tl',

    "Finnish": 'fi',

    "Hungarian": 'hu',

    "Hebrew": 'iw',

    "Malay": 'ms',

    "Ukrainian": 'uk'

}

# Dropdowns for Indian and International languages

st.write("#### Select Language")

indian\_language\_choice = st.selectbox("Choose an Indian Language", ["None"] + list(indian\_languages.keys()))

international\_language\_choice = st.selectbox("Choose an International Language", ["None"] + list(international\_languages.keys()))

# Logic to handle selection

if indian\_language\_choice != "None":

    international\_language\_choice = "None"  # Reset international selection

    st.warning("You have selected an Indian language. The international language selection has been reset.")

elif international\_language\_choice != "None":

    indian\_language\_choice = "None"  # Reset Indian selection

    st.warning("You have selected an International language. The Indian language selection has been reset.")

# File uploader for text input

uploaded\_file = st.file\_uploader("Upload a text file, PDF, Word document, or image", type=["txt", "pdf", "docx", "jpg", "jpeg", "png"])

# Variable to store the text to be read

mytext = ""

if uploaded\_file is not None:

    # Handling different file types

    if uploaded\_file.type == "text/plain":

        # Read and decode the text file using chardet

        raw\_data = uploaded\_file.read()

        result = chardet.detect(raw\_data)

        encoding = result['encoding']

        mytext = raw\_data.decode(encoding)

        st.success("Text file uploaded successfully!")

    elif uploaded\_file.type == "application/pdf":

        pdf\_reader = PyPDF2.PdfReader(uploaded\_file)

        mytext = ""

        for page in pdf\_reader.pages:

            mytext += page.extract\_text() + "\n"

        st.success("PDF file uploaded successfully!")

    elif uploaded\_file.type == "application/vnd.openxmlformats-officedocument.wordprocessingml.document":

        doc = Document(uploaded\_file)

        mytext = "\n".join([para.text for para in doc.paragraphs])

        st.success("Word document uploaded successfully!")

    elif uploaded\_file.type in ["image/jpeg", "image/png"]:

        # Open the image file with PIL

        image = Image.open(uploaded\_file)

        # Convert the image to a NumPy array

        image\_np = np.array(image)

        # Initialize EasyOCR reader

        reader = easyocr.Reader(['en'])  # Specify the language

        result = reader.readtext(image\_np)

        # Extract and concatenate text from image

        mytext = " ".join([text[1] for text in result])

        st.success("Image file uploaded successfully!")

    # Display the text to be translated

    st.write("### Extracted Text:")

    st.write(mytext)

# Text area for display and editing

user\_input\_text = st.text\_area("Text to be read aloud:", mytext, height=100)

# Add a variable to store the translated text

translated\_text = ""

col1, col2 = st.columns(2)

with col1:

    if st.button("Translate and speak"):

        if not user\_input\_text:

            st.warning("Please enter text to translate.")

        elif indian\_language\_choice == "None" and international\_language\_choice == "None":

            st.warning("Please select a language to translate.")

        else:

            try:

                # Ensure the text is not empty

                if not user\_input\_text.strip():

                    st.warning("The text is empty. Please upload a valid file or provide text.")

                else:

                    # Choose the correct language code

                    chosen\_language = indian\_languages.get(indian\_language\_choice) or international\_languages.get(international\_language\_choice)

                    if not chosen\_language:

                        st.warning("Language selection is invalid.")

                    else:

                        # Split the text into chunks of less than 5000 characters

                        chunk\_size = 5000

                        chunks = []

                        start = 0

                        # Split text carefully to avoid cutting in the middle of a word

                        while start < len(user\_input\_text):

                            end = min(start + chunk\_size, len(user\_input\_text))

                            if end < len(user\_input\_text) and user\_input\_text[end] != ' ':

                                end = user\_input\_text.rfind(' ', start, end)

                            chunks.append(user\_input\_text[start:end].strip())

                            start = end

                        # Log the chunk sizes to verify they are under the limit

                        st.write(f"Total chunks: {len(chunks)}")

                        for i, chunk in enumerate(chunks):

                            st.write(f"Chunk {i+1} size: {len(chunk)} characters")

                        # Loop through each chunk and convert to speech

                        for chunk in chunks:

                            if chunk.strip():  # Ensure there's text to process

                                try:

                                    # Translate the text

                                    translated\_chunk = GoogleTranslator(source='auto', target=chosen\_language).translate(chunk)

                                    translated\_text += translated\_chunk + " "  # Append each chunk to the translated text

                                    # Convert the translated text to speech

                                    output = gTTS(text=translated\_chunk, lang=chosen\_language, slow=False)

                                    output\_file = "translated\_output.mp3"

                                    output.save(output\_file)

                                    st.audio(output\_file)  # Play the audio

                                except Exception as e:

                                    st.error(f"An error occurred during translation or speech generation: {e}")

            except Exception as e:

                st.error(f"An error occurred during translation: {e}")

        # Display the translated text in a text box after translation

        st.text\_area("Translated Text:", translated\_text, height=200)

with col2:

    if st.button("Code!"):

        github\_url = "https://github.com/Pavansai20054/gTTS/blob/main/pages/2\_Text-To-Speech.py"

        st.markdown(f'<meta http-equiv="refresh" content="0; url={github\_url}">', unsafe\_allow\_html=True)

st.markdown("""

    <footer style="text-align: center; padding: 10px;">

        <p>Copyright (c) 2024 Pavansai. All rights reserved.</p>

        <p>Licensed under the MIT License. See LICENSE file for details.</p>

    </footer>

    """, unsafe\_allow\_html=True)

# CHAPTER – 10

## CONCLUSION

Project Xara represents a significant step forward in the integration of artificial intelligence into personalized and interactive user experiences. The chatbot’s ability to communicate naturally and perform diverse functionalities demonstrates its potential to transform various domains such as education, healthcare, customer support, and entertainment.

By combining advancements in natural language processing, multilingual support, and voice-based interaction, Xara sets a foundation for creating a seamless, user-friendly digital assistant. Its unique focus on customization and cultural relevance—symbolized by its Indian identity—makes it stand out in a rapidly evolving AI landscape.

**Key Takeaways**

* Xara provides a robust, scalable, and adaptable framework for conversational AI applications.
* The incorporation of text-to-speech, translation, and file-processing capabilities opens up opportunities for diverse applications.
* Its design is both user-centric and future-ready, with a roadmap for incorporating advanced AI features.

As Xara evolves, continuous learning, user feedback, and technology upgrades will play pivotal roles in ensuring its relevance and effectiveness. The journey of Xara is a testament to the transformative power of AI and its ability to enhance lives through meaningful interactions.

# CHAPTER – 11

## REFERENCES

1. Shijia Liao, Yuxuan Wang, Tianyu Li, Yifan Cheng, Ruoyi Zhang, Rongzhi Zhou, Yijin Xing. Fish-Speech: Leveraging Large Language Models for Advanced Multilingual Text-to-Speech Synthesis. arxiv.org/abs/2411.01156
2. Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. (2017). Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. IEEE Transactions on Image Processing, 26(7), 3142-3155.
3. Brown, T. B., et al. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, 33, 1877-1901.
4. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep Contextualized Word Representations. Proceedings of NAACL-HLT 2018, 2227-2237.
5. Rao, S., & Sharma, A. (2022). EasyOCR: A Practical Implementation for Multilingual Optical Character Recognition. International Journal of Computer Applications, 182(41), 17-22.
6. Chen, X., Duan, Y., & He, J. (2021). Neural Machine Translation Systems for Multilingual Applications. Springer Artificial Intelligence Review, 54(2), 312-336.
7. Takahashi, T., Nakagawa, K., & Matsuo, Y. (2020). Enhancing Multilingual OCR with Deep Learning and Pre-trained Models. Pattern Recognition Letters, 132, 119-126.
8. Gupta, P., & Jain, S. (2019). Advances in Multilingual Speech Synthesis Using Neural Architectures. Journal of Speech Communication Research, 42(3), 95-102.
9. Kumar, V., & Singh, R. (2021). Hybrid Techniques in Text-to-Speech Synthesis for Indian Languages. Springer Communications in Computer and Information Science, 1350, 215-230.
10. Rath, R., & Mohapatra, P. (2020). Applications of AI in Multilingual OCR and Translation. MDPI Applied Sciences, 10(4), 1247.
11. Xu, J., & Hu, Y. (2021). Leveraging Transformer Models for Multilingual Speech and Text Processing. IEEE Transactions on Speech and Language Processing, 8(2), 341-356.
12. Han, W., & Kim, J. (2022). Advances in OCR and Text-to-Speech for Multilingual Applications. Elsevier Journal of Information Science and Technology, 89, 117-129.