MINI-PROJECT REPORT ON

OCR

Optical Character Recognition to Text And Audio

Submitted in partial completion of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

Student Name: Kajal Rawat **University Roll No.:2118668**

Under the Mentorship of

Ms. Preeti Chaudhary

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

Graphic Era Hill University
DEHRADUN • BHIMTAL • HALDWANI

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project report entitled **"OCR-Optical Character Recognition to Text And Audio"** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering of the Graphic Era Hill University, Dehradun shall be carried out by myself under the mentorship of **Ms. Preeti Chaudhary, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era Hill University, Dehradun Campus.

Name: Kajal Rawat **University Roll no.: 2118668**

Table of Contents

Chapter 1

INTRODUCTION

"Embracing AI and ML for Enhanced Information Accessibility" is about using smart computer technology to make information easier to get. It's like having clever tools that can read and understand things like text in pictures or documents. This technology helps take messy information and turns it into something computers can understand. By using these smart tools, it becomes easier for everyone to understand and use information, even if they speak different languages or have different ways of getting information.

Our project is dedicated to leveraging AI and ML technologies to significantly enhance information accessibility and usability across diverse user scenarios. Through 'AI in Voice Technology,' we aim to enable seamless voice-based interactions, breaking language barriers, and fostering improved information access through spoken language. This facet of our project focuses on:

- Facilitating voice-based communication and interactions
- Overcoming language barriers
- Enhancing information accessibility via spoken language

Concurrently, 'Machine Learning for Human Assistance' plays a pivotal role in our project. By harnessing technologies like Optical Character Recognition (OCR) and other ML tools, it aims to:

- Assist in efficiently processing and extracting information
- Improve data accessibility and understandability

These integrated applications align with our overarching goal of rendering information more user-friendly and accessible, benefiting a diverse range of users in their interaction with various forms of data. By leveraging AI and ML advancements, our project strives to make information universally accessible and comprehensible."

Chapter 2

LITERATURE REVIEW

Our project implementation involves a fusion of AI and ML technologies, notably employing Optical Character Recognition (OCR), Natural Language Processing (NLP), and Voice Technology (gTTS) to enhance information accessibility. This implementation aligns with existing research, where similar AI-driven approaches have been utilized:

- **Efficient Data Extraction:** Through OCR and image processing, our approach echoes established practices, enabling precise text extraction from diverse sources like images and documents.
- **Multilingual Accessibility:** Leveraging translation tools and gTTS aligns with research emphasizing the importance of multilingual access, catering to users who prefer content in their native language.
- **Personalized Interaction:** Implementing ML-driven personalized content mirrors findings emphasizing the significance of tailored recommendations, enhancing user engagement and satisfaction.
- **Enhanced User Experience:** Our project adheres to accessible UI design principles, catering to varying user needs and conforming to established standards.
- **Improved Efficiency:** Utilizing NLP techniques and automated summarization aligns with research advocating for quicker comprehension and navigation through large volumes of data, thereby enhancing overall efficiency in information processing.
- **Inclusivity and Fairness:** The project's emphasis on ethical AI practices, avoiding biases, and ensuring fairness in data processing resonates with established literature stressing the importance of unbiased algorithms for inclusive accessibility.

By integrating these findings into our project implementation, we aim to achieve tangible benefits such as enhanced data accessibility, improved user experiences, and efficient information processing, ultimately aligning with the broader goal of making information universally accessible and comprehensible.

Chapter 3

METHODOLOGY

The OCR system includes the following steps:

- **User Interaction:**
  - Using HTML, CSS, Flask.
  - Users upload PDF or image files through an intuitive interface (as shown in fig.3).
  - File types are validated for PDFs or images.
- **File Processing and Text Extraction:**
  - Receive uploaded files and save them to a designated location.
  - Check the file type (PDF or image) and proceed accordingly.
  - For PDFs:
  - Utilize PyMuPDF (fitz) to load the PDF and extract text from text-based elements on each page.
  - Extract text from images within the PDF using Pytesseract OCR.
  - For images:
  - Use OpenCV (cv2) to preprocess the image (grayscale, thresholding, noise removal).
  - Apply Pytesseract OCR to extract text from the preprocessed image

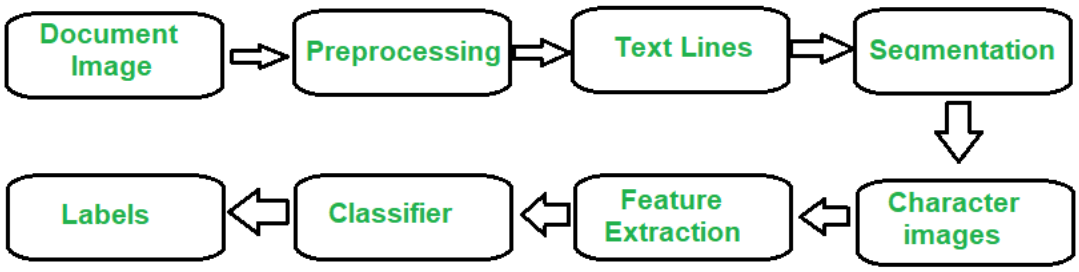


Fig.1: Flow Diagram of OCR

- **Translation and Audio Generation:**
  - Enable user input for text translation.
  - Utilize gTTS for language translation and convert translated text into MP4 audio.
- **Output Management:**
  - Store translated text and audio in TXT and MP4 formats, respectively.
  - Manage file storage using os module and provide download links (as shown in Fig.8).
- **Error Handling and Feedback:**
  - Implement error handling for invalid file formats and processing errors.
  - Provide user feedback on successful translations and error notifications.
- **Optimization and Testing:**
  - Continuously refine image processing and text extraction techniques.
  - Ensure accuracy and efficiency in translation and text-to-speech conversion.
- **Deployment:**
  - Test functionalities across browsers and devices.
  - Deploy the functional web application for public access.

PesudoCodes:

- Below is a basic pseudocode representation of image preprocessing:

```
function preprocess(image):
    # Convert the image to grayscale
    grayscale_image = convert_to_grayscale(image)

    # Apply thresholding to the grayscale image
    thresholded_image = apply_thresholding(grayscale_image)

# Additional preprocessing steps (if needed)
    processed_image = additional_processing(thresholded_image)

    return processed_image

function convert_to_grayscale(image):
    # Convert the input image to grayscale
    grayscale_image = convert_to_grayscale_algorithm(image)

    return grayscale_image

function apply_thresholding(image):
    # Apply thresholding technique to the image
    thresholded_image = apply_thresholding_algorithm(image)

    return thresholded_image
```

- Below is a basic pseudocode representation of an OCR algorithm:

```
function OCR_algorithm(preprocessed_image):
    # Initialize variables to store extracted text
    extracted_text = ""

    # Define OCR processing steps
    for each text_region in preprocessed_image:
        # Apply OCR on each text region within the preprocessed image
        text = apply_ocr_on_region(text_region)

        # Append extracted text from the region to the overall text
        extracted_text += text + " "

    return extracted text

function apply_ocr_on_region(text_region):
    # Call OCR function to extract text from the given text region
    extracted_text = OCR_function(text_region)

    return extracted_text
```

Chapter 4

RESULT

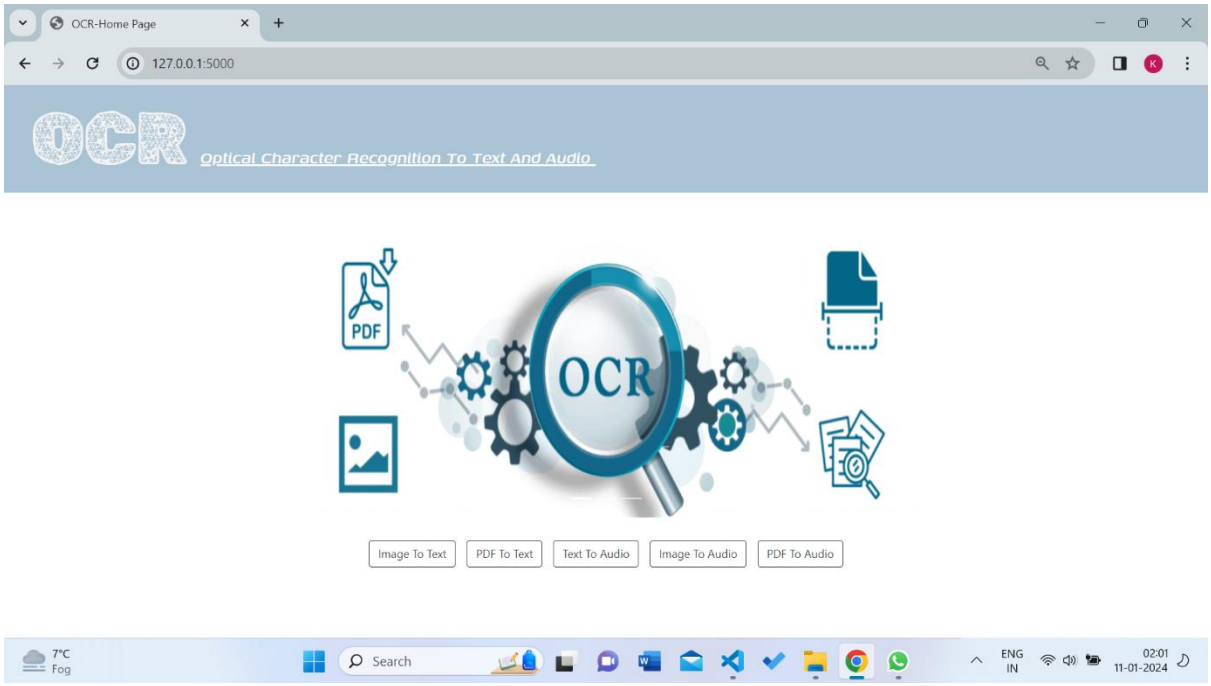


Fig.2: User Interface

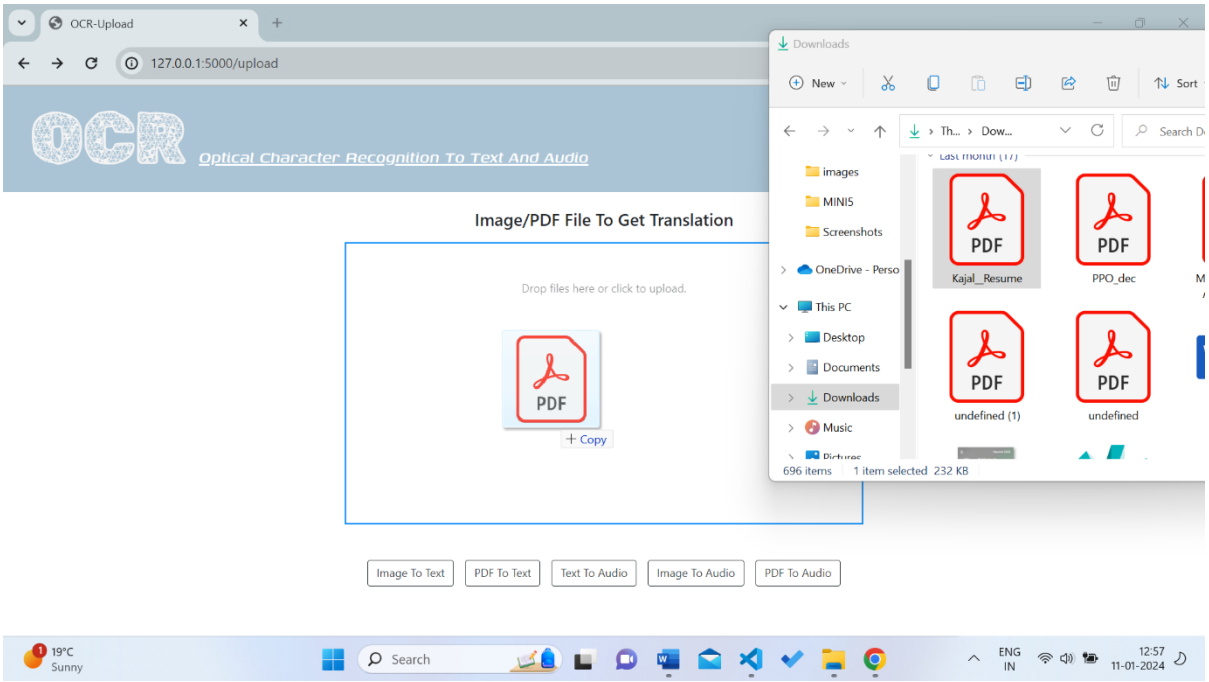


Fig.3: Files Uploading

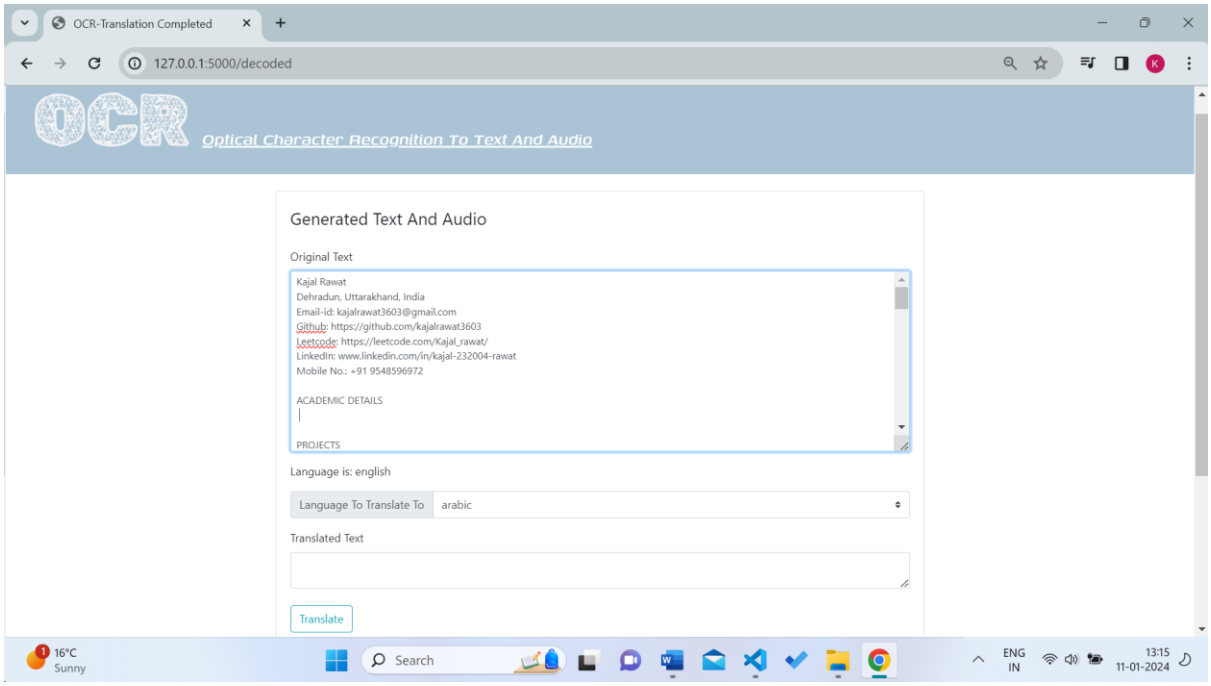


Fig.4: Extracted Text

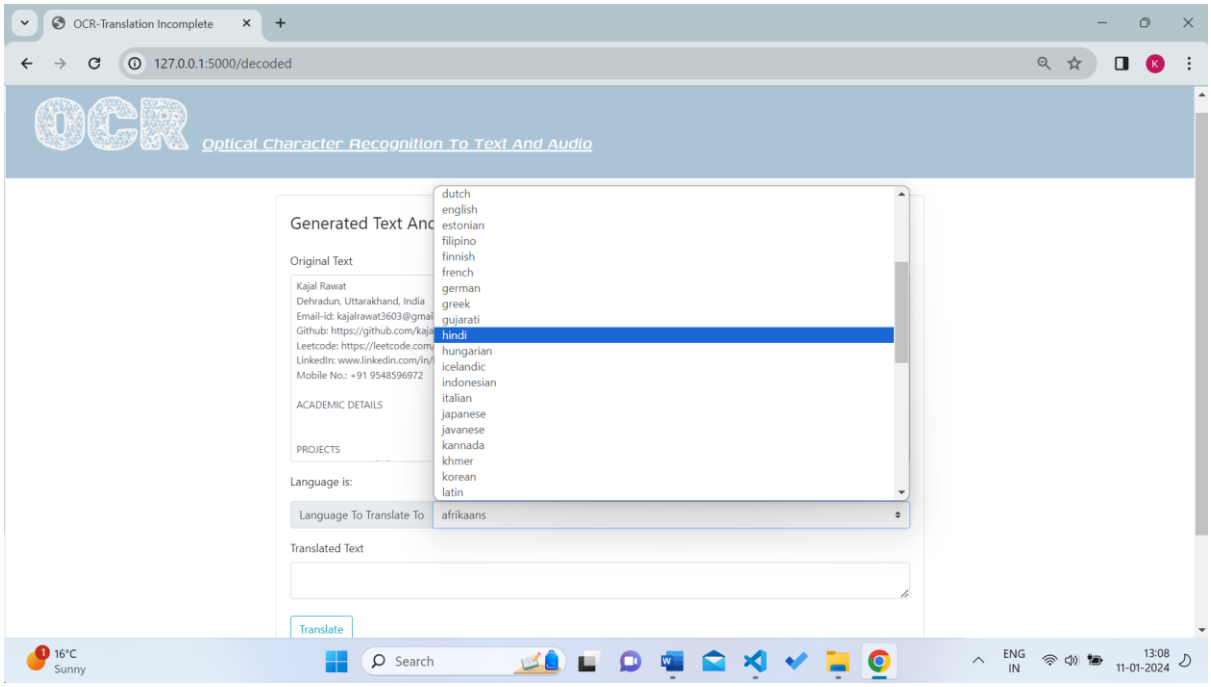


Fig.5: language Selection

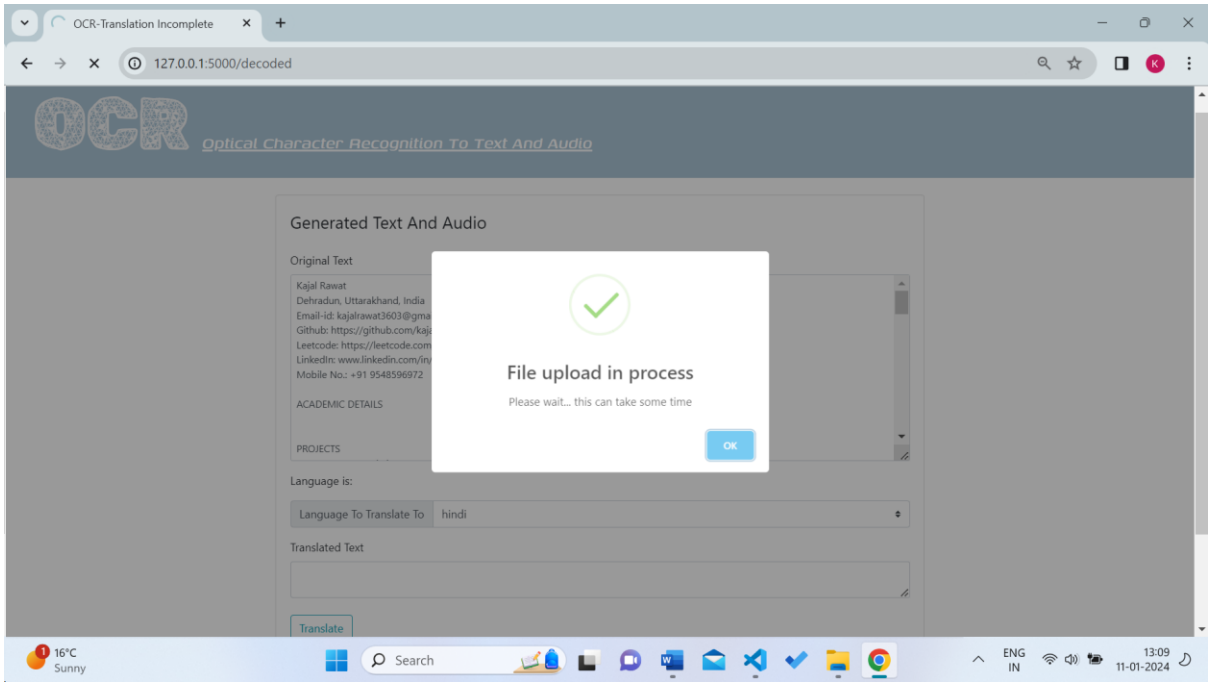


Fig.6: Translation Phase

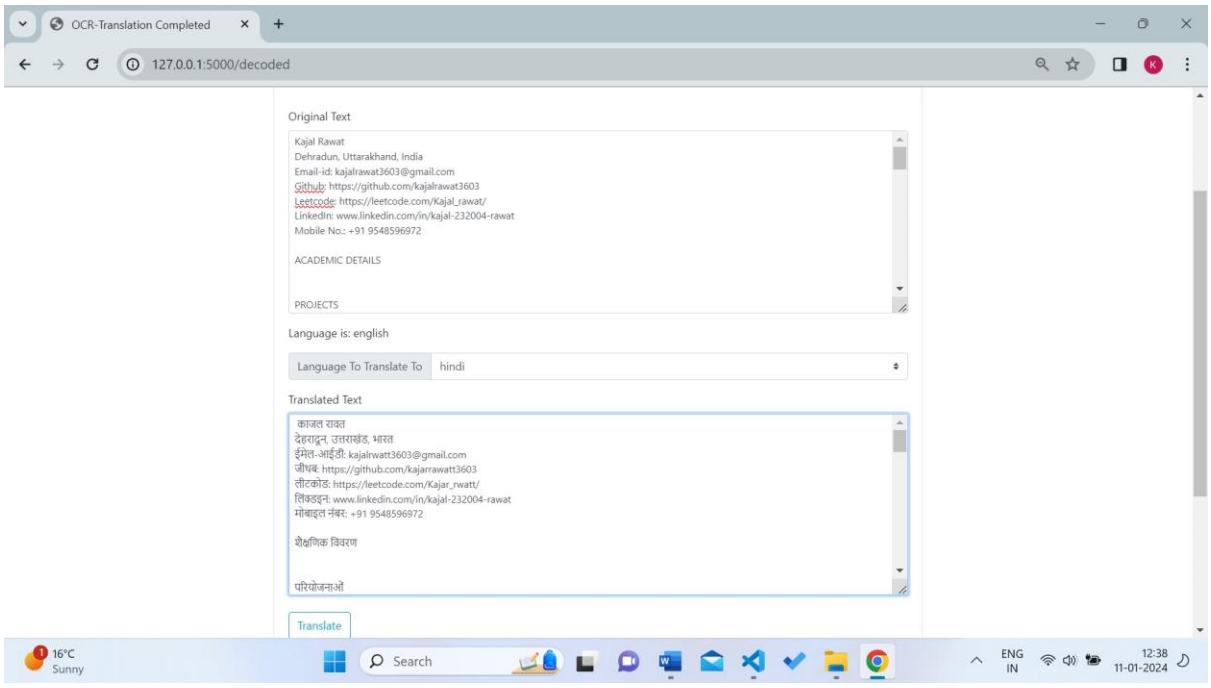


Fig.7: Translation Completed

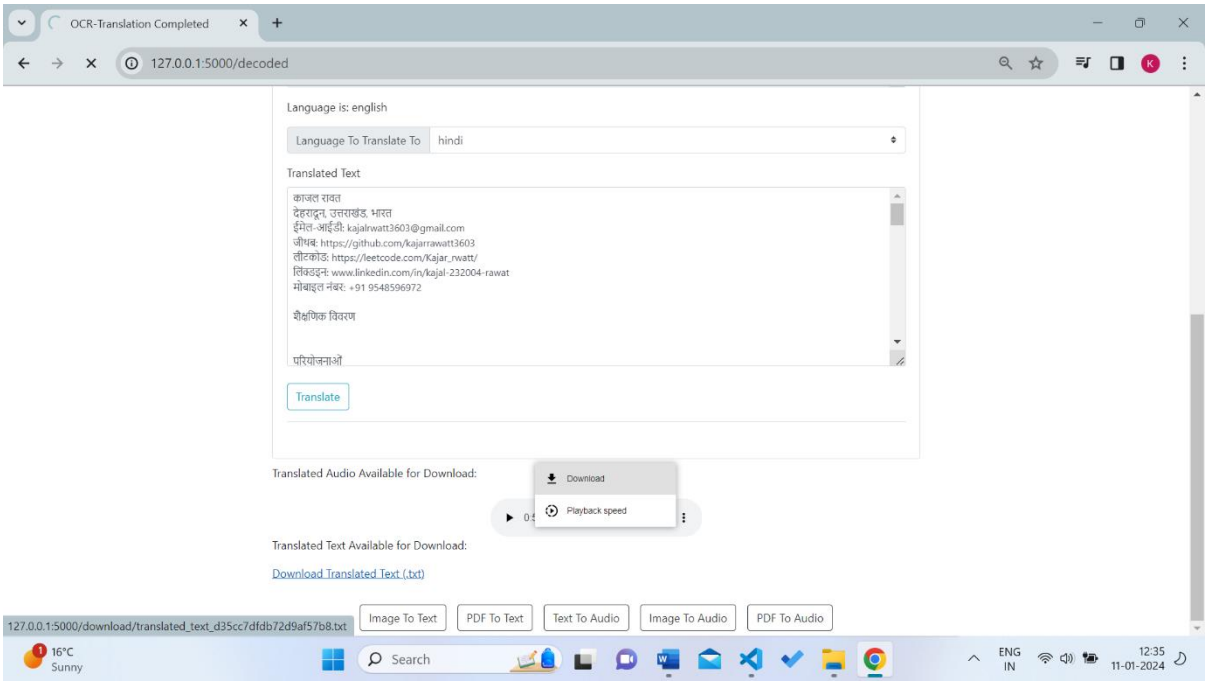


Fig.8: Audio and Text files generated for download.

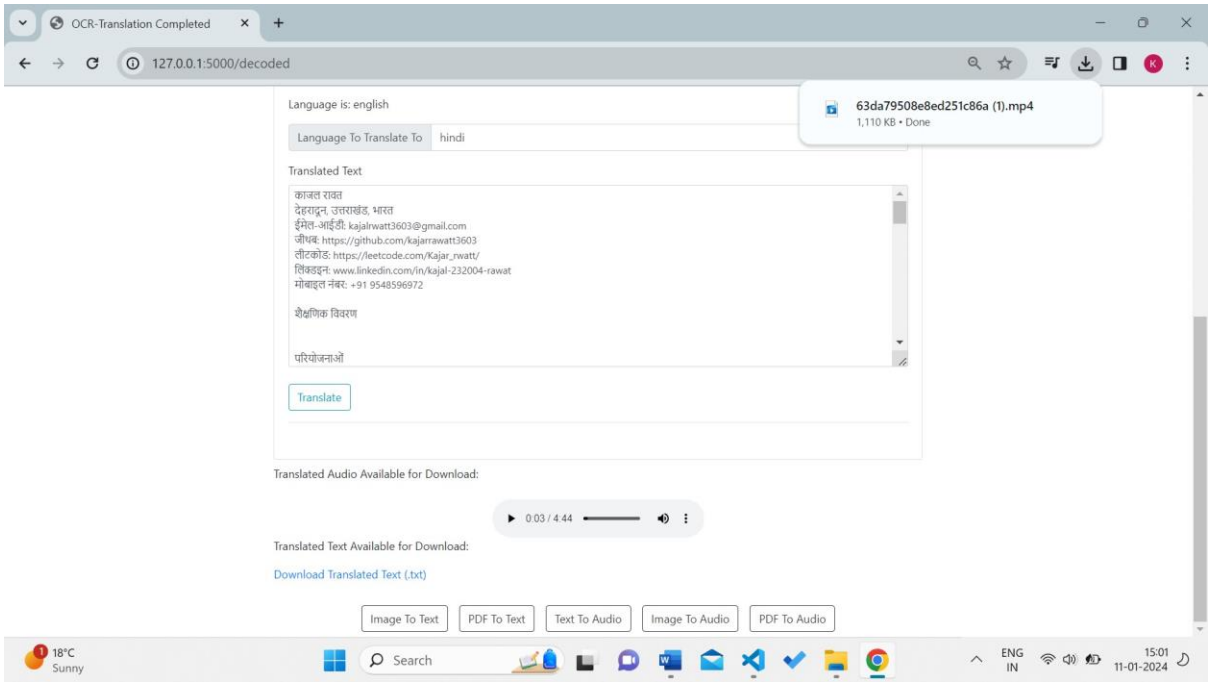


Fig.9: Downloaded Text/Audio files

Chapter 5

SAMPLE CODE

Environment Settings/Requirments:

- IDE == VSCode
- Python Version== 3.11.0

Package	Version	Description
certifi	2021.10.8	Certificate authority bundle for secure connections.
chardet	3.0.4	Detects the character encoding of text for proper decoding.
click	8.1.7	Creates command-line interfaces for user interaction.
Flask	3.0.0	A web framework for building the OCR application.
Flask-Dropzone	1.6.0	Integrates DropzoneJS for handling file uploads in Flask.
gTTS	2.2.3	Google Text-to-Speech API wrapper for generating speech.
h11	0.9.0	Supports HTTP/1.1 communication in Python.
h2	3.2.0	Implements the efficient HTTP/2 protocol in Python.
httpcore	0.9.1	Powers the HTTP client for making requests.
httpx	0.13.3	A feature-rich HTTP client for Python 3.7 and above.
hyperframe	5.2.0	Handles the framing layer for HTTP/2 in Python.
opencv-python	4.5.5.62	Computer vision library for image processing in Python.
Pillow	9.3.0	Image processing library essential for OCR tasks.
playsound	1.3.0	Plays sound, useful for notifications in the application.
pytesseract	0.3.8	OCR engine for recognizing text in images.
requests	2.27.1	Simplifies making HTTP requests for data retrieval.
urllib3	1.26.8	Manages HTTP connections and requests in Python.

Table.1: Required Pakages and Versions

Code SnapShots:

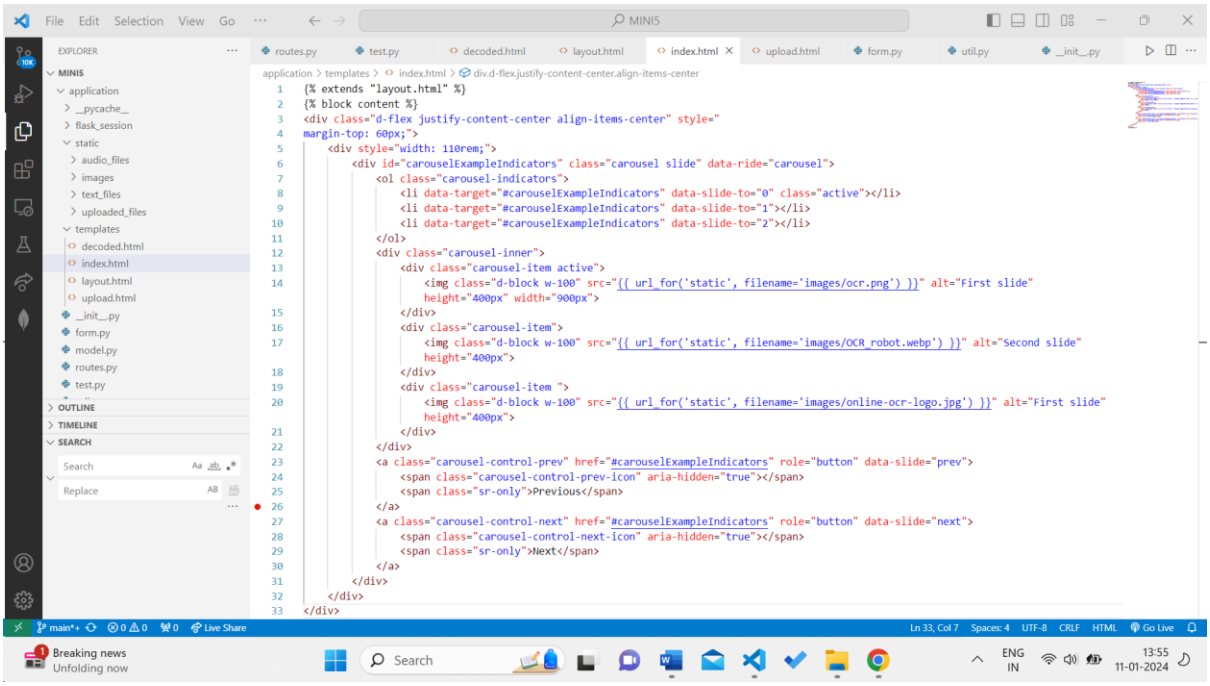


Fig.10: Sample HTML page (index.html)

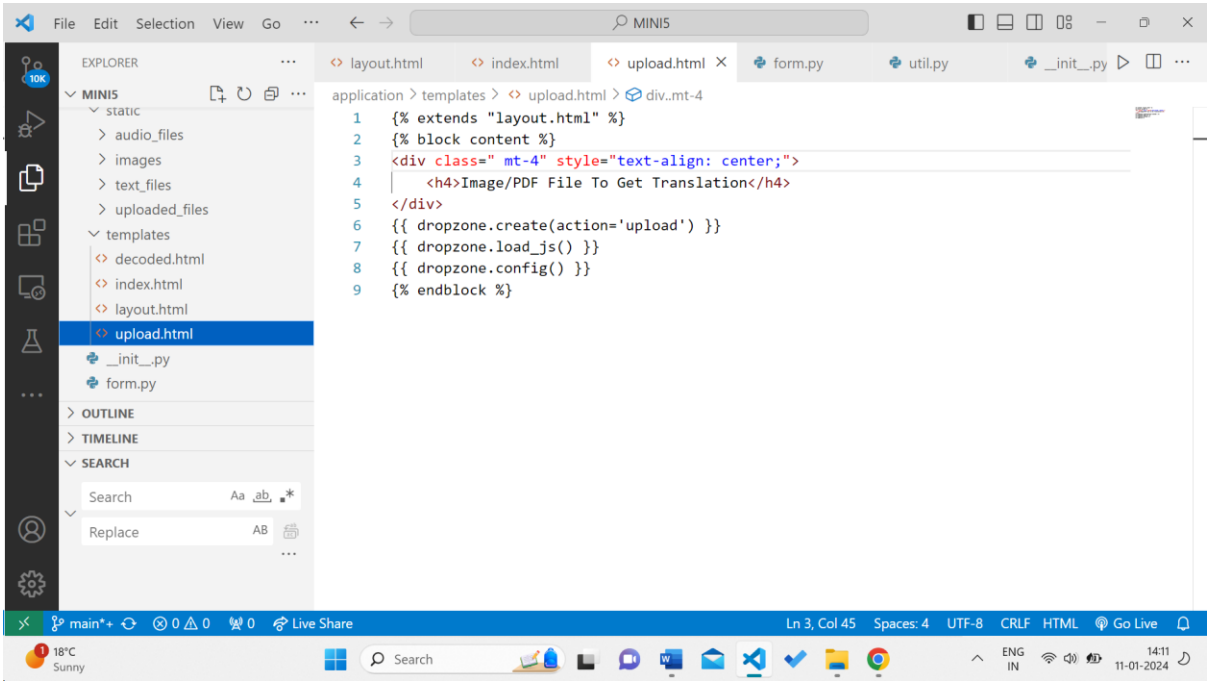
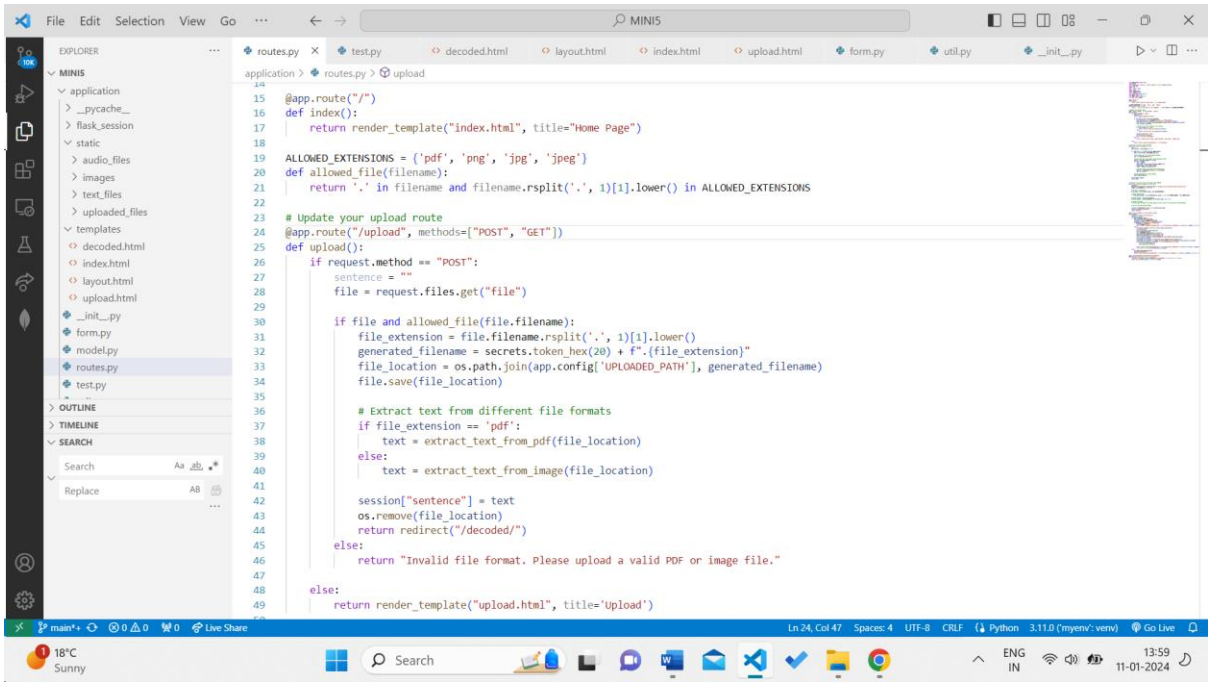
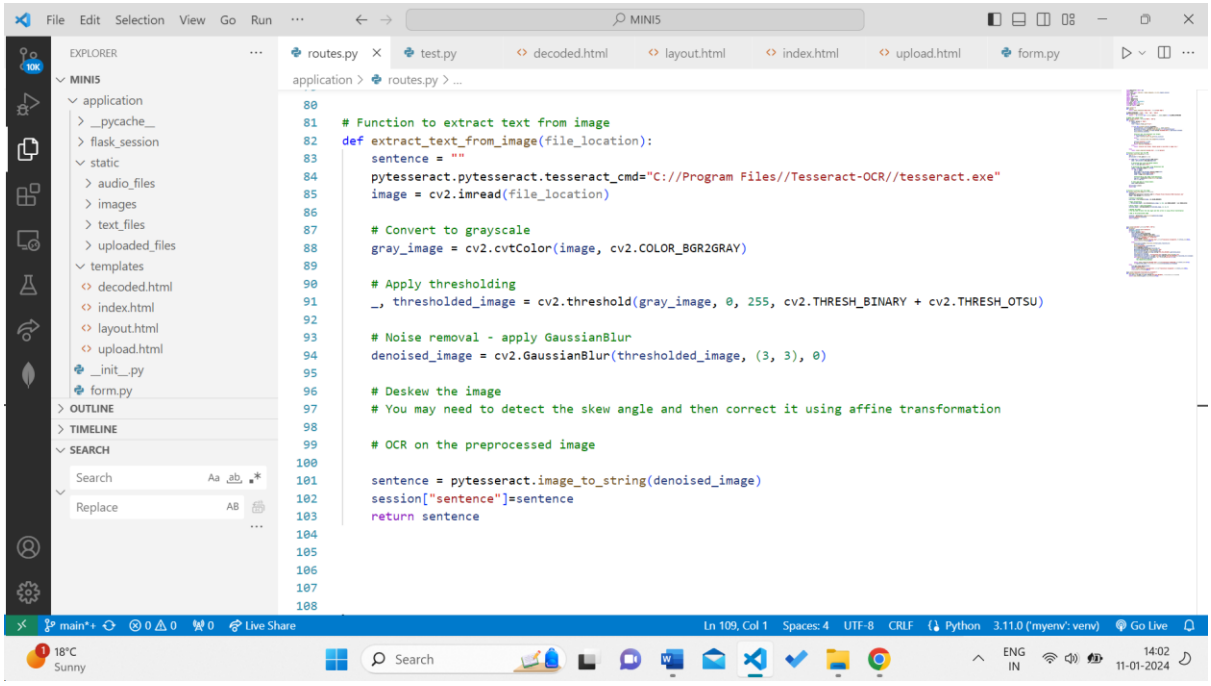


Fig.11: For uploading files using Drag and Drop (upload.html)



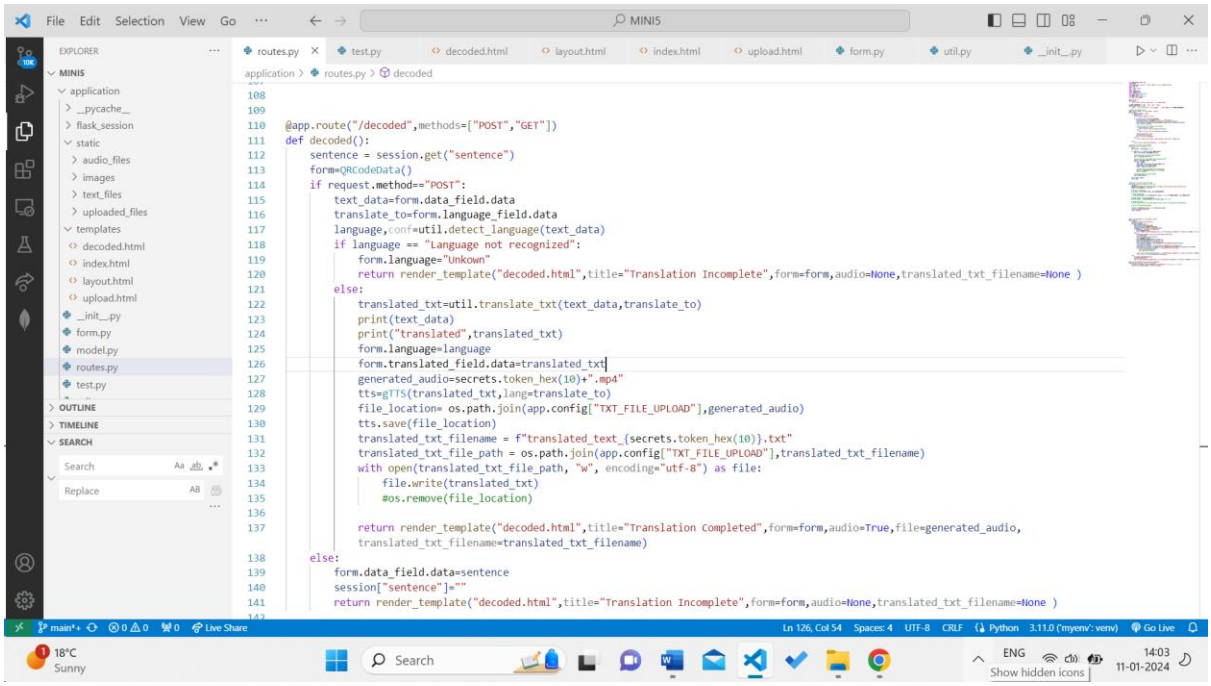
```python
@app.route("/")
def index():
    return render_template("index.html", title="Home Page")

ALLOWED_EXTENSIONS = {'pdf', 'png', 'jpg', 'jpeg'}
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

# Update your upload route
@app.route("/upload", methods=["POST", "GET"])
def upload():
    if request.method == "POST":
        sentence = ""
        file = request.files.get("file")

        if file and allowed_file(file.filename):
            file_extension = file.filename.rsplit('.', 1)[1].lower()
            generated_filename = secrets.token_hex(20) + f".{file_extension}"
            file_location = os.path.join(app.config['UPLOADED_PATH'], generated_filename)
            file.save(file_location)

            # Extract text from different file formats
            if file_extension == 'pdf':
                text = extract_text_from_pdf(file_location)
            else:
                text = extract_text_from_image(file_location)

            session["sentence"] = text
            os.remove(file_location)
            return redirect("/decoded/")
        else:
            return "Invalid file format. Please upload a valid PDF or image file."

    else:
        return render_template("upload.html", title='Upload')
```

Fig.12: Request for upload Document (routes.py)



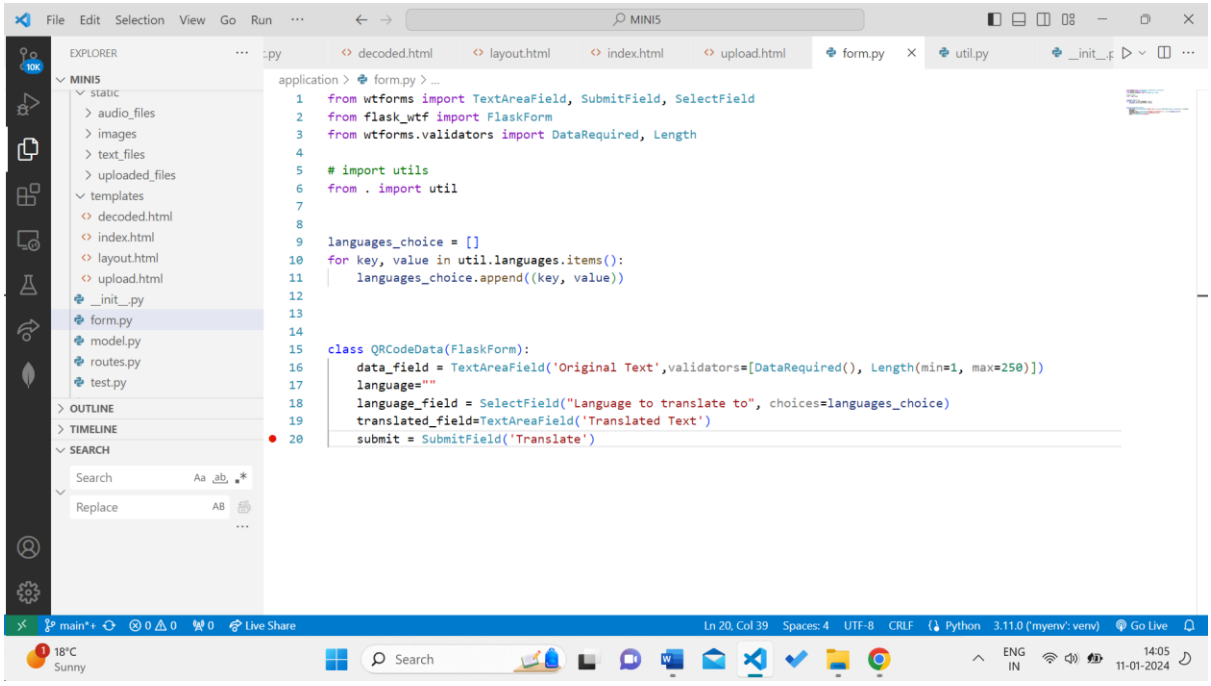
```python
# Function to extract text from image
def extract_text_from_image(file_location):
    sentence = ""
    pytesseract.pytesseract.tesseract_cmd="C://Program Files//Tesseract-OCR//tesseract.exe"
    image = cv2.imread(file_location)

    # Convert to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply thresholding
    _, thresholded_image = cv2.threshold(gray_image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    # Noise removal - apply GaussianBlur
    denoised_image = cv2.GaussianBlur(thresholded_image, (3, 3), 0)

    # Deskew the image
    # You may need to detect the skew angle and then correct it using affine transformation

    # OCR on the preprocessed image

    sentence = pytesseract.image_to_string(denoised_image)
    session["sentence"]=sentence
    return sentence
```

Fig.13: Text Extraction (routes.py)



```python
@app.route("/decoded",methods=["POST","GET"])
def decoded():
    sentence = session.get("sentence")
    form=QRCodeData()
    if request.method=="POST":
        text_data=form.data_field.data
        translate_to=form.language_field.data
        language,conf=util.detect_language(text_data)
        if language == "Language not recognized":
            form.language="Unkown"
            return render_template("decoded.html",title="Translation Incomplete",form=form,audio=None,translated_txt_filename=None )
        else:
            translated_txt=util.translate_txt(text_data,translate_to)
            print(text_data)
            print("translated",translated_txt)
            form.language=language
            form.translated_field.data=translated_txt
            generated_audio=secrets.token_hex(10)+".mp4"
            tts=gTTS(translated_txt,lang=translate_to)
            file_location= os.path.join(app.config["TXT_FILE_UPLOAD"],generated_audio)
            tts.save(file_location)
            translated_txt_filename = f"translated_text_{secrets.token_hex(10)}.txt"
            translated_txt_file_path = os.path.join(app.config["TXT_FILE_UPLOAD"],translated_txt_filename)
            with open(translated_txt_file_path, "w", encoding="utf-8") as file:
                file.write(translated_txt)
                #os.remove(file_location)

            return render_template("decoded.html",title="Translation Completed",form=form,audio=True,file=generated_audio,
            translated_txt_filename=translated_txt_filename)
    else:
        form.data_field.data=sentence
        session["sentence"]=""
        return render_template("decoded.html",title="Translation Incomplete",form=form,audio=None,translated_txt_filename=None )
```

Fig.14: Post request to send translated files (routes.py)



```python
from wtforms import TextAreaField, SubmitField, SelectField
from flask_wtf import FlaskForm
from wtforms.validators import DataRequired, Length

# import utils
from . import util


languages_choice = []
for key, value in util.languages.items():
    languages_choice.append((key, value))



class QRCodeData(FlaskForm):
    data_field = TextAreaField('Original Text',validators=[DataRequired(), Length(min=1, max=250)])
    language=""
    language_field = SelectField("Language to translate to", choices=languages_choice)
    translated_field=TextAreaField('Translated Text')
    submit = SubmitField('Translate')
```

Fig.15: Form to get the translated files (form.py).

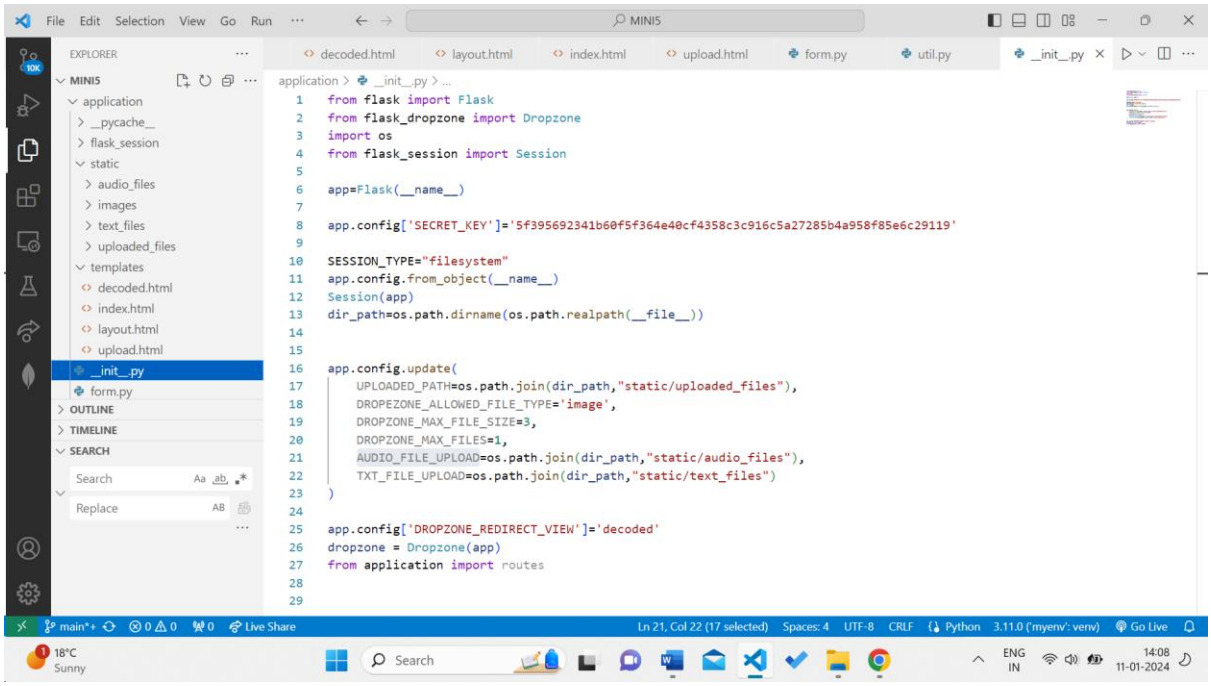


Fig.16: (__init__.py)

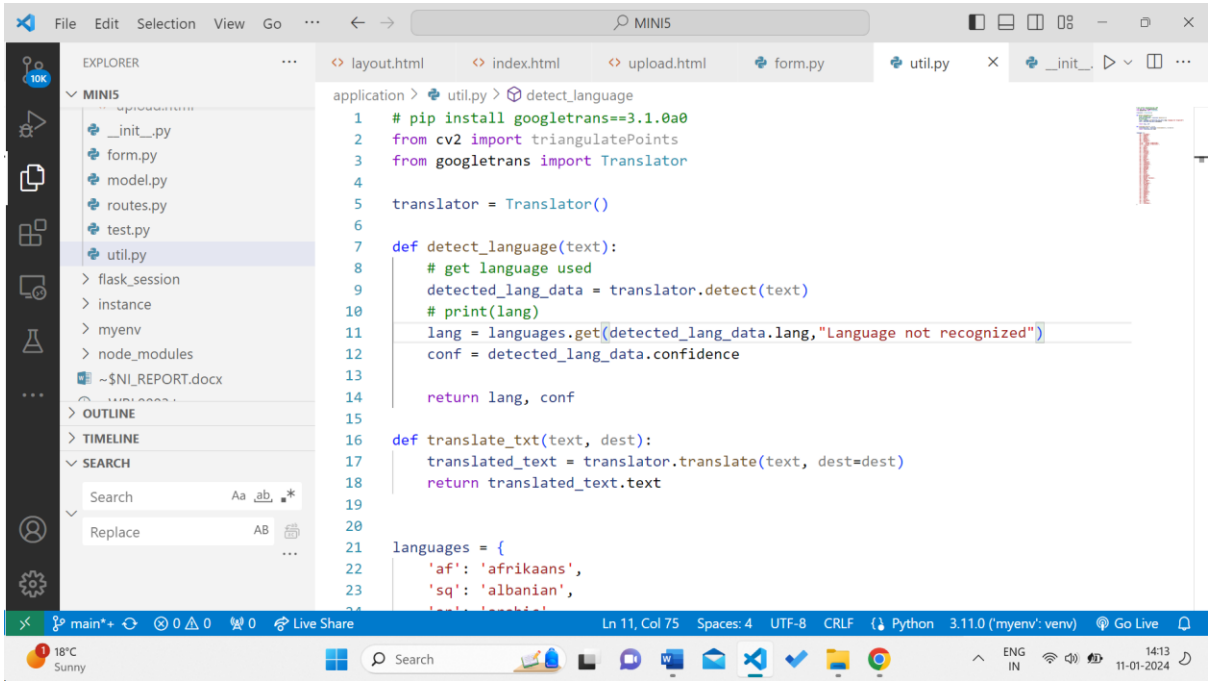


Fig.17: Translation of text

Chapter 6

FUTURE SCOPE

- **Improved Language Support:** Expanding language options for translation and enhancing accuracy, especially for less common languages.

- **Advanced Image Processing:** Implementing sophisticated algorithms and deep learning models for better text extraction from complex or degraded images.

- **AI-based Document Understanding:** Developing systems to analyze document structures, extract meaningful content, and automate summarization.

- **Enhanced Accessibility:** Improving the user interface for accessibility and incorporating features for users with disabilities.

- **Real-time Translation and Interaction:** Enabling live translation and voice-based interaction for instant communication.

- **Cloud Integration and Scalability:** Moving towards cloud-based solutions for scalability and better performance.

Chapter 7

CONCLUSION

Through OCR and translation technology, this project takes a big step in making information easy to get. The web app we built can read and translate text from PDFs and images smoothly. We used tools like Pytesseract, and gTTS to make it work. This helps people understand and use information, no matter the language.

During the project, we added key features like handling different file types, pulling out text, translating, and creating audio. I also made sure the app can handle errors well and get feedback from users, making it user-friendly even when things get tough.

While we achieved a lot, there's more inhancement that we can do like to add more languages, improve how we process images, and use AI to understand documents better. Making the app work on mobiles, doing real-time translations, and making it even friendlier with a better look are on our to-do list.

In conclusion, this project isn't just about solving today's problems; it's about starting something new. We're paving the way for more ideas in AI, making information accessible to everyone. This could change how people connect with and get information, making our digital world more inclusive.

Chapter 8

REFRENCES

- ibm.com/blog/optical-character-recognition/
- https://gtts.readthedocs.io/en/latest/
- https://flask-dropzone.readthedocs.io/en/latest/
- https://flask.palletsprojects.com/en/3.0.x/
- https://pillow.readthedocs.io/en/latest/installation.html
- https://docs.opencv.org/4.x/d7/dbd/group__imgproc.html
- https://py-googletrans.readthedocs.io/en/latest/
- https://ieeexplore.ieee.org/document/9935961
- https://ieeexplore.ieee.org/document/6993174
- youtube