MULTILINGUAL OCR AND AUDIO SYNTHESIS PLATFORM

MINI PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS TO

RGUKT- SRIKAKULAM

FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By:

B.SIREESHA	**S190649**
B.SREEHITHA	**S190605**
A.SRINIVAS	**S191041**

Under the Esteemed Guidance of

Mrs.K.Sita Durga, M.Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RGUKT-SRIKAKULAM, ETCHERLA- June 2024
RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

CERTIFICATE

This is to certify that the mini project report titled "**MULTILINGUAL OCR AND SPEECH SYNTHESIS PLATFORM**" was successfully completed by **BYREDDY SIREESHA (S190649)**, **BASIREDDY SREEHITHA (S190605)**, **APPANA SRINIVAS (S191041)** under the guidance of Mrs.K.SEETHA DURGA Assistant Professor. In partial fulfillment of the requirements for the Mini Project in Computer Science and Engineering of **Rajiv Gandhi University of Knowledge Technologies** under my guidance and output of the work carried out is satisfactory.

Project Guide
Mrs.K.Sita Durga, M.Tech
Assistant Professor

Head of the Department
Mrs.Ch.Lakshmi Bala, M.Tech, (Ph.D)
Assistant Professor

DECLARATION

We declared that this thesis work titled "**MULTILINGUAL OCR AND SPEECH SYNTHESIS PLATFORM**" is carried out by me during the year 2023-2024 in partial fulfilment of the requirements for the Mini Project in **Computer Science and Engineering**.

We further declare that this dissertation has not been submitted elsewhere for any Degree. The matter embodied in this dissertation report has not been submitted elsewhere for any other degree. Furthermore, the technical details furnished in various chapters of this thesis are purely relevant to the above project and there is no deviation from the theoretical point of view for design, development and implementation.

B.Sireesha (S190649)
B.Sreehitha (S190605)
A.Srinivas (S191041)

ACKNOWLEDGEMENT

We would like to articulate my profound gratitude and indebtedness to our project guide **Mrs.Sita Durga** Mam, who has always been a constant motivation and guiding factor throughout the project time. It has been a great pleasure for us to get an opportunity to work under her guidance and complete the thesis work successfully.

We wish to extend our sincere thanks to **Mrs.Ch.Lakshmi Bala** mam Head of the Computer Science and Engineering Department, for her constant encouragement throughout the project. We are also grateful to other members of the department without their support our work would have not been carried out so successfully.

I thank one and all who have rendered help to me directly or indirectly in the completion of my thesis work.

Project Associate

B.Sireesha (S190649)

B.Sreehitha (S190605)

A. Srinivas (S191041)

ABSTRACT

Digital images are rapidly gaining popularity across various fields such as education, engineering, and healthcare. Extracting meaningful information from the text present in these images can greatly enhance their utility. This process is pivotal in creating ebooks from scanned books, facilitating image searches, and providing multilingual and audio translations of text. Optical Character Recognition (OCR) systems, particularly Tesseract, developed by HP Labs and now owned by Google, are essential for this task. The text extraction process involves several steps: text detection, localization, segmentation, and binarization, ultimately converting images into ASCII text. By employing advanced OCR and Natural Language Processing (NLP) technologies, our system not only accurately extracts and translates text from images but also offers both written and spoken translations by using googletrans and google Text to Speech (gTTS). This dual approach ensures accessibility for diverse audiences, including those with visual impairments or those who prefer auditory learning.

Keywords: OCR, Tesseract, text detection, text localization, text segmentation, NLP, googletrans, gTTS.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction

With the increasing prevalence of digital images across various fields such as education [1], engineering [2], healthcare [3], business, public services, and for travelers, the need to extract and utilize the textual information embedded within these images has become more apparent. In education, for instance, digitizing textbooks and historical documents through text extraction allows for easier dissemination of knowledge and facilitates the development of interactive learning platforms. Engineers benefit from OCR technology by automating the analysis of technical drawings, specifications, and reports, enhancing efficiency in design and manufacturing processes. In healthcare, extracting texvt from medical records, prescriptions, and diagnostic images supports accurate data management and improves patient care through streamlined information access.

The ability to extract and utilize text from digital images not only enhances operational efficiency across diverse sectors but also promotes accessibility by enabling the conversion of printed information into formats that are searchable, translatable, and accessible to individuals with visual impairments. This underscores the significant role of OCR technology in modern information management, accessibility initiatives, and enhancing the travel experience for individuals worldwide.

1.2 Problem Statement

Despite technological advancements, accessing text-based information remains challenging for travelers and students. Existing OCR systems often lack the capability to convert text into both written and spoken forms, limiting their effectiveness. This gap hampers language learning opportunities for students and restricts travelers from understanding texts in diverse contexts. There is a critical need for an innovative system integrating advanced OCR and NLP to recognize text within images and convert it into written and spoken forms effectively. Such a system would enhance accessibility for students, facilitating language learning and providing access to educational materials in multiple languages, while empowering travelers with seamless access to vocalized and translated information, thereby improving their overall experience.

1.3 Objective

The objective of this project is to develop a robust software system that integrates advanced Optical Character Recognition (OCR) and Natural Language Processing (NLP) technologies. This system will allow users to upload images containing text, accurately extract and convert this text into multiple languages, and generate spoken audio outputs. By focusing on software development, the project aims to enhance accessibility for users with visual impairments and facilitate language learning opportunities for students.

1.4 Motivation for the work

The motivation behind this project stems from the critical need to improve accessibility to text-based information embedded within digital images. Current technologies often lack seamless integration and comprehensive functionality to effectively extract, translate, and vocalize text, limiting accessibility for users with visual impairments and hindering language learning opportunities [4]. By developing a unified software solution that combines advanced Optical Character Recognition (OCR) with Natural Language Processing (NLP), this project aims to fill these gaps and enhance usability across diverse user scenarios [5]. The software's ability to convert image-based text into both written and spoken formats not only empowers individuals with disabilities but also facilitates efficient information retrieval and comprehension for travellers, students, and professionals alike. Ultimately, the project seeks to leverage technology to foster inclusivity, improve educational outcomes, and enhance user experiences in accessing and interacting with textual content in today's digital age.

2. Literature Survey

Ref.	Authors	Title	Journal/Conference/Book	Year
[1]	V. Yadav, N. Ragot	Text Extraction in Document Images: Highlight on Using Corner Points	2016 12th IAPR Workshop on Document Analysis Systems (DAS)	2016
[3]	Xue, W., Q. Li, Q. Xue	Text detection and recognition for images of medical laboratory reports with a deep learning approach	IEEE Access	2019
[4]	Almeida, A. M. P., et al.	Development of an online digital resource accessible for students with visual impairment or blindness	Work	2020
[5]	Piotrowski, M.	Natural language processing for historical texts	Morgan & Claypool Publishers	2012
[6]	Sisodia, P., S. W. A. Rizvi	Optical character recognition development using Python	Journal of Informatics Electrical and Electronics Engineering	2023
[7]	Kamaraj, S. A., et al.	Enhancing Automatic Speech Recognition and Speech Translation Using Google Translate	Handbook of Research on Data Science and Cybersecurity Innovations in Industry 4.0 Technologies	2023
[8]	Tensmeyer, C., T. Martinez	Historical document image binarization: A review	SN Computer Science	2020
[9]	Nachtegael, M., et al.	Fuzzy filters for noise reduction: The case of Gaussian noise	The 14th IEEE International Conference on Fuzzy Systems (FUZZ'05)	2005
[10]	Dutoit, T.	An introduction to text-to-speech synthesis	Springer Science & Business Media	1997

Table 1: LIterature Survey

3. System Analysis

3.1 Proposed System

The proposed system allows users to upload an image and select the language of the text within it. Using pytesseract [6], the system extracts the text from the image. The user can then choose their preferred language for translation. The text is translated using Google Translate (googletrans) and converted into audio using Google Text-to-Speech (gTTS) [7]. The system displays the translated text and plays the corresponding audio, making the information accessible in both written and spoken formats, enhancing usability for users with diverse needs.

3.2 Design

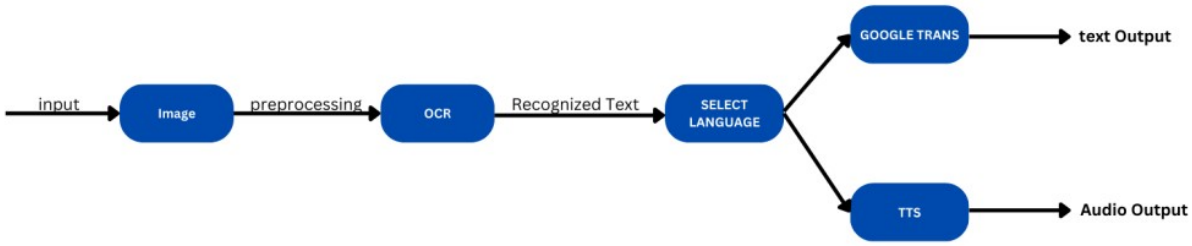

Figure 1: Image to text speech conversion flow

Image Input

At this stage, the process begins with the input of an image containing text. This image could be sourced from various places such as images, photographs, or screenshots.

Optical Character Recognition(OCR)

The image containing text is passed through an Optical Character Recognition (OCR) system. OCR software processes the image to recognize and extract the text embedded within it. This step is crucial for converting the visual information of the text into a machine-readable format.

Language Selection

Now select your desired option by clicking the provided options.

Googletrans

One of the best Python packages to help you with this task is Googletrans. The Google Translate API allows developers to easily access translations, allowing them to translate text from one language to another. The googletrans package provides a simple API for translating text from one language to another.

TTS

TTS, an acronym for Text-to-Speech, is speech synthesis technology that converts written text to spoken words. Note that it synthesizes words rather than playing back pre-recorded messages.

Language Translation

The extracted text, which is typically in one language, is then translated into multiple languages. Language translation services or APIs are used for this purpose. The translated text allows for broader accessibility and understanding across different linguistic backgrounds.

Translated Text

After translation, the text is available in multiple languages. Each translated version represents the original text in a different language. This step enables the content to be comprehensible to speakers of various languages, thus enhancing its reach and impact.

Audio Output

Finally, the audio output containing the synthesized speech is generated. This audio output represents the translated text in spoken form. It can be seamlessly integrated into audio playback systems or incorporated into applications, offering enhanced listening convenience and accessibility.

Overall, this block diagram outlines the process of extracting text from an image, translating it into multiple languages, and converting it into audio. Each step contributes to making the content more accessible and understandable to a diverse audience with different language preferences and accessibility needs.

3.3 System Requirements

3.3.1 Software Requirements

pytesseract: OCR tool for text extraction.

googletrans: Library for translating text.

gTTS (Google Text-to-Speech): Library for converting text to speech.

Flask: Web framework for creating the user interface.

OpenCV: Library for image processing.

Numpy: Library for numerical operations.

3.3.2 Hardware Requirements

Operating System: Ubuntu 23.02

CPU: AMD Ryzen 3 3250U with Radeon Graphics

RAM: 8GB

Disk Storage: 1TB of free disk space

Internet Connection: Required for translation and text-to-speech services.

4. Preliminaries

4.1 Binarization:

Binarization is the process in which the data features of the text are converted into the vectors of binary number, which will help the classification algorithms to work more efficiently [8]. When the binarization is applied to the grayscale image, the 0-255 spectrum is converted to 0-1 spectrum.

4.2 Noise Reduction:

In noise reduction, the noise within the image is reduced using different techniques such as smoothening of the image using Gaussian, Mean and Bilateral filter [9]. However, sometimes this may lead to hiding the fine details and contrast in the image. In the project we are applying the noise reduction to the image in the pre-processing, this affects the accuracy of the result at high margin. A subset of noise reduction is Color noise reduction in which the noise within the color is reduced to have stable colors.

4.3 Skew Correction:

Skew correction is a preprocessing step in optical character recognition (OCR) systems aimed at rectifying the angular deviation or tilt (skew) that text may exhibit within scanned or photographed documents.

4.4 Models/Algorithms

4.4.1 Optical Character Recognition

Optical Character Recognition (OCR) is a technology that enables computers to recognize and extract text from images, scanned documents, or other sources of text in digital form. OCR algorithms typically involve several steps to process images and identify text regions, followed by the recognition and interpretation of the extracted text.

Here's a basic overview of the OCR algorithm:

1.Image pre processing

The input image is preprocessed to enhance its quality and improve the accuracy of text extraction. Preprocessing steps may include noise reduction, contrast enhancement, binarization (converting the image to black and white), and deskewing (correcting for any rotation or skew in the image).

2. Text Detection

In this step, the algorithm identifies regions of the image that contain text. Various techniques can be used for text detection, such as sliding window approaches, connected component analysis, or deep learning-based object detection algorithms like EAST (Efficient and Accurate Scene Text Detector) or CRAFT (Character Region Awareness for Text Detection). The result is a set of bounding boxes or contours that enclose text regions in the image.

3. Segmentation

Once text regions are detected, the algorithm may refine the localization of individual characters or words within these regions. This step involves segmenting the text regions into smaller components corresponding to individual characters or words, typically using techniques like contour analysis, morphology operations, or connected component labeling.

4. Feature Extraction

Extracting distinctive attributes such as shape, size, texture, and intensity from the segmented characters or words, which are crucial for recognizing and distinguishing them accurately.

5. Character Recognition

With text regions localized, the algorithm proceeds to recognize and interpret the characters contained within these regions. Various approaches can be used for character recognition, including traditional pattern matching techniques, feature-based methods like Histogram of Oriented Gradients (HOG) or Scale-Invariant Feature Transform (SIFT), and deep learning-based models such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or Transformer-based architectures.

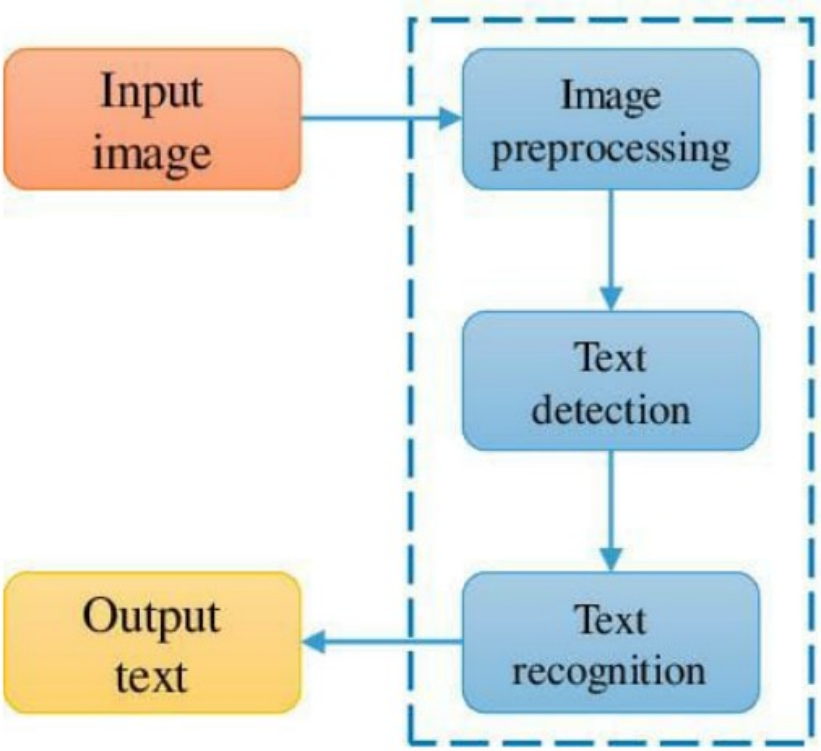

Figure 2: OCR block diagram

6. Post-processing

After character recognition, the OCR output may undergo post-processing steps to improve accuracy and correct errors. This may involve techniques such as language modeling, spell checking, context-based correction, or dictionary look-up to refine the OCR results and ensure the extracted text is accurate and coherent.

7. Output Generation

Finally, the recognized text is outputted in a usable format, such as plain text, searchable PDF, or structured data. The OCR algorithm may also provide additional metadata, such as confidence scores for each recognized character or word, to assess the reliability of the OCR output.

Overall, OCR algorithms play a crucial role in digitizing and extracting text from images, enabling a wide range of applications in document processing, content indexing, information retrieval, and more.

4.4.2 Text To Speech Algorithm

Text-to-Speech (TTS) algorithms are fundamental for converting written text into spoken words, enabling computers and devices to produce natural-sounding speech [10]. These algorithms play a pivotal role in diverse applications, such as accessibility tools for travellers, language learning programs, and automated customer service systems. A crucial aspect of TTS algorithms is their capability to analyze and synthesize text, replicating human speech patterns and intonations effectively. This empowers travellers and other users to access information audibly, enhancing their overall experience and ensuring inclusivity.

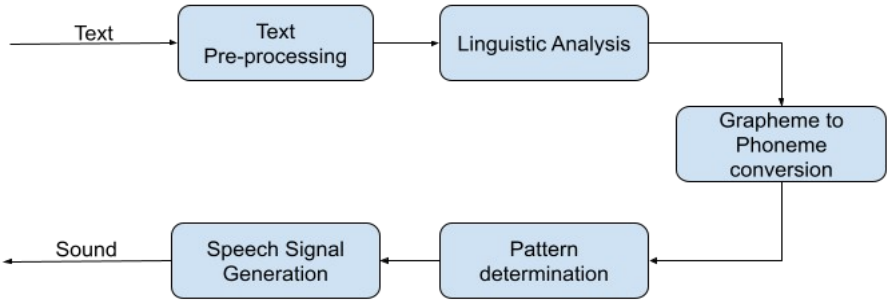

Figure 3: Text to Speech generation

1.Text

This is the input text provided by the user, which could be any sequence of words or sentences that need to be converted into speech.

2.Text Pre processing

The input text undergoes pre-processing to clean up any unwanted characters, normalize punctuation, and handle special cases like abbreviations or acronyms. This ensures that the text is in a suitable format for further analysis.

3.Linguistic Analysis

The preprocessed text is analyzed linguistically to understand the grammatical structure, word meanings, and linguistic rules such as pronunciation, stress, and intonation patterns. This analysis helps in generating speech that sounds natural and coherent.

4.Grapheme to Phoneme Conversion

This step involves converting the written representation of words (graphemes) into their corresponding spoken sounds (phonemes). Each word is analyzed phonetically to determine its pronunciation based on linguistic rules and dictionaries.

5.Pattern Determination

Once the phonetic representation of words is obtained, patterns for speech synthesis are determined. This involves selecting appropriate speech synthesis techniques, such as concatenative synthesis (joining pre-recorded speech segments) or parametric synthesis (generating speech waveform from acoustic parameters).

6.Speech Signal Generation

Using the determined patterns and phonetic information, the speech signal is generated. This involves synthesizing the speech waveform by either concatenating prerecorded speech segments or generating it from scratch based on acoustic parameters like pitch, duration, and amplitude.

7.Sound

Finally, the generated speech signal is converted into audible sound waves, which can be played through speakers or headphones. This sound represents the synthesized speech corresponding to the input text, allowing users to listen to the converted speech output. Overall, these steps in the TTS algorithm with gTTS ensure that the input text is accurately converted into natural-sounding speech, providing an effective means of communication for various applications.

5. Methodology

5.1 Methodology

This methodology outlines the steps involved in enhancing accessibility through image recognition, OCR, and audio description generation for visually challenged individuals. It emphasizes using advanced technologies while ensuring the system is user-friendly and effective in providing meaningful auditory feedback based on real-time image analysis.

1. **Data Collection:**

The project leverages pretrained models such as OCR (using Tesseract), Google Translate (googletrans), and text-to-speech synthesis (GTTS). While no training data is needed for these models during runtime, specific language data for OCR must be stored locally on the system. These include language-specific traineddata files such as "ell.traineddata" for Greek, "ara.traineddata" for Arabic and "chi_tra.traineddata" for Traditional Chinese. These traineddata files enable Tesseract OCR to recognize and process text in various languages effectively. The application accepts real-time images as input, processes them using these pretrained models, and displays the extracted text, ensuring accurate and language-appropriate OCR functionality based on the selected language model.

2. **Data preprocessing:**

The input image is processed using OpenCV, an image processing library, to optimize it for subsequent analysis. This involves several critical steps to enhance image quality and prepare it for accurate text extraction. Firstly, the image undergoes binarization to convert it into a binary format, distinguishing text from background for improved OCR performance. Secondly, the image is resized to a standardized format suitable for processing, ensuring consistency and efficiency. Noise reduction techniques are then applied. These preprocessing steps ensure that the image is properly prepared for effective text extraction and further analysis within the project's workflow.

3. **Text extraction using Tesseract:**

After the user uploads an image and selects the language, the next step involves passing the uploaded image to Tesseract OCR for text extraction. Tesseract, integrated through Pytesseract, processes the image to identify and extract text based on the selected language. This step leverages Tesseract's powerful optical character recognition capabilities to convert the textual content within the image into machine-readable text format.

4. **Language detection:**

The next step involves automatically detecting the language of the extracted text using the Google Translate API provided by googletrans. This API identifies the language of the extracted text based on its linguistic characteristics and context.

5. **Language translation:**

After detecting the language of the extracted text, the user selects their desired target language from a dropdown box. The project then utilizes the Google Translate API (googletrans) to translate the extracted text from its detected language into the user-selected language. This API facilitates accurate and efficient translation across multiple languages, ensuring that the content is accessible and understandable to users in their preferred language.

5.2 System Architecture

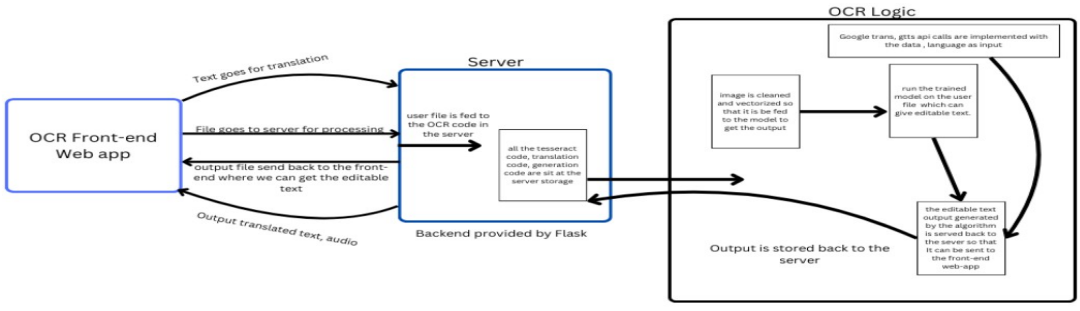

Figure 4: System architecture

5.3 Data Sets

Tesseract is capable of extracting text in over 100 languages, but it requires specific trained data for active extraction. The trained data allows Tesseract to compare extracted characters with pre-trained language models, ensuring accurate text recognition. By using the appropriate trained data files, Tesseract can effectively recognize and extract text from images, providing accurate and readable text output.Some of those traindatasets are,

ell.traineddata: Greek language trained data.

ara.traineddata: Arabic language trained data.

rus.traineddata: Russian language trained data.

kor.traineddata: Korean language trained data.

jpn.traineddata: Japanese language trained data.

chi_tra.traineddata: Traditional Chinese language trained data.

chi_sim.traineddata: Simplified Chinese language trained data.

por.traineddata: Portuguese language trained data.

5.4 Preprocessing

Preprocessing steps for image captioning:

OpenCV provides predefined functions to perform preprocessing. We are using the following functions to perform the same:

Grayscale Image:

Apply a grayscale filter to the image so the image has colors black and white only.

cv2.cvtColor(img_name,cv2.COLOR_BG R2GRAY)

Noise Removal:

Remove the noise and smoothen the image for further processing.

cv2.medianBlur(img_name,5)

Thresholding:

Apply threshold to the image and create a binary image.

cv2.threshold(img_name, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

Dilation:

Add boundaries to the objects in the image, so the text stands out.

processor_var = np.ones((5,5),np.uint8)

cv2.dilate(img_name, processor_var, iterations = 3)

Erosion:

Remove the boundaries after the dilation in the image, to unify the objects.

processor_var = np.ones((5,5),np.uint8)

cv2.erode(img_name, processor_var, iterations = 3)

5.5 Feature Extraction

1.Feature extraction in Tesseract:

In Tesseract, feature extraction involves identifying and isolating text characters from an image. The OCR engine processes the image by analyzing its pixel patterns to detect text regions. It then segments these regions into individual characters or words, leveraging trained data for the specific language to compare and recognize these segments accurately. The extracted text is refined through a series of algorithms that handle noise reduction, binarization, and other image preprocessing techniques, ensuring that the final output is a coherent and readable text string. This step is crucial for converting visual text data into machine-readable text.

2.Feature extraction in Google Translate:

Feature extraction in Google Translate (googletrans) involves analyzing the text to identify its linguistic features and context. The translation model processes the input text by breaking it down into smaller units such as words, phrases, and syntactic structures. It then uses these features to

understand the meaning and context of the text. By leveraging a vast database of bilingual text corpora and advanced machine learning algorithms, googletrans can accurately map these linguistic features from the source language to the target language, ensuring the translated text maintains the original meaning and context. This process is essential for delivering accurate and contextually appropriate translations.

3.Feature extraction in Google Text-To-Speech(GTTS):

In Google Text-to-Speech (GTTS), feature extraction involves converting text into phonetic and prosodic elements that can be synthesized into natural-sounding speech. The system analyzes the input text to determine its phonetic structure, including individual sounds and intonation patterns. It also identifies punctuation and other textual cues that influence speech rhythm and emphasis. By extracting these features, GTTS can generate audio output that accurately reflects the intended pronunciation, tone, and cadence of the text. This process is critical for producing clear and intelligible speech that effectively communicates the content of the text to users.

6. Implementation

6.1 Technologies used :

Following are the different technologies used to build the project:

Flask : Flask is a lightweight WSGI web application framework for Python. It provides the tools, libraries, and technologies needed to build a web application, allowing developers to quickly and efficiently create web interfaces for their projects.

HTML: HTML (Hyper Text Markup Language) is the markup language, which provides the meaning and structure of web content. It is used in the frontend of our project together with CSS and JavaScript.

CSS: Cascading Style Sheet language is the language (style sheet) that is used to describe the look and feel or presentation of the HTML documents.

JavaScript: JavaScript is a powerful and lightweight programming language that is used on both client and server side. It helps to create interactive web pages to manipulate, validate and calculate the data.

Python: Following Python libraries are used in our project:

- **OpenCV**: OpenCV is a library for different programming functions primarily aimed to provide real-time computer vision. It is used to import and perform segmentation of the image as well as extract the image in our project.
- **Flask-Dropzone :** Integrates Dropzone.js into Flask for easy file uploads.
- **Flask-Session :** Adds server-side session capabilities to Flask applications.
- **Flask-WTF :** Integrates WTForms with Flask for form rendering and validation.

- **Tesseract**: Tesseract is a library, which provides Optical Character Recognition engine with support for the Unicode and has the ability to recognize more than 100 languages in-built. It can also be trained to recognize other languages as well.
- **Googletrans :** Googletrans is a free and unlimited Python library that interfaces with the Google Translate API. It allows developers to translate text between multiple languages effortlessly. This library is crucial for handling language translation in your project.
- **gTTS :** Google Text-to-Speech (gTTS) is a Python library and CLI tool that converts text into spoken audio using Google Translate's text-to-speech API. It generates natural-sounding speech, enhancing the accessibility of text content. This tool is essential for providing audio feedback to users in your project.
- **Pillow :** A Python Imaging Library (PIL) fork for opening, manipulating, and saving image files.
- **Playsound :** Pure Python, cross platform, single function module with no dependencies for playing sounds.
- **Pytesseract :** A Python wrapper for Google's Tesseract-OCR Engine.
- **Requests :** Simple HTTP library for Python, used for making HTTP requests.
- **WTForms :** A flexible forms validation and rendering library for Python web development.

These libraries can be imported by using the following code :

```
from flask import Flask
import os
from flask_dropzone import Dropzone
from flask_session import Session
from wtforms import TextAreaField, SubmitField, SelectField
from flask_wtf import FlaskForm
from wtforms.validators import DataRequired, Length
from __init__ import app, dropzone
```

```
from flask import render_template, request, redirect, url_for, session

from forms import filed_data

import secrets

import os

import cv2

import pytesseract

from PIL import Image

import numpy as np

from gtts import gTTS

from googletrans import Translator
```

6.2 Working diagram :

In this section, we will discuss how the project actually works with the help of some code snippets. So we will have an idea for the operation of the WebApp.

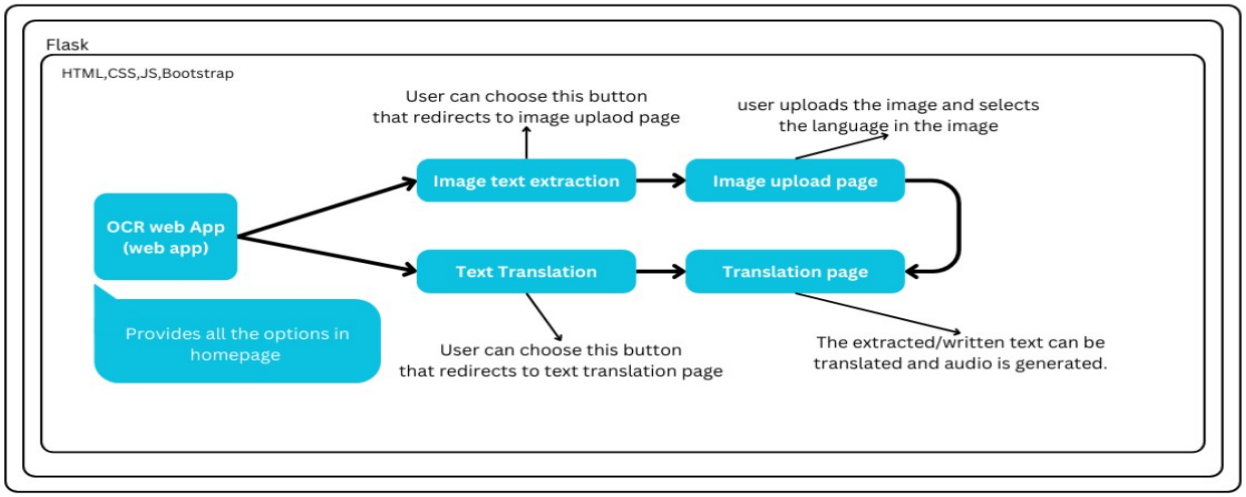

Figure 5: Working diagram

6.3 WorkFlow Representation :

The below flowchart represents the workflow of the project.

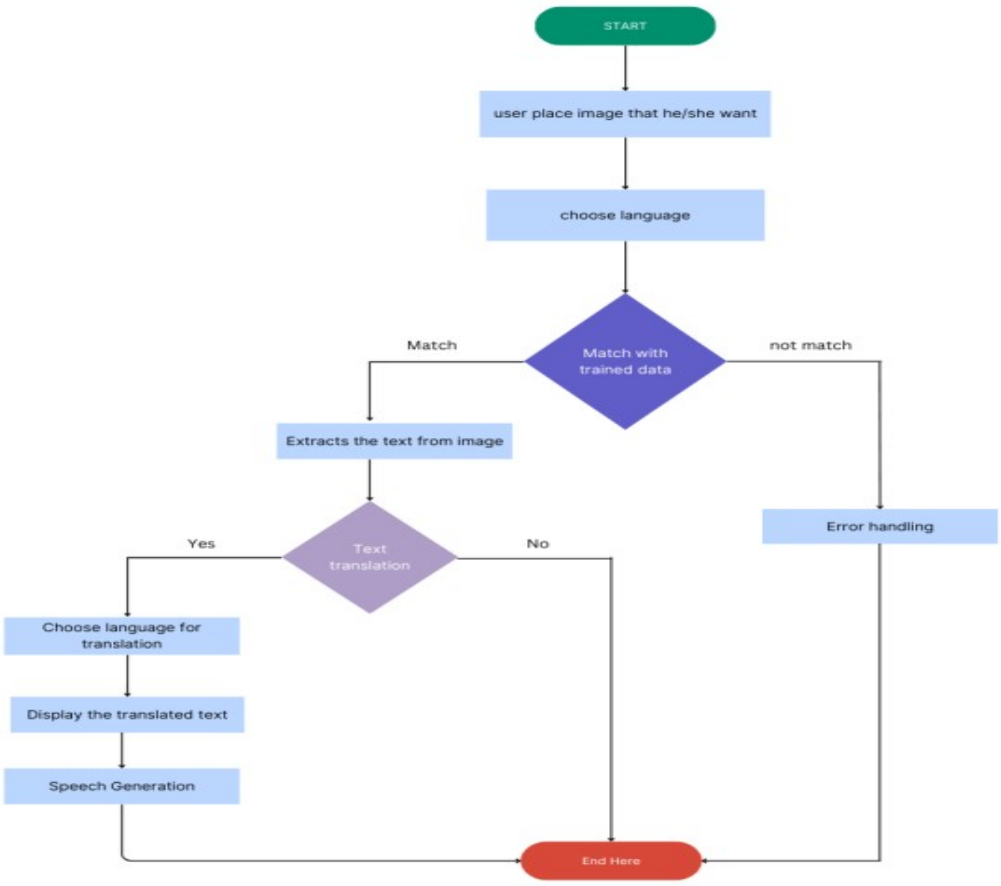

Figure 6: Workflow diagram

6.4 Usecase Representation :

The below diagram represents the set of actions that users can perform through this project.

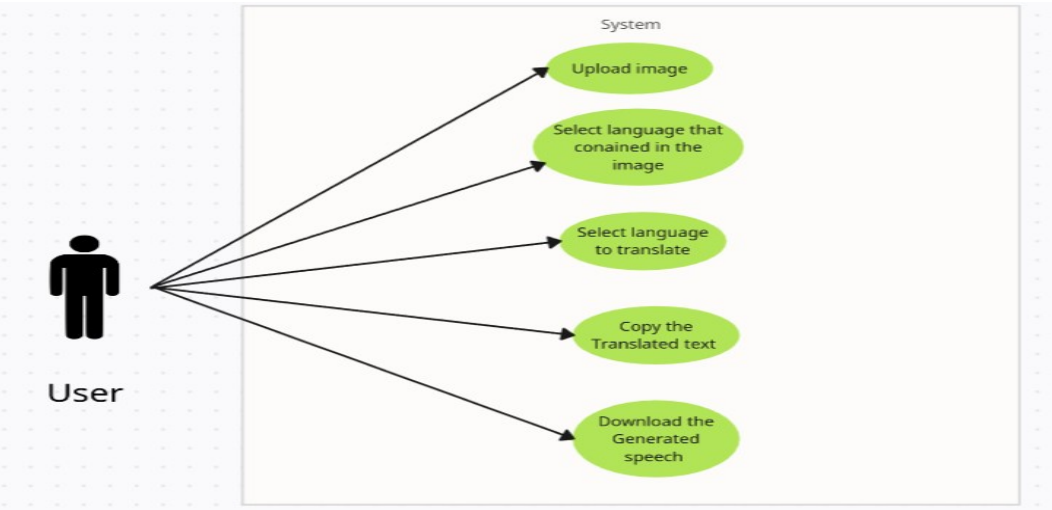

Figure 7: Use case diagram

When the user enters the homepage, they are presented with two options:

1. Image to Text: By clicking this option, the user is directed to the uploads page where they can upload the images from which they want to extract text.

2.Text Translation: By selecting this option, the user is taken to the text translation page where they can enter or paste the text they wish to translate.

Image to Text :

Here the user directed to an upload page:

- The user uploads the image that he/she wants to extract the text from.
- The user selects the language contained in the image, allowing Tesseract to use the appropriate trained dataset for character comparison and output generation.

After uploading , the user will redirect to the display page, where the extracted text will be displayed.

- The uploaded data was detected.
- The text extracted by Tesseract is displayed in an editable format, enabling users to interact with it directly. This functionality allows users to easily copy the text for other uses or make any necessary modifications right within the application.

The extracted text can be translated into the user's preferred language using Google Translate (googletrans). This translation feature broadens the accessibility of the text, making it useful for a diverse audience. Additionally, the translated text can be converted into audio format using Google Text-to-Speech (gTTS), providing an auditory representation of the text. This audio generation feature is particularly beneficial for visually impaired users.

Text Translation :

Here, the user is directly redirected to the display page with a blank text area, enabling them to write or paste the text they want to translate.The user selects the language they want to translate the text to. The output includes:

- Detection of the input language
- Editable translated text
- Translated audio generation

6.5 Working of tesseract:

The following code describes the implementation and working of the Tesseract OCR (Optical Character Recognition) engine used to extract text from images in our project.

```
@app.route("/upload", methods=["GET", "POST"])
def upload():
  if request.method == 'POST':
      selected_language = request.form.get('language')
      sentence = ""

      f = request.files.get('file')
      filename, extension = f.filename.split(".")
      generated_filename = secrets.token_hex(10) + f".{extension}"

      app.config['UPLOADED_PATH'] = 'images'
      file_location = os.path.join(app.config['UPLOADED_PATH'],
generated_filename)

      f.save(file_location)
      pytesseract.pytesseract.tesseract_cmd = r'/usr/bin/tesseract'

      img = cv2.imread(file_location)
      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

      boxes = pytesseract.image_to_data(img, lang=selected_language)


      for i, box in enumerate(boxes.splitlines()):
          if i == 0:
              continue
          box = box.split()
          if len(box) == 12:
              sentence += box[11] + " "
      session["sentence"] = sentence
```

```
    os.remove(file_location)

    return redirect("/decoded")
else:
    return render_template("upload.html", title="Home")
```

1.**Preprocessing**:

The uploaded image undergoes preprocessing to prepare it for text extraction.

Reading and Converting Image: The image is read using OpenCV and converted from BGR to RGB color space. OpenCV provides predefined functions to perform preprocessing.

img = cv2.imread(file_location)

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

2.Text Extraction with Tesseract:

The Tesseract OCR engine is used to extract text from the preprocessed image.

The image_to_data function of Tesseract is used to extract detailed text data, including bounding box coordinates and confidence levels.

pytesseract.pytesseract.tesseract_cmd = r'/usr/bin/tesseract'

boxes = pytesseract.image_to_data(img, lang=selected_language)

3.Processing Extracted Text:

The extracted text data is processed to form a complete sentence.

The output from image_to_data is processed line by line. The first line is skipped as it contains the header.

The formed sentence is stored in the session for later use.

for i, box in enumerate(boxes.splitlines()):

```
    if i == 0:
    continue
box = box.split()
if len(box) == 12:
sentence += box[11] + " "
session["sentence"] = sentence
```

4.Redirecting:

After processing, the user is redirected to another route (/decoded) to handle the next steps of the workflow, such as translation and text-to-speech conversion.

```
return redirect("/decoded")
```

6.6 Working of googletrans and gtts:

The following section describes the implementation and working of the Google Translate API used for language detection and translation in our project.

1.Language detection :

We used the Google Translate API to detect the language of the extracted text.

```
from cv2 import triangulatePoints
from googletrans import Translator

translator = Translator()

def detect_language(text):
  detected_lang_data = translator.detect(text)
  lang = detected_lang_data.lang
  conf = detected_lang_data.confidence

  return lang, conf

def translate_text(text, dest):
  translated_text = translator.translate(text, dest=dest)
  return translated_text.text
```

- An instance of the Translator class is created.
- The detect method of the Translator instance is used to identify the language of the provided text. It returns an object containing the detected language and the confidence level.
- The translate method of the Translator instance is used to translate the provided text into the specified target language. It returns an object containing the translated text.

2.Language Translation and speech generation:

We used googletrans to translate language and gtts to generate audio.

1.Translation :

The following code is useful for the translation of text into the defined language.

```
def translate_text(text, dest):
    translated_text = translator.translate(text, dest=dest)
    return translated_text.text
```

The dest is a string that holds the destination language, which is selected by the user from the languages list in utils.py. This function uses the Google Translate API to translate the input text into the specified dest language and returns the translated text.

2.Speech generation :

Below code demonstrates the working of the gtts for speech generation.

```
tts = gTTS(translated_text, lang=translate_to)
    file_location = os.path.join(app.config['AUDIO_FILE_UPLOAD'],
    generated_audio_filename) tts.save(file_location)
```

The process described above utilizes gTTS where the translated text is provided as input along with the target language for translation. gTTS then generates synthetic audio based on this input. The generated audio is saved and also streamed to the frontend for the user to listen to.

Loading page :

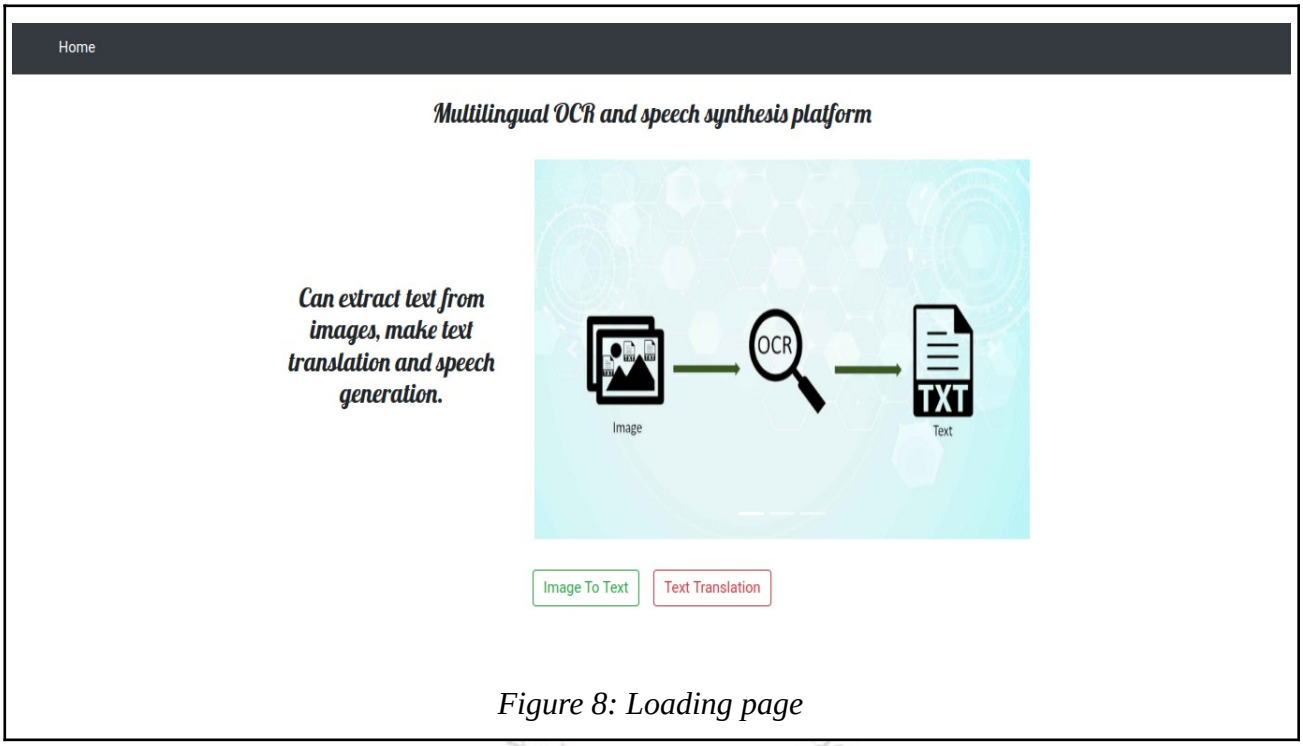

Figure 8: Loading page

Upload image:

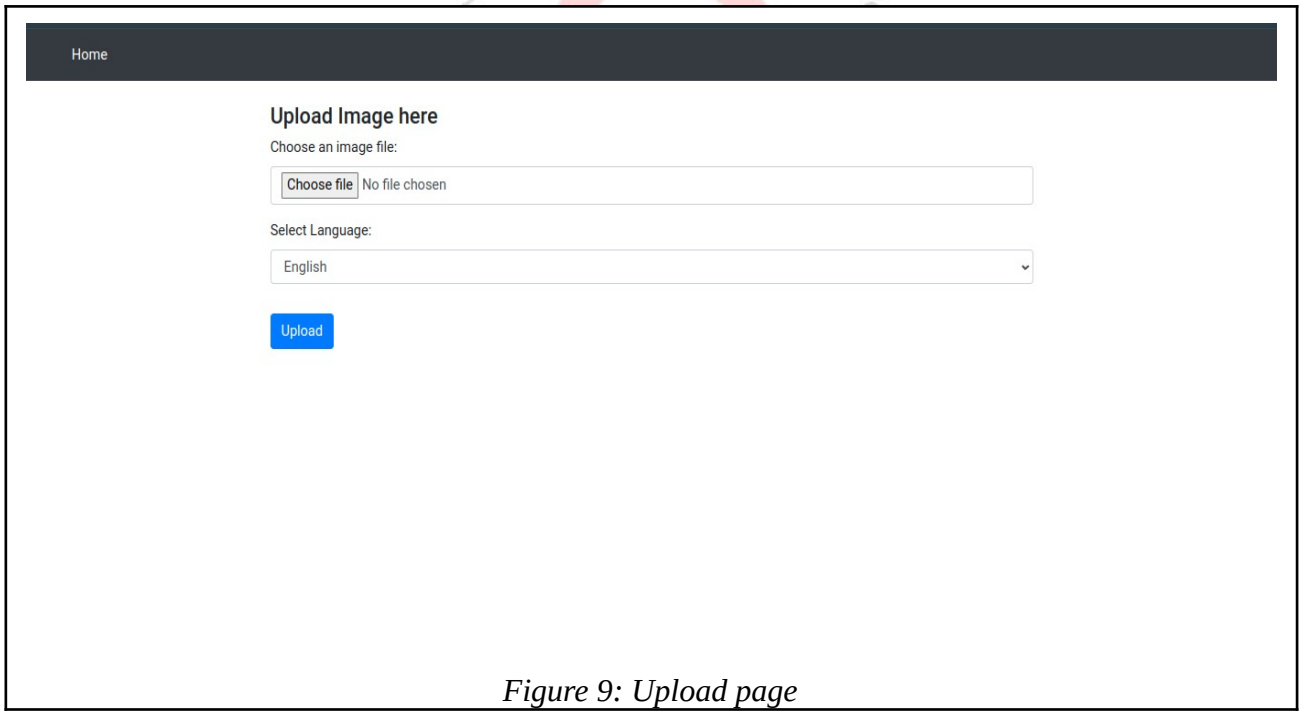

Figure 9: Upload page

Display page:

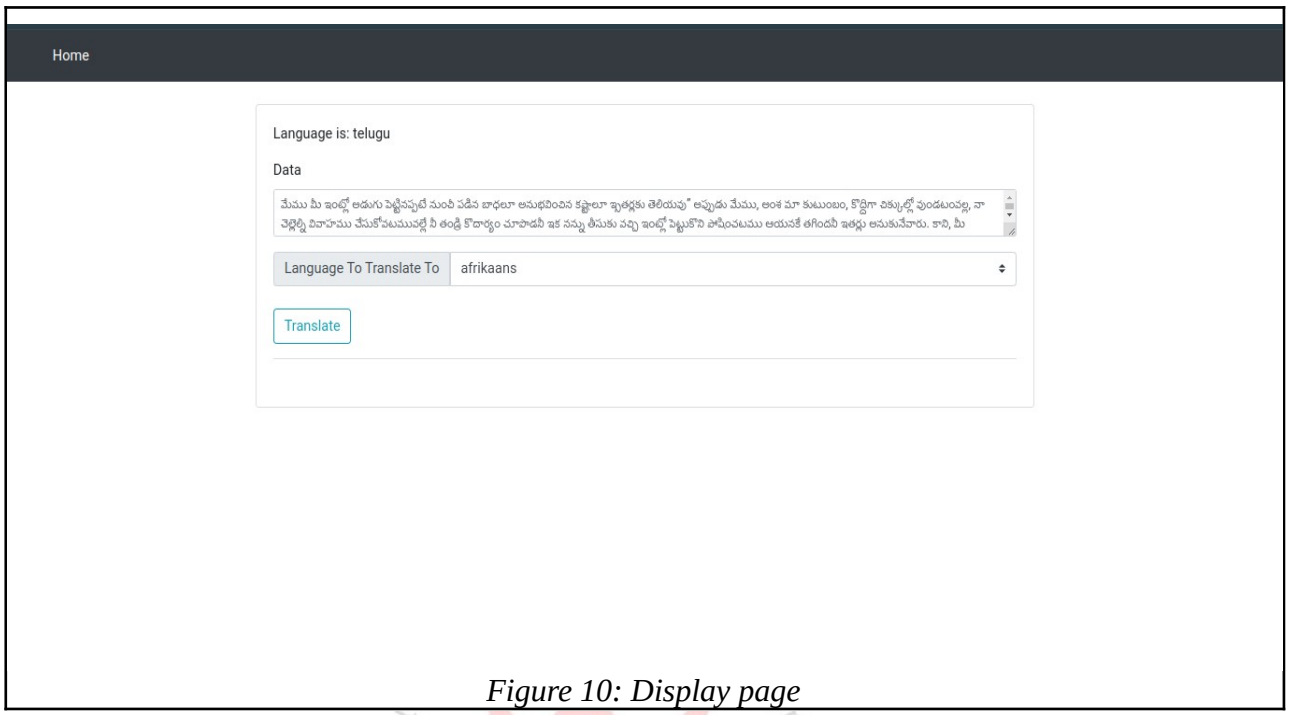

Figure 10: Display page

Transited text and audio generation:

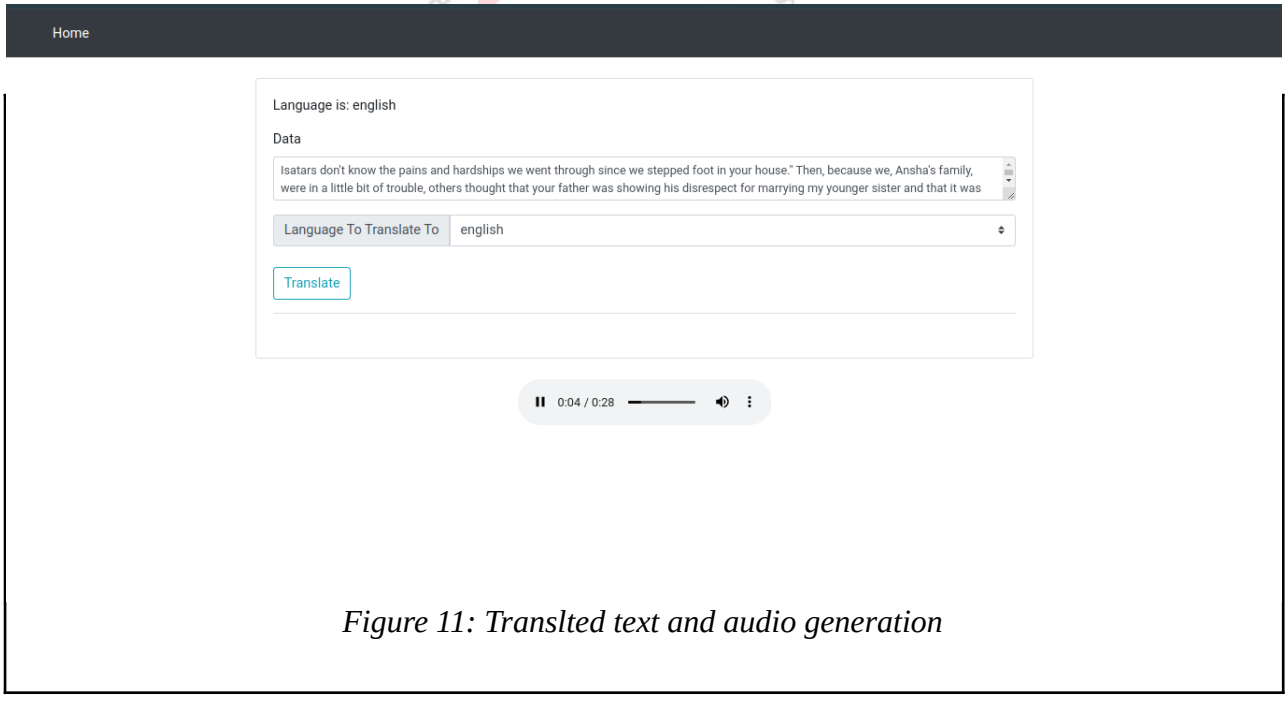

Figure 11: Translted text and audio generation

Text translation:

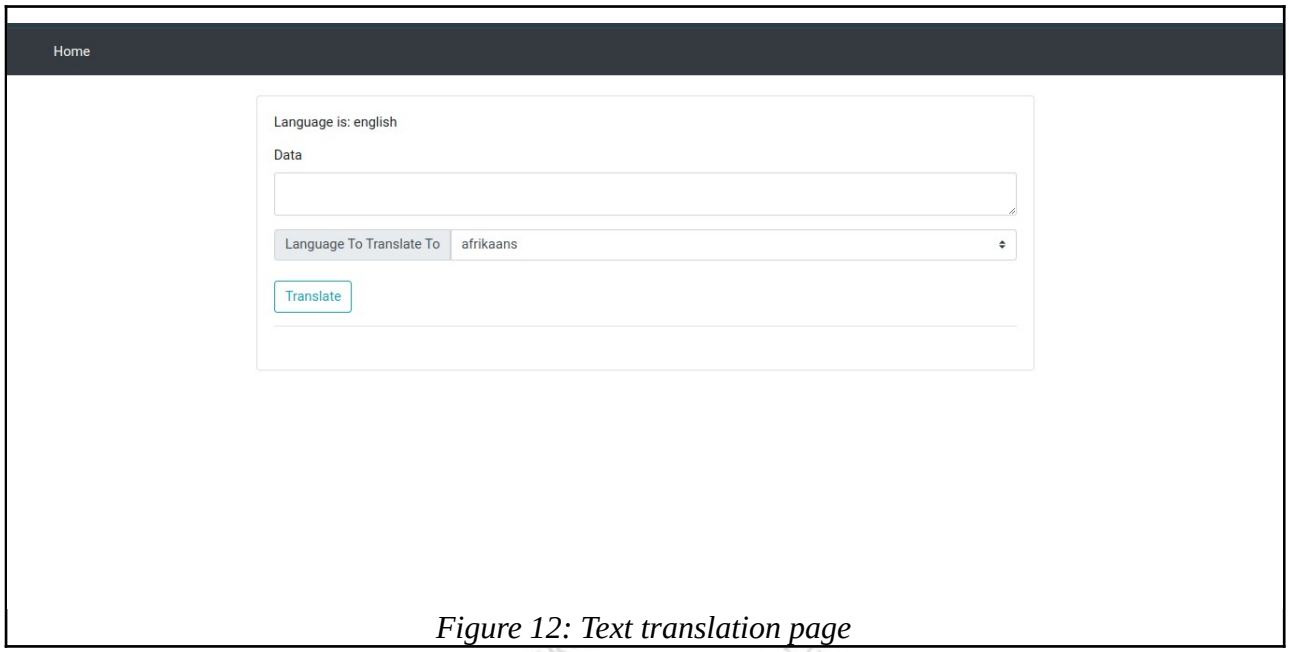

Figure 12: Text translation page

7. Conclusion

The project aimed to develop a comprehensive system for recognizing text from images, translating it into multiple languages, and converting it to speech.This system serves as a valuable tool for both students and individuals, offering enhanced accessibility and usability in accessing and comprehending textual content.

The project utilized advanced technologies such as Optical Character Recognition (OCR), language translation APIs, and text-to-speech synthesis to achieve its objectives.By leveraging OCR algorithms, text could be accurately extracted from images, enabling users to access information from various sources such as circulars, notices, and books.The integration of language translation APIs facilitated the seamless translation of text into multiple languages, catering to diverse linguistic needs and preferences.

8. Future Work

a. Enhanced Handwriting Recognition:

Improving the ability of the system to accurately understand and convert handwritten text by using advanced neural networks.

b. Expanded Offline Capabilities:

Allowing the system to perform language translation even without an internet connection, enhancing accessibility in diverse environments.

9. References

1: V. Yadav, N. Ragot, Text Extraction in Document Images: Highlight on Using Corner Points , 2016
2: Asif, A. M. A. M., et al., An overview and applications of optical character recognition , 2014
3: Xue, W., Q. Li, Q. Xue, Text detection and recognition for images of medical laboratory reports with a deep learning approach , 2019
4: Almeida, A. M. P., et al., Development of an online digital resource accessible for students with visual impairment or blindness, 2020
5: Piotrowski, M. , Natural language processing for historical texts, 2012
6: Sisodia, P., S. W. A. Rizvi , Optical character recognition development using Python , 2023
7: Kamaraj, S. A., et al. , Enhancing Automatic Speech Recognition and Speech Translation Using Google Translate , 2023
8: Tensmeyer, C., T. Martinez , Historical document image binarization: A review , 2020
9: Nachtegael, M., et al. , Fuzzy filters for noise reduction: The case of Gaussian noise , 2005
10: Dutoit, T. , An introduction to text-to-speech synthesis, 1997