DocuEase - Project Documentation

Live Demo

- **Frontend (React on Netlify):** https://docuease-ss.netlify.app/
  -users interact via the frontend
- **Backend API (Flask on Render):** https://ocr-backend-gr8w.onrender.com
  -API testing via Postman

---

1. Title & Tagline

DocuEase – Simplifying Document Digitization through OCR

A lightweight web application for seamless text extraction from images.

---

2. Abstract / Introduction

DocuEase is a web-based Optical Character Recognition (OCR) platform designed to digitize printed or handwritten documents efficiently.

Built with a ***Python Flask backend*** integrated with ***Tesseract OCR*** and a responsive ReactJS frontend.
DocuEase supports English, Hindi, and Marathi text extraction.
The platform aims to simplify document digitization for academic, administrative, and research purposes with a minimal, aesthetic, and intuitive interface.

---

3. Problem Statement

In educational institutions, offices, and research domains, there is a constant need to digitize printed or handwritten documents for record-keeping, data analysis, and streamlined accessibility. Existing OCR solutions are often heavy, paid, or limited to specific languages. Furthermore, most tools lack a user-friendly interface tailored for multi-language support, aesthetic readability, and deployment ease.

There is a pressing need for a **lightweight, web-based OCR system** that:

- Supports **multiple Indian and English languages**.
- Provides **quick and accurate text extraction**.
- Is accessible via any device with internet connectivity without complex installations.
- Can be **easily extended** for additional features like text summarisation, speech conversion, or database integration.

> **Tech Stack**
>
> - **Frontend:** React.js (JavaScript, JSX, CSS)
> - **Backend:** Flask (Python), Tesseract OCR
> - **Deployment:**
>   - **Backend:* Render (Python web service)
>   - **Frontend:* Netlify (React static site)
> - Libraries & Tools:**
>   - Axios for API requests
>   - Gunicorn for production WSGI server
>   - Git & GitHub for version control
>   - Postman for API testing

---

4. Objective

The primary objective of this project is to **develop a web-based OCR application** that simplifies document digitization by:

- Providing an intuitive **frontend interface (DocuEase)** for users to upload images and select their preferred language (English, Hindi, Marathi).
- Implementing a robust **Python-based backend using Flask and Tesseract OCR** to process images and extract text efficiently.
- Deploying the solution using **Render for backend** and **Netlify for frontend**, ensuring global accessibility without the need for local installations.
- Offering a **responsive, aesthetic UI** with dark mode and animations to enhance user experience.
- Laying a foundation for **future enhancements** like summarization, text-to-speech, file export, or database integrations for institutional use.

---

5. Methodology

The project was implemented systematically using the following approach:

5.1. Problem Definition & Research

- Identified the need for **easy document digitization** in multiple languages.
- Researched existing OCR tools and frameworks, finalising **Tesseract OCR** for its accuracy and language support.

5.2. Backend Development (OCR API)

- Developed a **Flask backend** in Python with endpoints to:
  - Receive image and language input via POST requests.
  - Process images using Tesseract OCR with language models (eng, hin, mar).
  - Return the **extracted text as JSON** to the frontend.
- Dockerized the backend with **Tesseract dependencies** and deployed it on **Render** for continuous availability.

5.3. Frontend Development (User Interface)

- Created a **ReactJS frontend** enabling:
  - File upload functionality.
  - Language selection dropdown.
  - API integration using **Axios** to send data and receive OCR results.
  - Displaying extracted text dynamically.

5.4. UI/UX Enhancement

- Implemented:
  - **Dark mode toggle** for better accessibility.
  - Neon-themed aesthetics with **Fjalla One, Michroma, Bebas Neue fonts**.
  - Responsive design for mobile and desktop users.
  - Animated buttons and subtle transitions for interactive experience.

5.5. Deployment

- **Backend:** Deployed on **Render** with environment variables configured for language data.
- **Frontend:** Built production files and deployed via **Netlify** for free global hosting.
- Connected the frontend with backend API endpoints to ensure seamless operation.

---

6. Implementation

6.1. Backend

- Developed using **Python Flask**.
- Endpoint `/ocr` handles POST requests with file & language data.
- Uses **Pytesseract** to extract text based on selected language.
- Dockerized for consistent deployment with necessary Tesseract dependencies.

6.2. Frontend

- Developed using **React.js**.
- User selects image and language, submits form.
- Uses **Axios** to call backend API and displays extracted text dynamically.
- Includes **Dark Mode toggle**, aesthetic themes, and responsive design.

6.3. Deployment

- **Backend:**
  - Pushed to GitHub
  - Deployed on **Render** with automatic build from repo
  - Runs Gunicorn to serve Flask app in production
- **Frontend:**
  - Built with `npm run build`
  - Drag-and-drop deployed to **Netlify**
  - Accessible via custom Netlify URL

---

7. Features Implemented

✔ 7.1. Multi-language OCR Support

- Supports text extraction from images in **English, Hindi, and Marathi**.
- Easy language selection dropdown on the frontend.

✔ 7.2. File Upload Functionality

- Users can upload any **image file** (JPG, PNG, etc.) for text extraction.

✔ 7.3. Real-time API Integration

- Frontend sends image and language data to the backend OCR API.
- Extracted text is **returned and displayed dynamically**.

✔ 7.4. Responsive Frontend Design

- Works seamlessly on **mobile, tablet, and desktop**.
- Adjusts layout and input components for small screens.

✔ 7.5. Dark Mode Toggle

- Users can switch between **dark and light modes** for better accessibility.

✔ 7.6. Modern UI/UX Design

- Implemented using:
  - **Fjalla One, Michroma, Bebas Neue** fonts.
  - Neon glow effects for buttons.
  - Smooth transitions and hover animations.

✔ 7.7. Deployment

- **Backend deployed** on Render.
- **Frontend deployed** on Netlify with a unique domain name.

---

8. Limitations

⚠ 8.1. Cold Start Delay (Backend)

- Since the backend is deployed on **Render Free Tier**, if inactive for more than **15 minutes**, it goes into sleep mode.
- First request after inactivity experiences a **cold start delay** of **20–30 seconds**.

⚠ 8.2. Limited Language Models

- Currently supports only **English, Hindi, and Marathi**.
- For other languages, **additional traineddata files** need to be installed and configured.

⚠ 8.3. OCR Accuracy Constraints

- Tesseract OCR accuracy depends on:
  - Image clarity and resolution.
  - Font styles and handwriting quality.
- Might misinterpret **handwritten or stylized text**.

⚠ 8.4. File Size Limitation

- Extremely large image files may cause:
  - **Longer processing time**.
  - Backend timeout on Render free tier.

⚠️ 8.5. No User Authentication

- Currently, the platform is **open to all users** without login or data storage.

⚠️ 8.6. No Database Integration

- Extracted text is only displayed; **not saved in any database** for future retrieval or analysis.

⚠️ 8.7. Concurrency Limitations

- Free deployment tiers handle limited simultaneous requests. High concurrent usage may lead to:
  - **API failures**
  - Temporary unavailability until restarted automatically.

---

9. Future Scope

- **Add more languages** by integrating additional Tesseract traineddata files.
- **Improve handwriting recognition** using advanced OCR APIs.
- **Enable user login and database storage** for saving extraction history.
- **Support PDFs and multi-page documents** for broader use cases.
- **Integrate text-to-speech and translation** for accessibility.
- **Develop a mobile app** for on-the-go document digitization.
- **Enhance scalability and deployment** to handle more users efficiently.
- **Add AI-based text correction** to improve accuracy of results.
