“Images to Editable Documents Converter”

A

MINOR PROJECT REPORT

Submitted for fulfilment of the requirement for the award of degree

Bachelor of Engineering

In

Computer Science & Engineering

Submitted By:

Chitransh S. Vishwakarma (0101CS171029)

Bhupendra S. Choudhary (0101CS161026)

Guided By:

Prof. Uday Chourasia

Submitted To:

Department of Computer Science & Engineering,

University Institute of Technology,

Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal 462033

May 2020

UNIVERSITY INSTITUTE OF TECHNOLOGY

RAJIV GANDHI PROUDYOGIKI VISHWAVIDHYALAYA, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that **Chitransh S. Vishwakarma** and **Bhupendra S. Choudhary** of BE 3rd year, Computer Science & Engineering have completed their Minor Project entitled "**Images to Editable Document Converter**" during the academic year 2019 – 20 under my guidance and supervision.

I approve of the project for submission for the fulfilment of the requirement for the award of degree in Computer Science and Engineering.

Prof. Uday Chourasia

Project Guide

DoCSE, UIT RGPV

DECLARATION BY THE CANDIDATES

We hereby declare that the work which is being presented in the Minor Project entitled – **Images to Editable Document Converter** is submitted for the fulfilment requirement for the award of the degree in Computer Science & Engineering. The work has been carried out at the University Institute of Technology, RGPV, Bhopal in the academic session 2019 – 2020 is an authentication record of our work carried under the guidance of **Prof. Uday Chourasia,** Department of Computer Science & Engineering, UIT RGPV, Bhopal.

The matter written in this project has not been submitted for award of any other degree.

Chitransh S. Vishwakarma (0101CS171029)

Bhupendra S. Choudhary (0101CS161026)

ACKNOWLEDGEMENT

We are thankful to all the individuals who have lent us support and guidelines without which we could not have completed our project in the stipulated time.

First and foremost, we would like to express the deepest gratitude to our project supervisors **Prof. Uday Chourasia** for the invaluable support, guidance, motivation, and encouragement throughout the period. It was there enthusiastic and progressive outlook toward the project which inspired us throughout the work during this period.

We are grateful for our source of inspiration **Dr. Sanjay Silakari**, Prof. and Head of DoCSE, UIT RGPV, Bhopal, for his unforgettable support and inspiration and staff of the department who were involved in the project either directly or indirectly for their valuable cooperation.

In the end, we would like to forward our gratitude to our teammates, friends, and colleagues for their help and cooperation throughout this work.

Chitransh S. Vishwakarma (0101CS171029)

Bhupendra S. Choudhary (0101CS161026)

ABSTRACT

OCR (Optical Character Recognition) is gaining attention in the field of computer vision due to its promising applications in the areas of artificial intelligence, education, documentation, and pattern recognition. Conversion of Images containing textual matter into Editable Documents has always been a cumbersome task until the past few years. As of now, we have various tools and software available to accomplish this task with ease and get accurate results within a matter of seconds. With the advent of the field of OCR and computer vision techniques, it is possible to identify even the handwritten characters, which is one of the most significant research topics in the field of pattern recognition and computer vision. Existing technologies do not provide a local, easy-to-run web interface for students who are getting to know this field. Thus, the main goal of this study was to create an easy-to-use, getting started guide for students who wish to get started with research at the early stages of their careers in the field of OCR. We are providing a ready-made web-app and the core code along with the assembly of all the required libraries and tools which would be used by any student to get started in the field of optical character recognition. All that the students would have to do is to follow the step-by-step guidelines mentioned in the documentation for the project and they are prepared to explore the world with endless possibilities with OCR. The core functionality of the project begins with uploaded an image with some textual matter in it, to the web app interface which then gets passed on to the Tesseract-OCR Engine which extracts all textual matter from the image and then the 'python-docx' tool appends the extracted text from the image onto a blank MS Word document and the web interface enables the download button which may then be used to download the converted editable document.

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION

1.1 Introduction

Optical Character Recognition is the conversion of images containing typed, handwritten, or printed textual matter into machine-encoded text. OCR has been widely used as a form of data entry for various printed paper data records. It is a common method of digitizing the printed texts so that they can be electronically modified, stored, and searched. OCR is a field of research in pattern recognition, artificial intelligence, and computer vision. Early versions needed to be trained with images of each character and worked on each font separately. Advanced systems are capable of producing highly accurate character recognition outputs with support for a variety of digital image file format inputs.

Various techniques used in OCR are listed below

Pre-processing – Techniques included in pre-processing are

- **De-skew** – if the document was not aligned properly when scanned, it may need to be tilted a few degrees clockwise or counter-clockwise in order to make lines of text perfectly horizontal or vertical.
- **De-speckle** – Remove positive and negative spots, smoothing edges.

- **Binarisation** – Convert an image from color or greyscale to black & white (called a binary-image because there are two colors). The task of binarization is performed as simple way of separating the text from the background. The task of binarization itself is necessary since most commercial recognition algorithms work only on binary images since it proves to be simpler to do so. In addition, the effectiveness of the binarization step influences to a significant extent the quality of the character recognition stage and the careful decisions are made in the choice of the binarization employed for a given input image type.
- **Line Removal** – cleans up non-glyph boxes and lines
- **Layout Analysis** – Identifies columns, paragraphs, captions etc. as distinct blocks. Especially important in multi-column layouts and tables.
- **Line & Word Detection** – Establishes baselines for word and character shapes, separate words if necessary.
- **Script Recognition** – In multilingual documents, the script may change at the level of the words and hence, identification of the script is necessary, before the right OCR can be invoked to handle the specific script.
- **Segmentation** – For per-character OCR, multiple characters that are connected due to image artifacts must be separated, single characters are broken into multiple pieces.
- **Normalise Aspect Ratio & Scale**

Segmentation of fixed-pitch points is accomplished relatively simply by aligning the image to a uniform grid based on where vertical grid lines will least often intersect black areas.

Text Recognition - There are two basic types of core OCR Algorithm, which may produce ranked list of candidate characters.

1. Matrix Matching
2. Feature Extraction

Matrix Matching – it involves comparing of an image to a stored glyph on a pixel-by-pixel basis, it's also known as **pattern recognition** or **image correlation**.

This relied on

1. the input glyph being correctly isolated from the rest of the image
2. the stored glyph being in a similar font and at the same scale

This technique works best with the typewritten text and does not work well when new fonts are encountered. This is the technique the early physical based OCR implemented, rather directly.

Feature Extraction – it decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections. The extraction features reduce the dimensionality of the representation and makes the recognition process computationally efficient. These features are compared with an abstract vector-like representation of a character, which might reduce to one or more glyph prototypes.

Post – processing – OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document. This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains word not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy.

1.2 History

The Tesseract Engine was originally developed as a proprietary software at **Hewlett Packard** labs in Bristol, England & Greeley, Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some migrations from C to C++ in 1988. A lot of the code was written in C, and then some more was written in C++. Since then all the code has been converted to at least compile with a C++ compiler. Very little work was done in the following decade. It was then released as open source in 2005 by Hewlett Packard and the University of Nevada, Las Vegas (UNLV). Tesseract development has been sponsored by Google since 2006.

1.3 Architecture

Tesseract OCR works in step by step manner as per the block diagram shown in fig. 1. First step is Adaptive Thresholding, which converts the image into binary images. Next Step is connected component analysis, which is used to extract character outlines. This method is very useful because it does the OCR of image with white text and black background.

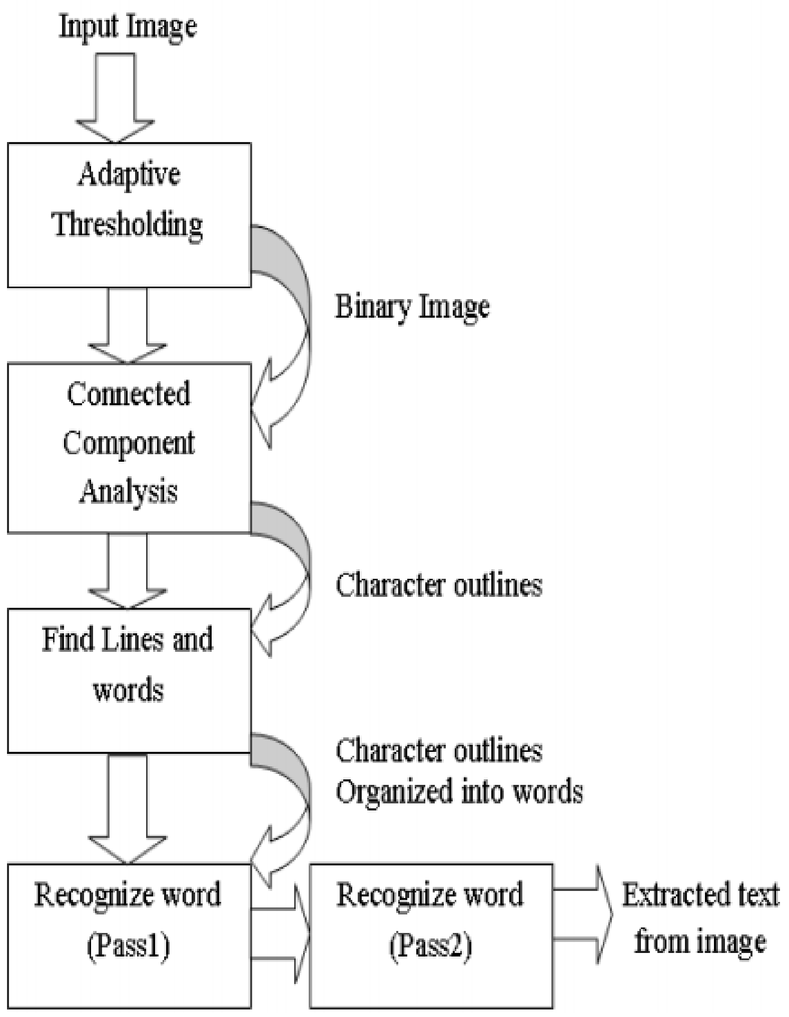


Fig. 1. Architecture of Tesseract OCR

Tesseract was probably first to provide this kind of processing. Then after, the outlines are converted into Blobs. Blobs are organized into text lines, and the lines and regions are analysed for some fixed area or equivalent text size. Text is divided into words using definite spaced and fuzzy spaces. Recognition of text is then started as two-pass process shown in fig. 1. In the first pass, an attempt is made to recognize each word from the text. Each word passed satisfactory is passed to an adaptive classifier as training data. The adaptive classifier tries to recognize text in more accurate manner. As adaptive classifier has received some training data it has learned something new, so final phase is used to resolve various issues and to extract text from the images.

1.4 Limitations

Tesseract works best when there is a clean segmentation of the foreground text from the background. In practice, it can be extremely challenging to guarantee these types of setup. There area variety of reasons you might not get good quality output from Tesseract like if the image has noise on the background. The better the image quality the better the recognition result. It requires a bit of pre-processing to improve the OCR results, images need to be scaled appropriately, have as much image contrast as possible, and the text must be horizontally aligned. Tesseract OCR is quite powerful but does have the following limitations

- The OCR is not as accurate as some commercial solutions available to us.
- Doesn't do well with images affected by artifacts including partial occlusion, distorted perspective, and complex background.
- It is not capable of recognizing handwriting.
- It may find gibberish and report this as OCR output.
- If a document contains language outside of those given in the LANG arguments, results may be poor.
- It is not always good at analysing the natural reading order of the documents. For example, it may fail to recognize that a document contains two columns, and may try to join text across columns.

- Poor quality scans may produce poor quality OCR.
- It does not expose information about what font family text belongs to.
- Structuring the data involves more than just OCR. When users take a picture of their ID document with their smartphones or webcam, multiple steps are required to extract and structure the information.
- OCR must combine with image rectification
- Images with colored backgrounds can be problematic for OCR
- Glare and blur can cause mistakes

CHAPTER 2 – LITERATURE SURVERY

2.1 Literature Survey

Character recognition is not a new problem but its roots can be traced back to systems before the inventions of computers. The earliest OCR systems were not computers but mechanical devices that were able to recognize characters, but very slow speed and low accuracy. In 1951, M. Sheppard invented a reading and robot GISMO that can be considered as the earliest work on modern OCR. GISMO can read musical notations as well as words on a printed page one by one. However, it can only recognize 23 characters. The machine also has the capability to copy a typewritten page.

J. Rainbow, in 1954, devised a machine that can read uppercase typewritten English characters, one per minute. The early OCR systems where criticised due to errors and slow recognition speed. Hence, not much research efforts were put on the topic during 60's and 70's. The only developments were done by government agencies and large corporations like banks, newspapers, airlines etc.

Because of the complexities associated with recognition, it was felt that there should be standardized OCR fonts for easing the task of recognition for OCR. Hence, OCRA and OCRB were developed by

ANSI and EMCA in 1970, that provided comparatively acceptable recognition rates.

During the past thirty years, substantial research has been done on OCR. This has led to the emergence of document image analysis (DIA), multi-lingual, handwritten and omni-font OCRs. Despite these extensive research efforts, the machine's ability to reliably read text is still far below the human. Hence, current OCR research for diverse style documents printed/written in unconstrained environments. There has not been availability of any open source or commercial software available for complex languages like Urdu, Sindhi etc.

CHAPTER 3 – PROBLEM STATEMENT

2.1 Problem Statement

The conversion of images to editable documents has always been a cumbersome task, many times we need to convert images to documents which we can edit and then use the same for some kind of work. Extracting important textual matter from the images can prove to be extremely beneficial sometimes especially when you have to extract the data to edit, store, and distribute the same.

For example, if a college starts collecting entries for its annual magazine and even after specifying that the uploads should be done in document formats some students uploaded their entries in image formats. Now, the student members of the magazine committee will have to extract the data in text format from the submitted images by reading and replicating the content on a new word document file for further use, in order to eliminate this cumbersome process of manually extracting and replicating the text from the images one could think of using an OCR for the same, and that is exactly what IED would be used in place of, it is an offline ready-to-use tool which not only motivates students to get onto the hands-on with OCR technologies but also serves as a local tool which could be used to avoid problems as faced by the magazine committee members.

CHAPTER 4 – PROPOSED WORK

4.1 Proposed Work

- This is a service provided based locally - run web app for the conversion of images containing textual matter into editable documents
- It's an open – source tool, under the MIT licence, maintained & updated periodically with bug fixes and other updates on GitHub
- It has a user-friendly Web Interface created using Python, Flask, HTML5, CSS3 & JavaScript
- This versatile portal can be run on assorted browsers like Google Chrome, Mozilla Firefox etc.
- Since it's an open – source project, anyone following the contribution guidelines is free to contribute

4.2 Project Feasibility

An important outcome of preliminary investigation is the determination that the system requested in feasible i.e. preliminary investigation examine project feasibility – the likelihood of the system being useful to the organization and individuals

Tests of feasibility are –

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility
- Schedule Feasibility

Technical Feasibility - A system is said to be technically feasible only if the system can be developed using the existing technology. Our system uses Python 3, Flask, HTML5, CSS3 & JavaScript or its designing and implementation. Once developed, it needs very little efforts for maintenance. This makes our project technically feasible.

Economic Feasibility – This test is carried out to determine the costs of conducting a full system investigation, costs of required hardware and software and the benefits in the form of reduced costs. Our system is designed to reduce the usage of stationary to a very minimal levels, which reduced the implementation costs, increased economic feasibility. Also, the development phase does not involve any special requirements. Therefore, the benefits in deploying the system are substantial.

Operational Feasibility – A system is said to be operationally feasible only if it can be run on systems that meet the organizations' operating requirements. Our system is totally local and does not require internet access, it can be used in local machines without internet and still provide with the best conversions, it is a simple and user – friendly web application that can easily be worked with. Hence, our system is operationally feasible.

Schedule Feasibility – A project will fail if it takes too long to be completed before it is useful. Schedule feasibility is a measure of how reasonable the project timeline is. Due to the system's simplicity and modularity it can be implemented very quickly. This capability of our system allows us to implement it at a high success rate. Hence, it is schedule feasible.

CHAPTER 5 – DESIGN & DEVELOPMENT

4.1 System Analysis

To build an efficient and reliable system for this domain it is necessary to gather some ground facts about the issues that the system might face such as upload errors, unknown file type, unsupported file format etc. Therefore, we gather all the relevant facts about various issues that may be possible during the implementation of the project, so that we can solve them accordingly.

This information is combined and put together in the form of a logical working model. This model will help us to start the development of the project. The initial development involved deciding the layout of the website i.e. the UI of the website and the next phase involves the UX i.e. the user experience that our website will provide and the ease with which the users of our website will be able to interact with the application with ease, after the UI/UX being completed we move on to the core functionality of the project, the conversion of image files to documents using Tesseract-OCR engine and then creation of document files using the python library "python-docx".

4.2 System Design

The system design starts by converting the logical model of the system into a working physical model.

The physical model represents the various operations that take place in the system and the components that are involved like its various alert messages, input fields, upload, download, reset & cancel upload buttons and their functioning.

The system design focuses on the actual layout of the system that was proposed, it serves as the base for further development of the system.

CHAPTER 6 – IMPLEMENTATION

6.1 Requirements

6.1.1 Hardware Requirements

- Processor: Intel Dual Core or above
- Processor Speed: 1.0 GHz or above
- RAM: 1 GB
- HDD Space: 512MB

6.1.2 Software Requirements

- Languages: JavaScript, Python 3, HTML, CSS
- VCS: Git Bash, GitHub, GitHub Desktop, GitHub Projects
- Browser: Google Chrome
- Python Libraries: pytesseract, python-docx, os
- Frameworks: Flask, Bootstrap, Slim, Popper

HTML – HyperText Mark-up Language is the standard mark-up language for structuring the web pages and web interfaces of various applications. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

CSS – it is a style sheet language used for describing the presentation of a document written in a mark-up language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML & JavaScript.

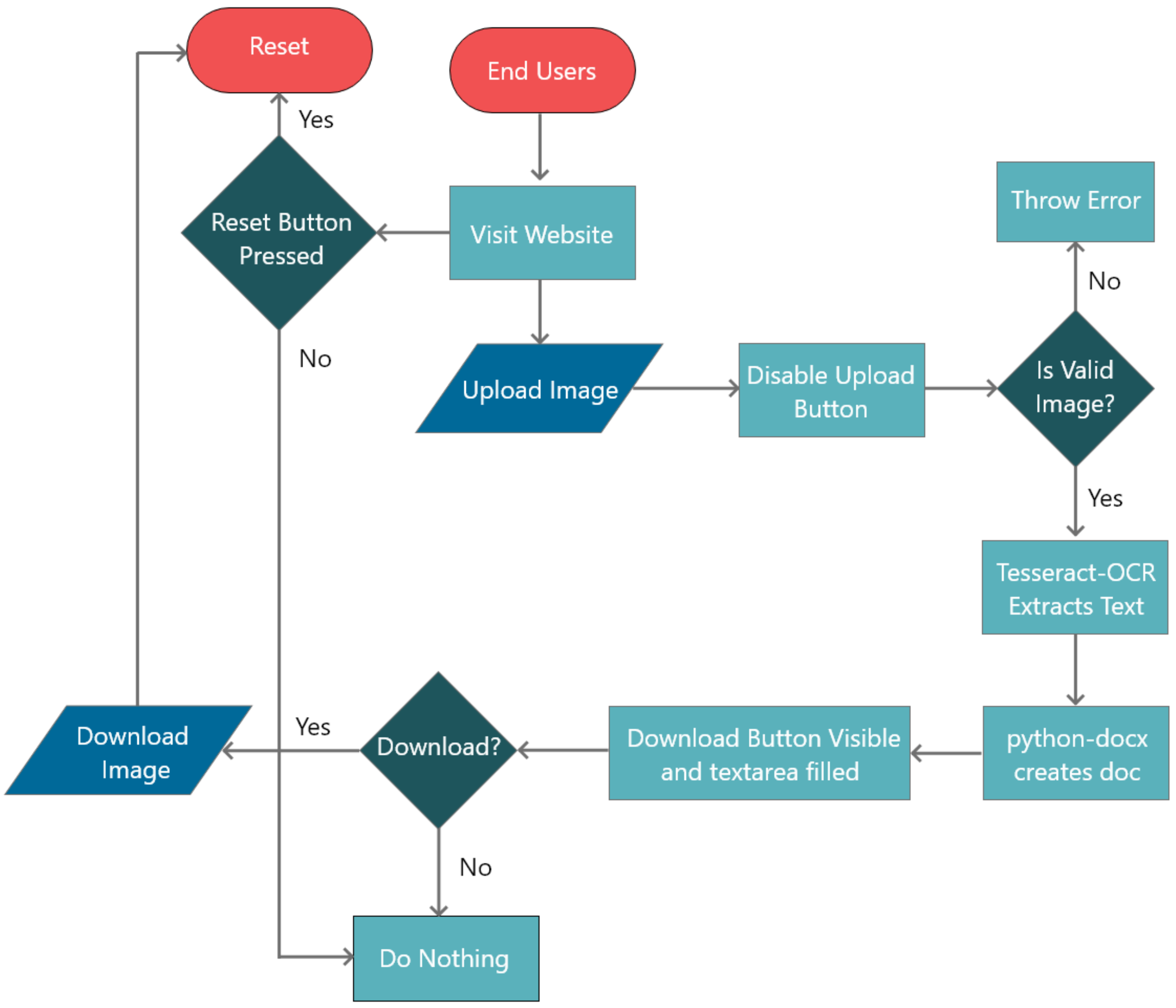
JavaScript – it is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-base object-orientation, and first-class functions.

Python – it is an interpreted, high-level, general-purpose programming language, created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Git – it's a distributed version control system (VCS) for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data-integrity, and support for distributed, non-linear workflows.

Flask – it's a micro web-framework written in Python. It is classified as micro-framework because it does not require particular tools or libraries.

6.2 Data Flow Diagram



6.3 Source Code

app.py (core logic)

```python
import os
from flask import Flask, render_template, request, jsonify, make_response
import pytesseract as tess
from PIL import Image
from docx import Document
from flask.helpers import send_from_directory
from os import abort
app = Flask(__name__)

UPLOAD_FOLDER = 'uploads/'
DOCS_FOLDER = 'docs/'

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['DOCS_FOLDER'] = DOCS_FOLDER


@app.route('/get-file/<file_name>')
def get_file(file_name):
    try:
        return send_from_directory(app.config['DOCS_FOLDER'], filename=fi
le_name, as_attachment=True)
    except FileNotFoundError:
        abort(404)



@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        file = request.files["file"]

        # Converting image to text
        img = Image.open(file)
        text = tess.image_to_string(img)

        # Saving uploaded file
        filename = file.filename.split('.')[0]
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

        # Creating .docx file
        doc = Document()
        para = doc.add_paragraph(text)
        doc.save(f"docs/{filename}.docx")

        # Preparing a response
        res = make_response(jsonify(
            {"message": f"{file.filename} uploaded", "filename": f"{filen
ame}", "text": text}))
        return res

    return render_template('index.html', name="Index")
```

index.html (web interface + script)

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
        <title>ITED</title>
        <link rel="stylesheet" href="../static/index.css" />
        <link rel="stylesheet" href="../static/bootstrap.css" />
    </head>
    <body>
        <h1 class="jumbotron text-center main-heading">
            Images to Editable Documents Converter
        </h1>
        <div class="container">
            <div class="row">
                <div class="col">
                    <div class="mb-3 mt-3">
                        <div class="form-group mb-3">
                            <div class="custom-file">
                                <input
                                    type="file"
                                    class="custom-file-input"
                                    id="file-input"
                                    oninput="inputFilename();"
                                />
                                <label
                                    for="file-input"
                                    id="file-input-label"
                                    class="custom-file-label"
                                    >Select File</label
                                >
                            </div>
                        </div>

                        <button
                            onclick="upload('{{ request.url }}')"
                            id="upload-btn"
                            class="btn btn-secondary"
                        >
                            Upload File
                        </button>
                        <button
                            onclick="hardReset()"
                            class="btn btn-secondary float-right"
                        >
```

```
                            Reset
                        </button>
                        <button
                            class="btn btn-secondary d-none"
                            id="loading-btn"
                            type="button"
                            disabled
                        >
                            <span
                                class="spinner-border spinner-border-sm"
                                role="status"
                                aria-hidden="true"
                            >
                            </span>
                            Uploading...
                        </button>

                        <button
                            class="btn btn-secondary d-none"
                            id="cancel-btn"
                            type="button"
                        >
                            Cancel Upload
                        </button>

                        <button
                            class="btn btn-secondary d-none"
                            id="download-btn"
                            type="button"
                        >
                            <a
                                href=""
                                class="text-white download-link"
                                download
                            >
                                Download Document
                            </a>
                        </button>
                    </div>
                    <div id="progress-wrapper" class="d-none">
                        <label id="progress-status"></label>
                        <div class="progress mb-3">
                            <div
                                id="progress"
                                class="progress-bar"
                                role="progressbar"
                                aria-valuenow="25"
                                aria-valuemin="0"
                                aria-valuemax="100"
                            ></div>
                        </div>
                    </div>
                    <div id="alert-wrapper"></div>
                </div>
            </div>
        </div>
        <div class="container">
            <div class="col">
                <div class="row">
                    <textarea
                        id="text-output"
                        class="w-100"
                        rows="12"
                    ></textarea>
                </div>
            </div>
        </div>

        <script src="../static/slim.min.js"></script>
        <script src="../static/popper.min.js"></script>
        <script src="../static/bootstrap.min.js"></script>
        <script>
            const progress = document.querySelector("#progress");
            const progressWrapper = document.querySelector("#progress-
wrapper");
            const progressStatus = document.querySelector("#progress-
status");

            const uploadBtn = document.querySelector("#upload-btn");
            const loadingBtn = document.querySelector("#loading-btn");
            const cancelBtn = document.querySelector("#cancel-btn");
            const downloadBtn = document.querySelector("#download-btn");

            const alertWrapper = document.querySelector("#alert-wrapper");

            const input = document.querySelector("#file-input");
            const fileInputLabel = document.querySelector("#file-input-
label");
            const textOutput = document.querySelector("#text-output");

            const showAlert = (message, alert) => {
                alertWrapper.innerHTML = `
                <div class="alert alert-${alert} alert-
dismissible fade show" role="alert">
                    <span>${message}</span>
                    <button type="button" class="close" data-
dismiss="alert" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
                `;
            };

            const inputFilename = () => {
                fileInputLabel.innerHTML = input.files[0].name;
            };

            const upload = (url) => {
                if (!input.value) {
                    showAlert("No File Selected", "warning");
                    return;
                }

                const data = new FormData();
                const request = new XMLHttpRequest();
                request.responseType = "json";

                alertWrapper.innerHTML = "";

                input.disabled = true;
                uploadBtn.classList.add("d-none");
                loadingBtn.classList.remove("d-none");
                cancelBtn.classList.remove("d-none");
                progressWrapper.classList.remove("d-none");

                const file = input.files[0];
                const filename = file.name;
                const filesize = file.size;

                document.cookie = `filesize=${filesize}`;

                data.append("file", file);

                request.upload.addEventListener("progress", (e) => {
                    const loaded = e.loaded;
                    const total = e.total;

                    let percentComplete = (loaded / total) * 100;

                    progress.setAttribute(
                        "style",
                        `width: ${Math.floor(percentComplete)}%`
                    );
```

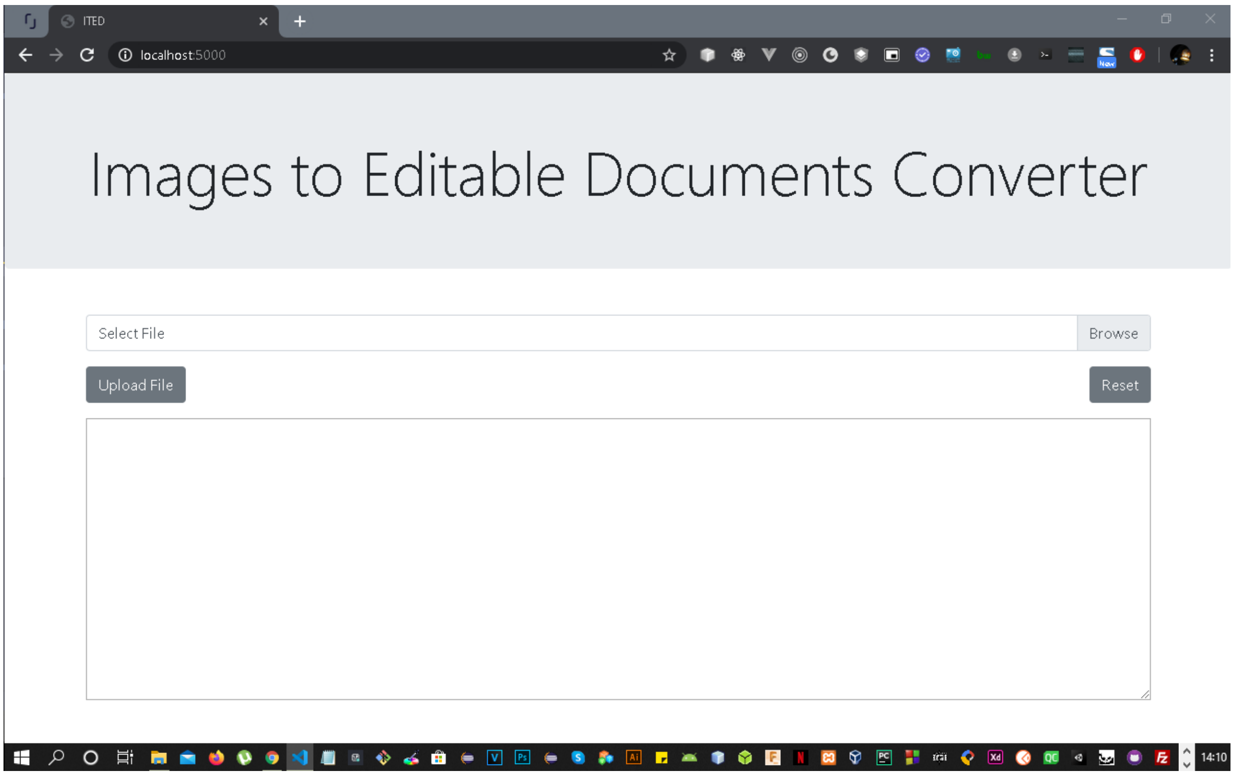
```
        progressStatus.innerText = `${Math.floor(
            percentComplete
        )}% uploaded`;
    });

    request.addEventListener("load", (e) => {
        if (request.status == 200) {
            showAlert(`${request.response.message}`, "success");
            downloadBtn.classList.remove("d-none");
            textOutput.innerText = request.response.text;
            document
                .querySelector(".download-link")
                .setAttribute(
                    "href",
                    `/get-file/${request.response.filename}.docx`


                );
        } else {
            showAlert("Error Uploading File", "danger");
        }

        reset();
    });

    request.addEventListener("error", (e) => {
        reset();

        showAlert("Error Uploading File", "danger");
    });

    request.addEventListener("abort", (e) => {
        reset();
        showAlert("Upload Cancelled", "primary");
    });

    request.open("post", url);
    request.send(data);

    cancelBtn.addEventListener("click", () => {
        request.abort();
    });
};

const hardReset = () => {
    input.value = null;
    input.disabled = false;
    cancelBtn.classList.add("d-none");
```

```
                loadingBtn.classList.add("d-none");
                uploadBtn.classList.remove("d-none");
                textOutput.innerText = "";
                alertWrapper.innerHTML = "";
                downloadBtn.classList.add("d-none");

                progressWrapper.classList.add("d-none");
                progress.setAttribute("style", "width: 0%");
                fileInputLabel.innerText = "Select File";
            };

            const reset = () => {
                input.value = null;
                input.disabled = false;
                cancelBtn.classList.add("d-none");
                loadingBtn.classList.add("d-none");
                uploadBtn.classList.remove("d-none");

                progressWrapper.classList.add("d-none");
                progress.setAttribute("style", "width: 0%");
                fileInputLabel.innerText = "Select File";
            };
        </script>
    </body>
</html>
```

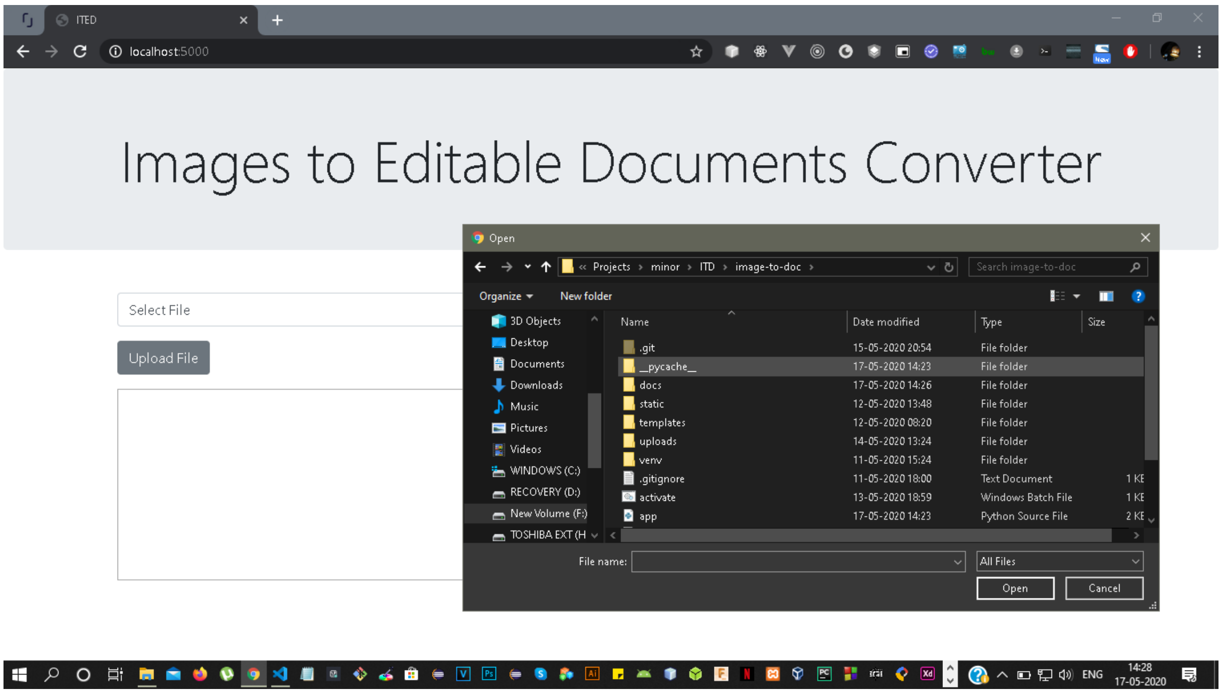
index.css (stylesheet)

```
.main-heading {
    font-weight: 300;
    font-size: 4em;
}

footer {
    position: absolute;
    bottom: 0;
}

.download-btn:hover {
    cursor: not-allowed;
}

.download-link:hover {
    text-decoration: none;
}
```

6.4 Output

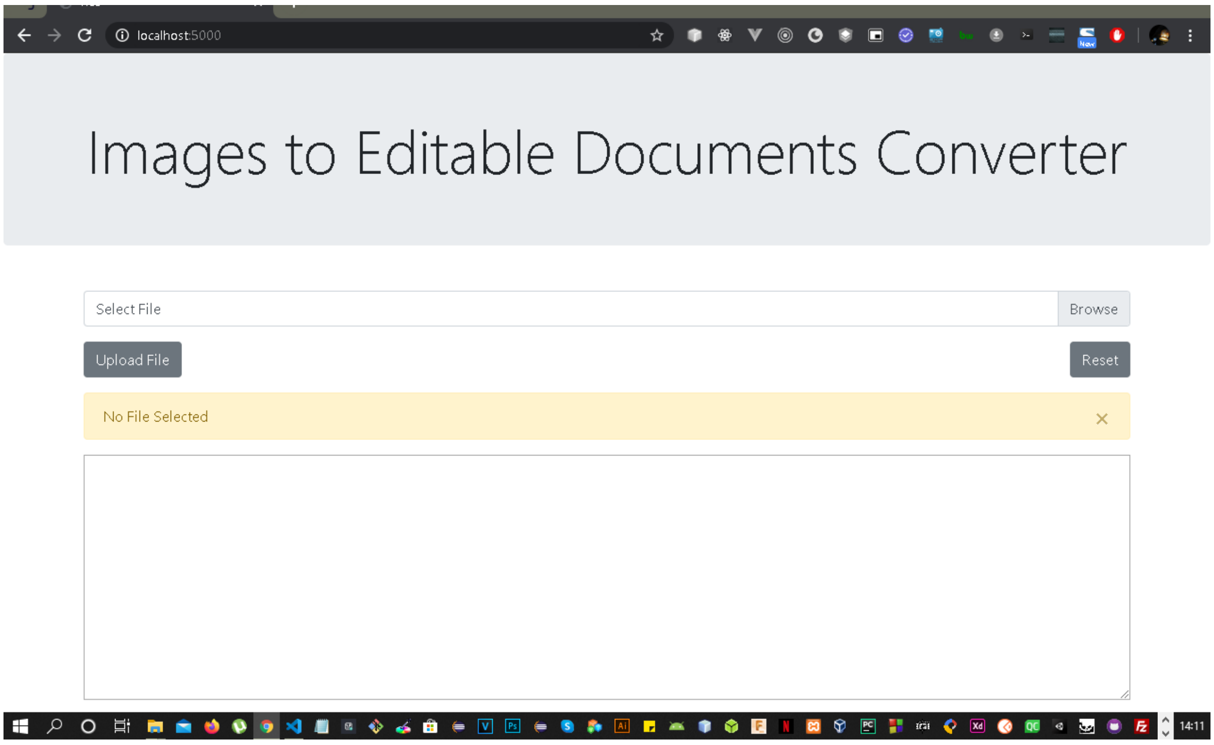


The main web interface of our app



Name	Date modified	Type	Size
.git	15-05-2020 20:54	File folder	
__pycache__	17-05-2020 14:23	File folder	
docs	17-05-2020 14:26	File folder	
static	12-05-2020 13:48	File folder	
templates	12-05-2020 08:20	File folder	
uploads	14-05-2020 13:24	File folder	
venv	11-05-2020 15:24	File folder	
.gitignore	11-05-2020 18:00	Text Document	1 KE
activate	13-05-2020 18:59	Windows Batch File	1 KE
app	17-05-2020 14:23	Python Source File	2 KE

Option to Browse & Upload Image Files


localhost:5000

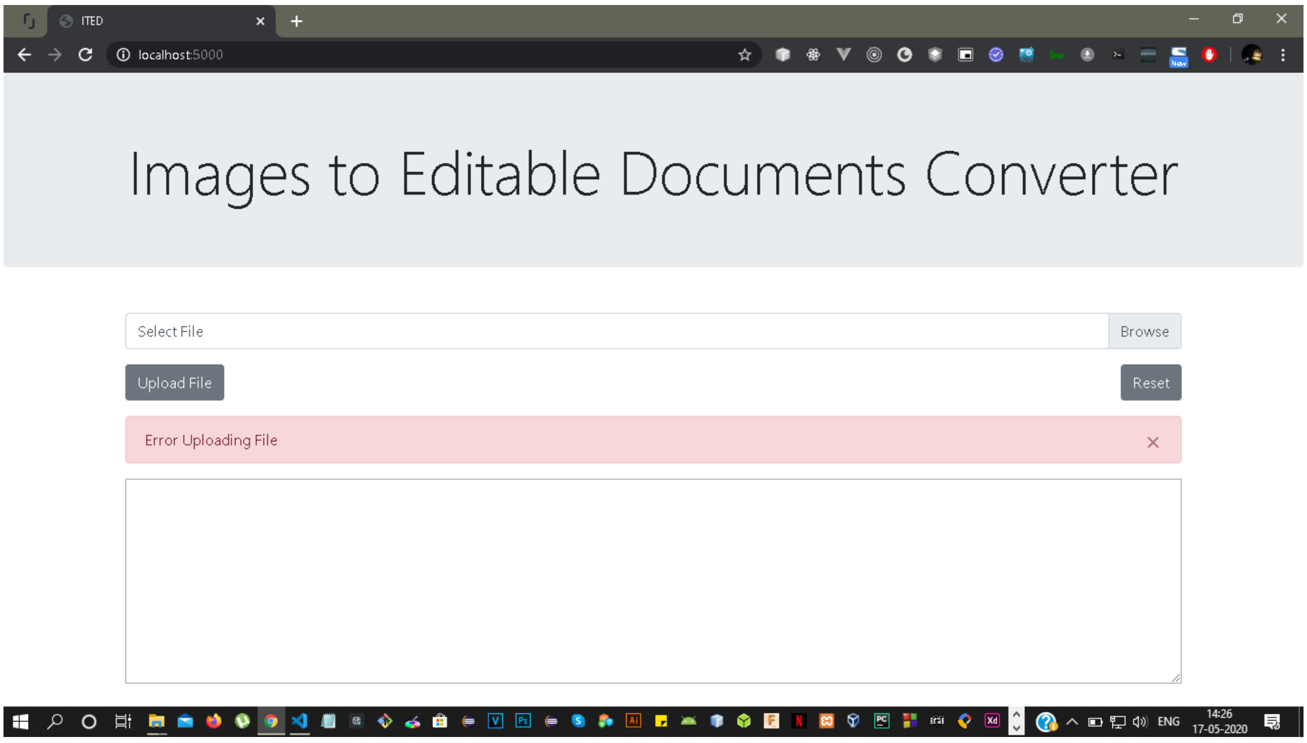
Images to Editable Documents Converter

Select File Browse

Upload File Reset

No File Selected ×

No File Selected Error


ITED

localhost:5000

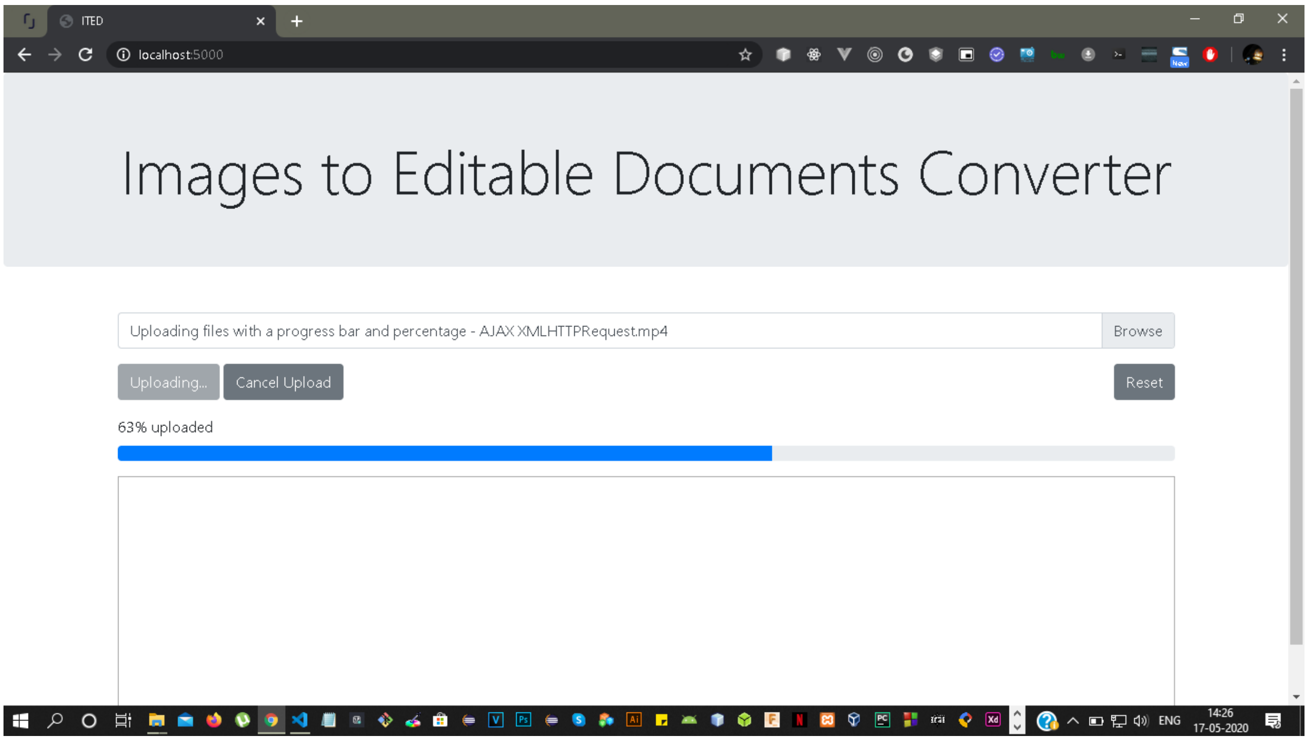
Images to Editable Documents Converter

Select File Browse

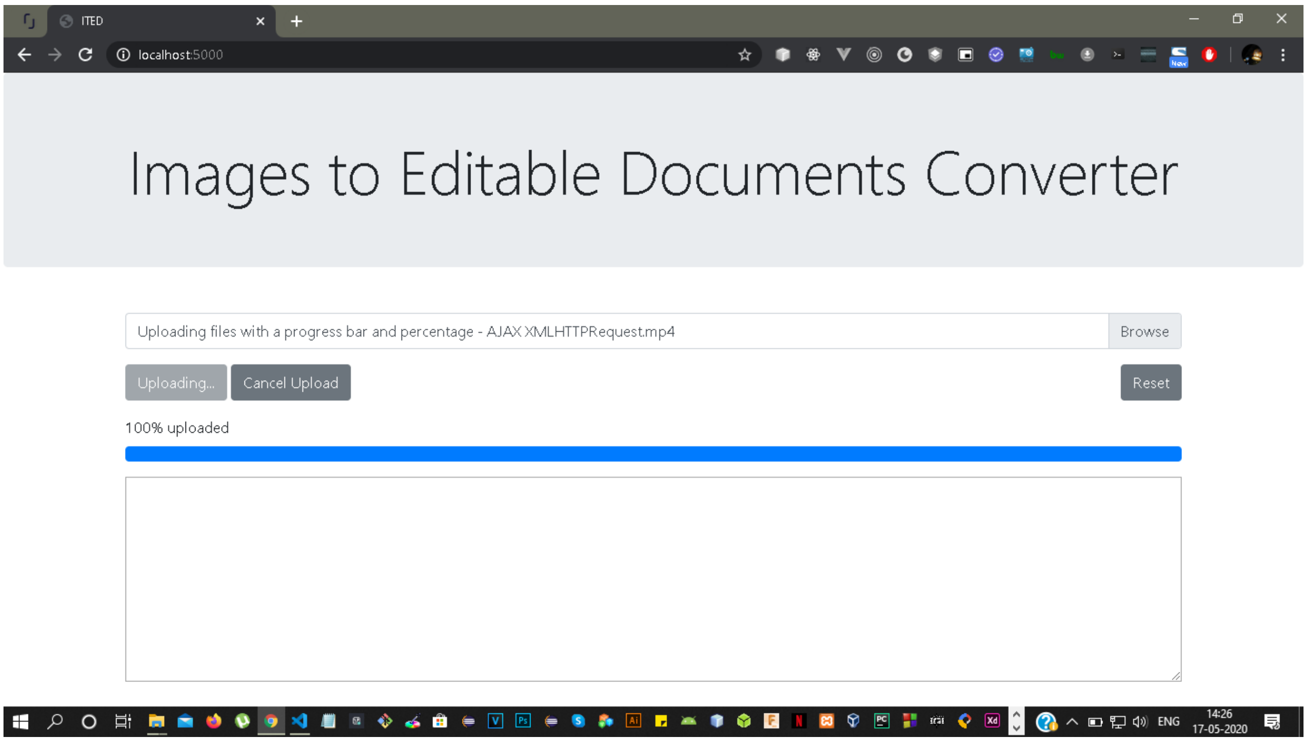
Upload File Reset

Error Uploading File ×

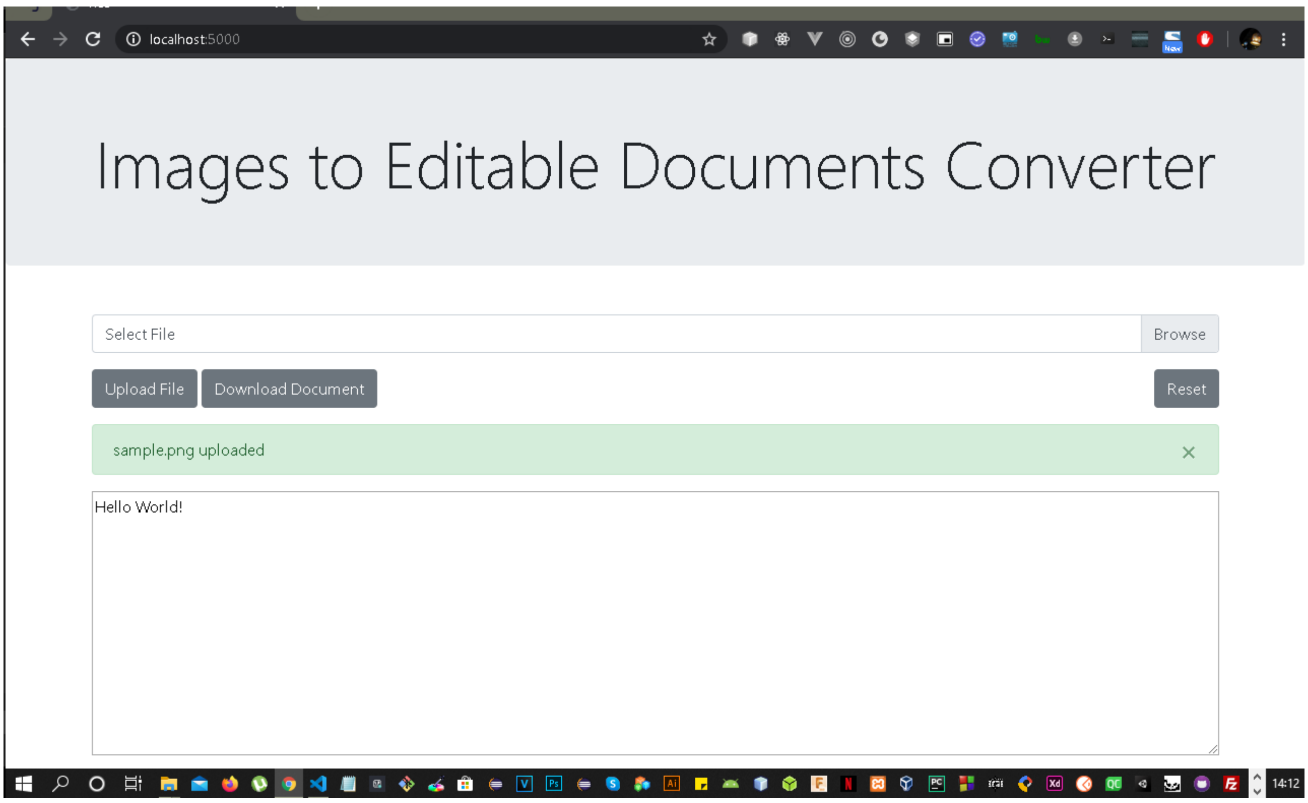
Error Uploading File Error


ITED
localhost:5000
Images to Editable Documents Converter
Uploading files with a progress bar and percentage - AJAX XMLHTTPRequest.mp4
Browse
Uploading...
Cancel Upload
Reset
63% uploaded
14:26
17-05-2020

Uploading File Progress Bar with Cancel Upload Button


ITED
localhost:5000
Images to Editable Documents Converter
Uploading files with a progress bar and percentage - AJAX XMLHTTPRequest.mp4
Browse
Uploading...
Cancel Upload
Reset
100% uploaded
14:26
17-05-2020

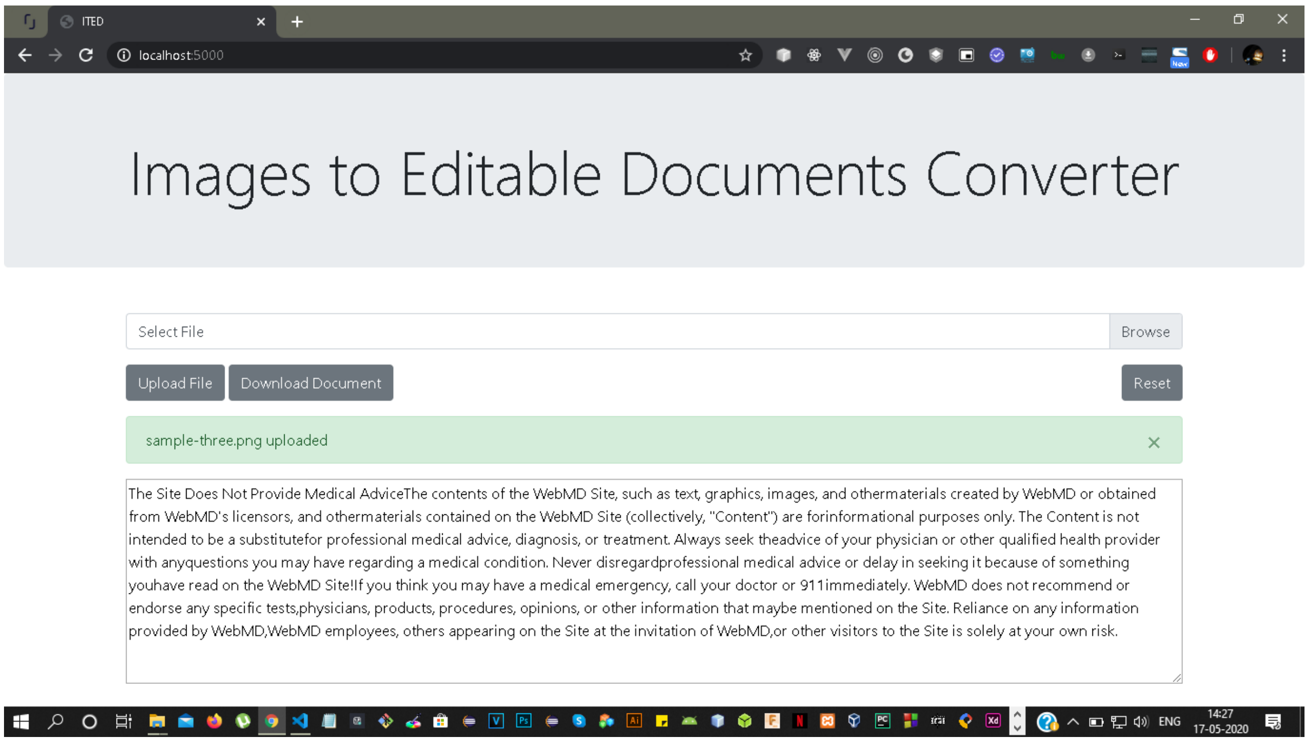
100% Upload Completed



sample.png uploaded

```
Hello World!
```

File Upload Success Message Alert with Download Button



sample-three.png uploaded

```
The Site Does Not Provide Medical AdviceThe contents of the WebMD Site, such as text, graphics, images, and othermaterials created by WebMD or obtained
from WebMD's licensors, and othermaterials contained on the WebMD Site (collectively, "Content") are forinformational purposes only. The Content is not
intended to be a substitutefor professional medical advice, diagnosis, or treatment. Always seek theadvice of your physician or other qualified health provider
with anyquestions you may have regarding a medical condition. Never disregardprofessional medical advice or delay in seeking it because of something
youhave read on the WebMD Site!If you think you may have a medical emergency, call your doctor or 911immediately. WebMD does not recommend or
endorse any specific tests,physicians, products, procedures, opinions, or other information that maybe mentioned on the Site. Reliance on any information
provided by WebMD,WebMD employees, others appearing on the Site at the invitation of WebMD,or other visitors to the Site is solely at your own risk.
```

File Uploaded & Converted with Text Output

6.5 Implementation of Software System

After the design of the system is developed, the web app components start to come together, the correct languages for the app are selected. The algorithms developed earlier are now implemented in the selected languages and tools, and checked for any undesired behavior. Its compatibility is matched with various hardware environments.

6.6 Testing

The website is checked for its accuracy, efficiency, and reliability. The testing is carried out by uploading sample images containing textual matter on the web portal and the output document is downloaded via the same web app portal and is matched with the uploaded images. In order to ensure reliability, images with larger amounts of text are also testing and the converted document is compared with the input images (stress testing).

The system should meet the proposed specifications which would help in determining its working limits. The testing is performed to determine how much the system can endure before crashing. After these tests, if changes are to be made to it, it's done in the current process so that the system can be reliable even while being developed.

CHAPTER 7 – RESULT

7.1 Result

The project is working as expected and it would be greatly helpful to all those students who wish to start learning about the field of OCR and computer vision, moreover, it's going to be extremely helpful for all users who wish to have a handy locally-run tool for their image conversions and creation of documents from existing image files. Also, the example of the magazine committee is just a drop in the ocean which has many such problems as faced by them. The project was tested with many images and their outputs have been quite accurate as expected.

CHAPTER 8 – CONCLUSION

8.1 Conclusion

In this work, we present a ready-to-use, locally run images to editable documents converter with a very user – friendly web app interface which could be easily setup and run locally on the user's machine. To conclude the whole work, we were able to not only recognize the complex working process and algorithms behind the functioning of an OCR engine. Also, we explored the various future possibilities that could be made true with time like ICR (Intelligent Character Recognition), IWR (Intelligent Word Recognition) & HTR (Handwritten Text Recognition).

8.1 Future Enhancements

Detection of handwritten text i.e. HTR would be the next advancement we could be waiting for, and that kind of software application would have huge impact on a lot of working systems such as education system, printing and drafting, data entry etc. For example, HTR could be used to recognize handwritten answer sheets and evaluation of the same using developed software apps.

CHAPTER 9 - REFERENCES

- https://en.wikipedia.org/wiki/Optical_character_recognition
- https://nanonets.com/blog/ocr-with-tesseract/#trainingtesseractoncustomdata
- http://www.assistivetechnology.vcu.edu/wp-content/uploads/sites/1864/2013/09/pxc3882784.pdf
- https://arxiv.org/ftp/arxiv/papers/1710/1710.05703.pdf
- https://en.wikipedia.org/wiki/Cascading_Style_Sheets
- https://en.wikipedia.org/wiki/HTML
- https://en.wikipedia.org/wiki/JavaScript
- https://en.wikipedia.org/wiki/Git
- https://en.wikipedia.org/wiki/Flask_(web_framework)