# **Internship at PARIMAL IIT Roorkee**

# **Project Documentation**

# **Title: Project Documentation: OCR + Document Search**

Name: Nihaarika Jami

**Introduction:**

Optical Character Recognition (OCR) is referred to as Text Recognition. We can extract text from the image, cameras, scanned documents, and much more.

OCR is most commonly used to convert hard copies of medical or historical documents into text that users can easily edit. The primary goal of OCR software is to identify and capture all words in various languages from written text.

**Project Overview**

The Web OCR Project is a web-based prototype designed for Optical Character Recognition (OCR) that can process images containing text in both English and Hindi. The application allows users to upload images for text extraction and provides various functionalities such as keyword search, text download, and historical tracking of extractions.

**Features**

* **Multi-language** Support: Processes text in both Hindi and English.
* **Image Upload:** Users can upload images containing text for processing**.**
* **Text Extraction:** Displays extracted text along with the total word count.
* **Keyword Search:** Implements a basic search functionality that highlights matching sections in the extracted text.
* **History Tab:** Tracks and displays previous extractions for user reference.
* **Download Options**: Allows users to download extracted text in JSON and CSV formats.
* **User Interface**: An attractive UI, including an "About Us" section which tells about the function of this website and how to work with it.

**Significance:**

The Web OCR Project holds considerable significance for various stakeholders, including individual users, educational institutions, businesses, and researchers. Here are some key points highlighting its importance:

1. Enhanced Accessibility: The ability to extract text from images makes information more accessible, especially for individuals with visual impairments or reading difficulties. The OCR technology democratizes access to printed materials, ensuring that important content is available to a wider audience.
2. Efficiency in Data Processing: For businesses and organizations, the project streamlines the process of data extraction from documents, invoices, and forms. By automating the extraction process, users save time and reduce the likelihood of errors associated with manual data entry.
3. Multilingual Support: The inclusion of both Hindi and English language support caters to a diverse user base, particularly in regions where these languages are prevalent. This feature promotes inclusivity and enhances the utility of the application in multilingual contexts.
4. Educational Applications: Students and educators can benefit significantly from the OCR capabilities, allowing them to digitize printed materials, lecture notes, and textbooks. This functionality supports learning and research by enabling easier access to information.
5. Research and Documentation: Researchers can utilize the application to digitize and organize materials from various sources, facilitating literature reviews and data analysis. The ability to maintain a history of extractions allows for better documentation and retrieval of past work.
6. User-Centric Design: By prioritizing user experience, the project demonstrates a commitment to meeting the needs of its audience. The intuitive interface and useful features enhance usability, encouraging more people to adopt the technology.
7. Continuous Improvement and Innovation: The project's dedication to ongoing development ensures that it remains relevant in a rapidly evolving technological landscape. Future enhancements, such as spellcheck and grammar check functionalities, will further increase its value and appeal.
8. Contribution to Open Source Community: By being an open-source project, it encourages collaboration and knowledge sharing within the developer community. Contributions from various individuals can lead to new features and improvements, fostering innovation and creativity.
9. Environmental Impact: By promoting digital documentation and reducing reliance on physical papers, the Web OCR Project contributes to sustainability efforts. It encourages the digitization of documents, minimizing paper waste and promoting eco-friendly practices.

**Future Enhancements**

* **User Authentication**: Implement user accounts for personalized experiences.
* **Spellcheck and Grammar Check:** Suggests corrections for the extracted text.
* **Handwriting Recognition**: A specialized mode for recognizing handwritten text, with an option to switch between printed text and handwriting.
* **Advanced Analytics:** Provide insights on text extraction and usage statistics.
* **Multi-file Upload:** Allow users to upload multiple images at once.

**Limitations and Constraints:**

* **File Size Limitation:** The system restricts the size of the uploaded .csv files to a maximum of 200MB to ensure smooth processing and avoid performance issues.
* **Input Format Restriction:** The application only accepts image files and they must be of type: .jpg, .jepg, .png and must include only Hindi and English languages in proper fonts. Any deviation from this format results in “no text detected” error.
* **Computational Resources:** The efficiency and speed of real-time fraud detection depend on the computational resources available. Limited resources may affect the performance of the syste

**Technology Stack**

* **Streamlit:** A Python library for creating web applications easily and quickly.
* **Pytesseract:** A Python wrapper for Google's Tesseract-OCR Engine to extract text from images.
* **Pillow**: A Python Imaging Library to open, manipulate, and save various image file formats.
* **Pandas:** A data manipulation and analysis library for Python.
* **NumPy:** A library for numerical operations in Python**.**
* **Setuptools**: A package development and distribution library.
* **Distlib:** A library for working with distributions of Python packages**.**
* **Distutils:** A library for building and installing Python packages.

### **•Architecture/Design:**

**Overview:**

Web OCR System consists of several high-level components that interact seamlessly to provide real-time extraction of text from images. The architecture is designed to be modular, scalable, and user-friendly.

**High-level components:**

User Interface (UI)

Tesseract

Error Handling

**Design Decisions:**

* **Design Pattern:** MVC (Model-View-Controller) pattern for separation of concerns.
* **Trade-offs:** Chose Streamlit for rapid development despite its limitations in creating highly customizable UI components.
* **Training Model:** Choose Tesseract-OCR for its robustness and performance.

## **Development:**

* The project was developed using Python and a combination of machine learning libraries like Scikit-learn and TensorFlow.
* Python's wide range of machine learning libraries required were crucial for developing the model.
* The model was built using a structured approach, starting with data exploration, cleaning and preprocessing, followed by model selection, and hyperparameter tuning, splitting and training the datasets.
* The development process adhered to coding standards and best practices, including modular code design, documentation, and code review. **Challenges:**

### **Initial usage of Qwen2-VL:** Faced issues like “ImportError: cannot import name 'QwenImageProcessor' from 'transformer.” “ImportError: cannot import name 'SAVE\_STATE\_WARNING' from 'torch.optim.lr\_scheduler'”. So, I’ve used

### Tesseract-OCR for implementing the application.

### **Deployment:** While deploying “error: tesseract is not installed or it’s not in you PATH. See README File for mode information.” To resolve that I’ve created packages.exe file to install those dependencies.

### **Testing:**

**7.1. Testing Approach:**

1. **Unit Tests**: Individual components of the system, such as the machine learning model, data processing, and prediction logic, were tested in isolation to ensure they functioned correctly.
2. **Integration Tests**: Components were tested together to ensure they interacted correctly and produced the expected results.
3. **System Tests**: The entire system was tested end-to-end to ensure it met the requirements and worked as expected.

**Deployment process:**

The deployment process for the Web OCR System using Machine Learning involves the following steps:

1. **Testing and Validation**: The code is tested and validated to ensure it meets the requirements and works as expected.
2. **Packaging**: The code is packaged into a deployable format, including any dependencies and configurations.

**Deployment Scripting**: Deployment scripts are used to automate the deployment process, including environment setup, configuration, and deployment of the application.

### **User Guide for Fraud Detection System**

#### **Setup and Configuration**

1. **Environment Setup:**
   * Ensure you have a stable internet connection.
   * Use a modern web browser (Chrome, Firefox, Safari, etc.).
2. **Application Access:**
   * Navigate to the provided URL or local host where the Streamlit application is hosted.

## **File Structure**

/Web\_OCR\_Project

│

├── model.py # Main application file

├── requirements.txt # Project dependencies

├── packages.txt # To install dependencies

├── README.md # Documentation file

├── Deployed app link

└── lgog.jpg # Logo for the application

#### **Using the Application**

#### **Home Page**

#### The Home page serves as the main entry point for users accessing the Web OCR application. It includes:

#### **Image Upload Section:** A prominent area where users can drag and drop or browse their files to upload images containing text. This section should be user-friendly, with clear instructions for uploading.

#### **Extraction Button:** Once an image is uploaded, users can click a button labeled "Perform OCR" to initiate the OCR process. This button should be easily identifiable and accessible.

#### **Overview of Features:** A brief summary of the key features available in the application, such as:

#### Multi-language support (Hindi and English)

#### Keyword search functionality

#### Download options for extracted text

#### **Navigation Menu:** A menu at the side of the page, allowing users to navigate to other sections such as About Us, History, and any future enhancements we plan to include.

#### **About Us Page**

#### The About Us page provides users with insight into the mission and vision of the Web OCR Project. This page typically includes:

#### **History Page**

#### The History page allows users to view and manage their previous extractions. It includes**:**

#### **Extraction History Table:** A well-organized table listing all previous text extractions, including:

#### Image Thumbnail: A small preview of the uploaded image for easy identification.

#### Extracted Text Summary: A brief snippet or word count of the extracted text.

#### **Download Options:** Provide buttons for users to download the extracted text for each entry in various formats (e.g., JSON, CSV).

#### **Troubleshooting Tips**

1. **Session State Issues:** If session variables (e.g., prediction history) are not persisting, try refreshing the page or restarting the application.
2. **File Upload Errors:** Ensure CSV files uploaded for batch prediction follow the specified format and contain required columns.

**Project Conclusion**The Web OCR Project is a powerful and versatile tool designed to streamline the process of extracting text from images, catering to users who require OCR capabilities in both Hindi and English. With its user-friendly interface, innovative features, and commitment to continuous improvement, this application stands as a valuable resource for individuals and organizations alike.

### By integrating advanced technologies such as Pytesseract and Streamlit, the project not only addresses the core needs of text extraction but also enhances user experience with additional functionalities like keyword search, download options, and handwriting recognition. The inclusion of a History page further empowers users to track their previous extractions, making the application efficient and convenient for ongoing use.

### As the project evolves, future enhancements will focus on expanding capabilities, optimizing performance, and fostering community engagement. The dedication to user-centric design ensures that the Web OCR Project will adapt to the changing needs of its users while maintaining high standards of quality and reliability.

### In summary, the Web OCR Project represents a significant step forward in OCR technology, aiming to simplify text extraction for diverse applications and contribute to a more accessible and efficient digital world. We invite users to explore the application, provide feedback, and join us on this exciting journey of innovation and growth.

### **Appendices:**

Code:

import streamlit as st

import pytesseract

from PIL import Image

import re

import pandas as pd

import json

from io import BytesIO

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# Function for OCR using pytesseract with exception handling

def perform\_ocr(image):

    try:

        # Perform OCR

        text = pytesseract.image\_to\_string(image, lang='eng+hin')

        # Check if any text was detected

        if not text.strip():

            raise ValueError("No text detected in the image.")

        # Check for languages other than Hindi and English

        if not re.search(r'[a-zA-Z\u0900-\u097F]', text):

            raise ValueError("Text in other languages detected. This application only supports Hindi and English.")

        return text

    except Exception as e:

        st.error(f"Error: {e}")

        return ""

# Function to count words in the extracted text

def count\_words(text):

    words = text.split()

    return len(words)

# Save history of extracted text and images

def save\_history(image, extracted\_text):

    if 'history' not in st.session\_state:

        st.session\_state.history = []

    # Save the image as bytes and keep the extracted text

    img\_bytes = BytesIO()

    image.save(img\_bytes, format='PNG')

    st.session\_state.history.append({

        'image': img\_bytes.getvalue(),

        'text': extracted\_text

    })

# Function to download extracted text as JSON

def download\_json(extracted\_text):

    json\_data = json.dumps({"extracted\_text": extracted\_text}, indent=4)

    st.download\_button(

        label="Download Extracted Text as JSON",

        file\_name="extracted\_text.json",

        mime="application/json",

        data=json\_data

    )

# Function to export history as CSV

def download\_history\_csv():

    if 'history' in st.session\_state:

        history\_data = [{

            'text': entry['text']

        } for entry in st.session\_state.history]

        df = pd.DataFrame(history\_data)

        csv = df.to\_csv(index=False).encode('utf-8')

        st.download\_button(

            label="Download History as CSV",

            data=csv,

            file\_name="history.csv",

            mime="text/csv"

        )

# Main Streamlit app

def main():

    # Set page title, logo, and sidebar navigation

    st.set\_page\_config(page\_title="Hindi-English OCR", page\_icon="🖼️")

    # Set background color

    st.markdown(

        """

        <style>

        .stApp {

            background-color: #87CEEB;  /\* Sky blue color \*/

        }

        .stButton button {

            background-color: #87CEEB;  /\* Sky blue color for buttons \*/

            color: white;

        }

        </style>

        """,

        unsafe\_allow\_html=True

    )

    st.sidebar.image("lgog.jpg", width=200)  # Placeholder for your logo image

    st.sidebar.title("Online-OCR")

    nav = st.sidebar.radio("Go to", ["Home", "About Us", "History"])

    # About Us Section

    if nav == "About Us":

        st.title("About Us")

        st.write("""

        \*\*OCR Web Application\*\* allows you to upload images containing text in Hindi and English, extract text, search for keywords, and download results in various formats.

        This project uses \*\*Pytesseract\*\*, an optical character recognition (OCR) tool for Python.

        """)

        st.write("""\*\*THE STEPS TO FOLLOW:\*\* """)

        st.write("""->Select the image according to the constraints shown.""")

        st.write("""->Click on "Perform OCR".""")

        st.write("""->If needed, you can also search for a keyword from the extracted text.""")

        return

    # History Section

    if nav == "History":

        st.title("OCR History")

        if 'history' in st.session\_state and st.session\_state.history:

            for i, record in enumerate(st.session\_state.history):

                st.subheader(f"Entry {i+1}")

                st.image(record['image'], caption="Uploaded Image")

                st.write(record['text'])

            download\_history\_csv()

        else:

            st.write("No history available yet.")

        return

    # Home (OCR functionality)

    st.title("Hindi-English OCR Application")

    st.subheader("Upload an image containing text in both Hindi and English")

    # Initialize session state for extracted text

    if 'extracted\_text' not in st.session\_state:

        st.session\_state.extracted\_text = ""

    # File uploader for image

    uploaded\_file = st.file\_uploader("Choose an image...", type=["jpg", "jpeg", "png"])

    if uploaded\_file is not None:

        # Display uploaded image

        image = Image.open(uploaded\_file).convert('RGB')  # Ensure image is in RGB format

        st.image(image, caption='Uploaded Image', use\_column\_width=True)

        # Perform OCR on image

        if st.button("Perform OCR"):

            extracted\_text = perform\_ocr(image)

            if extracted\_text:  # Only display text if OCR was successful

                word\_count = count\_words(extracted\_text)

                st.subheader("Extracted Text")

                st.write(extracted\_text)

                st.write(f"Number of words: {word\_count}")

                # Store the extracted text in session state

                st.session\_state.extracted\_text = extracted\_text

                # Save extraction to session state (for history)

                save\_history(image, extracted\_text)

                # Provide download options

                download\_json(extracted\_text)

    # Keyword search functionality

    if st.session\_state.extracted\_text:

        keyword = st.text\_input("Enter keyword to search:")

        if keyword:

            if re.search(keyword, st.session\_state.extracted\_text, re.IGNORECASE):

                st.success(f'Keyword "{keyword}" found in the extracted text!')

                highlighted\_text = re.sub(f'({keyword})', r'\*\*\1\*\*', st.session\_state.extracted\_text, flags=re.IGNORECASE)

                st.write(highlighted\_text)

            else:

                st.warning(f'Keyword "{keyword}" not found in the extracted text.')

if \_\_name\_\_ == "\_\_main\_\_":

    main()

WEB URL: https://ocr-and-document-search-web-application-8fyarmrxcfuwv7dhl6kfnf.streamlit.app/