**NAME -RUPSA RUPA PRIYADARSHINI OJHA**

**Position- Web Development Intern**

**PROJECT ON pdf to Wordfile converter**

**Table of Contents**

1. Abstract 2
2. Introduction 3
3. Benefits of pdf to Wordfile 4
4. Tools and Technologies Used 5
5. Code 6
6. Working of the pdf to Wordfile  (Screenshots) 9
7. Conclusion 11

**ABSTRACT:**

The PDF to Word File Conversion Project aims to develop a software tool that enables users to convert PDF documents into editable Word files efficiently. This project focuses on delivering a seamless conversion process while maintaining the original content's formatting and layout. Converting PDF files to Word documents poses challenges due to the inherent differences in their file formats. PDF files are designed for fixed-format viewing and sharing, while Word files allow for content editing. The project seeks to address this gap by implementing a precise and reliable conversion algorithm. To accomplish this, the software tool will incorporate Optical Character Recognition (OCR) technology, which will analyze the visual elements of the PDF files. By recognizing text and images, the OCR algorithm will convert them into editable text and graphics suitable for Word processing. Additionally, the tool will accurately handle formatting elements, including fonts, styles, tables, and bullet points, to ensure the integrity of the original document. A rigorous testing and validation process will be implemented to ensure the quality and accuracy of the converted filesThe PDF to Word File Conversion Project offers numerous benefits to users. It provides a time-saving solution for individuals and businesses frequently requiring PDF document editing. By converting PDFs to Word files, users can easily modify content, add annotations, and collaborate on documents with others.

**Introduction**

The PDF to Word File Conversion Project is a software development endeavor that aims to create a proficient tool for converting PDF documents into editable Word files. This project addresses the need for a seamless conversion process while ensuring the preservation of the original formatting and layout of the content.

The conversion of PDF files to Word documents presents a challenge due to the inherent differences in their file formats. PDF files are designed for static document representation, while Word files allow for dynamic content editing. This project endeavors to bridge this gap by implementing an accurate and efficient conversion algorithm.

The software tool will employ cutting-edge Optical Character Recognition (OCR) technology to analyze and interpret the textual and graphical elements within the PDF files. By recognizing and extracting the text and images, the OCR algorithm will transform them into editable components suitable for Word processing. Additionally, the tool will intelligently handle formatting attributes such as fonts, styles, tables, and bullet points, thereby ensuring the faithful reproduction of the original document's visual appearance. To guarantee the reliability and accuracy of the converted files, the project will undergo a meticulous testing and validation process.

**Benefits of pdf to Wordfile project**

Editability: The project allows for converting PDF documents into editable Word files. This enables users to make changes, add or remove content, and modify the formatting of the document using widely available word processing software.

Preservation of structure: By converting PDFs to Word files, the project helps maintain the structure and layout of the original document. It retains elements such as headings, paragraphs, tables, and images, allowing for easier editing and formatting adjustments.

Text extraction: The project extracts text from PDF files, making it easier to analyze and process the textual content. It enables users to extract specific information, perform text mining, or use natural language processing techniques on the extracted text.

Compatibility: This compatibility ensures that the converted files can be easily shared and opened by others, regardless of the software they use.

Time and resource efficiency: Automating the PDF to Word conversion process through Python saves time and effort, particularly when dealing with large volumes of PDF documents.

Flexibility: The project provides flexibility by offering options to choose from different Python libraries for PDF parsing and conversion

**Tools and Technologies Used**

To create a PDF to Word file project with website building using Python in Visual Studio Code (VS Code), the following tools and technologies can be utilized:

Python: Python serves as the primary programming language for developing the project. It offers a wide range of libraries and frameworks for handling PDF files, such as PyPDF2 or pdf2docx, as well as web development frameworks like Flask or Django.

VS Code: Visual Studio Code is an integrated development environment (IDE) widely used by developers.

Flask or Django: Flask and Django are popular Python web frameworks used for building web applications. These frameworks provide a structured approach to developing web projects and include features like routing, templates, and database integration.

HTML/CSS: HyperText Markup Language (HTML) and Cascading Style Sheets (CSS) are fundamental web technologies used for building web pages and defining their visual presentation. HTML structures the content, while CSS controls the layout and styling.

PDF parsing libraries: Python libraries like PyPDF2, pdf2docx, or textract can be utilized to extract text and images from PDF files. These libraries offer functionalities to parse the content of PDFs and convert them into editable Word files.

**CODE:**

app.py

from flask import Flask, render\_template, request, send\_file

from pdf2docx import Converter

import os

app = Flask(\_\_name\_\_)

@app.route('/', methods=['GET', 'POST'])

def home():

    if request.method == 'POST':

        # Get the uploaded PDF file from the request

        pdf\_file = request.files['pdf\_file']

        # Get the output file name from the input box

        output\_file\_name = request.form['output\_file\_name']

        # Save the uploaded PDF file to a temporary location

        pdf\_path = f'tmp/{pdf\_file.filename}'

        pdf\_file.save(pdf\_path)

        # Convert PDF to DOCX

        docx\_file = convert\_pdf\_to\_docx(pdf\_path, output\_file\_name)

        # Clean up the temporary PDF file

        os.remove(pdf\_path)

        # Send the converted file as a download

        return send\_file(docx\_file, as\_attachment=True)

    # Render the home page

    return render\_template('home.html')

def convert\_pdf\_to\_docx(pdf\_path, output\_file\_name):

    # Create a temporary file with a .docx extension

    docx\_file = f'tmp/{output\_file\_name}.docx'

    # Convert the PDF file to DOCX

    cv = Converter(pdf\_path)

    cv.convert(docx\_file, start=0, end=None)

    cv.close()

    return docx\_file

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

home.html

<!DOCTYPE html>

<html>

<head>

    <title>PDF to DOCX Converter</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <style>

        body {

            padding: 50px;

        }

        h1 {

            margin-bottom: 30px;

        }

        form {

            max-width: 500px;

            margin: 0 auto;

        }

        input[type="file"] {

            display: none;

        }

        .custom-file-upload {

            display: inline-block;

            padding: 8px 12px;

            cursor: pointer;

            background-color: #337ab7;

            color: #fff;

            border-radius: 4px;

        }

        .custom-file-upload:hover {

            background-color: #286090;

        }

        input[type="text"] {

            width: 100%;

            padding: 8px;

            margin-bottom: 15px;

            border: 1px solid #ccc;

            border-radius: 4px;

            box-sizing: border-box;

        }

        input[type="submit"] {

            padding: 10px 16px;

            background-color: #337ab7;

            color: #fff;

            border: none;

            border-radius: 4px;

            cursor: pointer;

        }

        input[type="submit"]:hover {

            background-color: #286090;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1 class="text-center">PDF to DOCX Converter</h1>

        <form action="/" method="POST" enctype="multipart/form-data">

            <div class="form-group">

                <label for="pdf\_file" class="custom-file-upload">Choose PDF File</label>

                <input type="file" name="pdf\_file" id="pdf\_file" accept=".pdf" required>

            </div>

            <div class="form-group">

                <label for="output\_file\_name">Output File Name:</label>

                <input type="text" name="output\_file\_name" id="output\_file\_name" class="form-control" required>

            </div>

            <div class="form-group">

                <input type="submit" value="Convert and Download" class="btn btn-primary">

            </div>

        </form>

    </div>

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@1.16.0/dist/umd/popper.min.js"></script>

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

</body>

</html>

Styles.css

body {

    font-family: Arial, sans-serif;

    margin: 20px;

  }

  h1 {

    color: #333;

  }

  form {

    margin-top: 20px;

  }

  input[type="file"] {

    margin-right: 10px;

  }

  button[type="submit"] {

    padding: 5px 10px;

  }

  h2 {

    margin-top: 20px;

  }

  p {

    margin-bottom: 10px;

  }

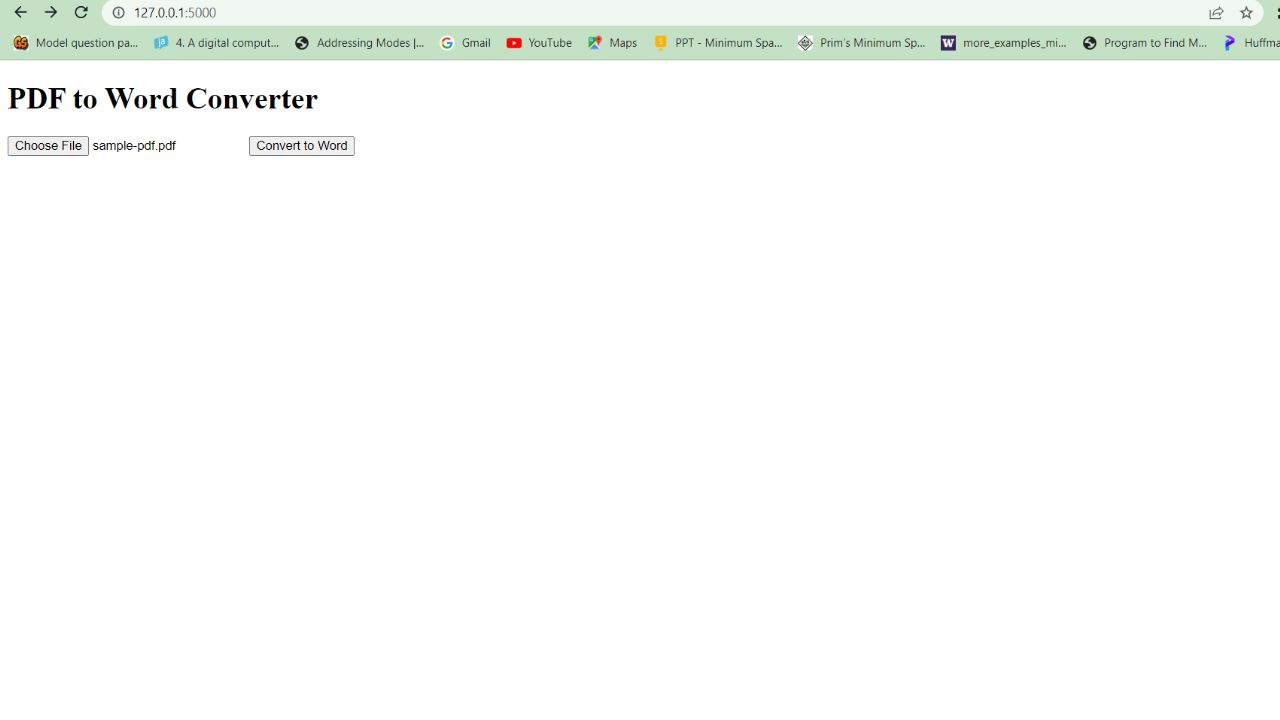
  a {

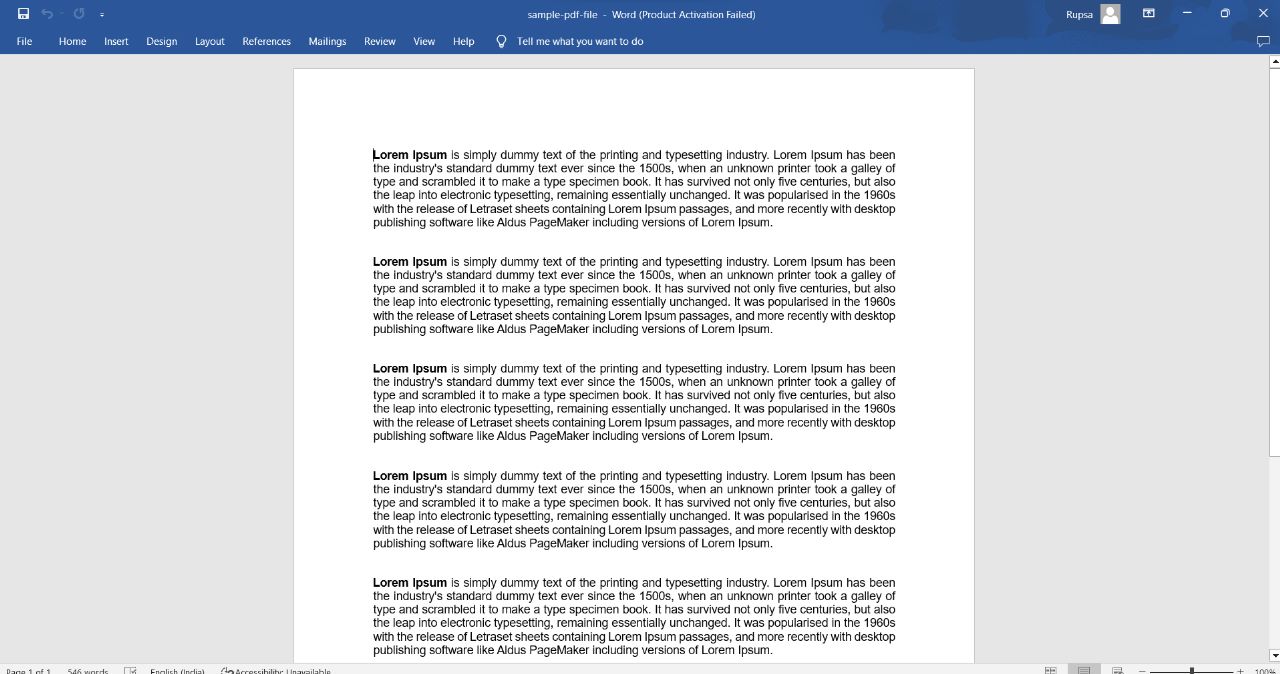
    display: block;

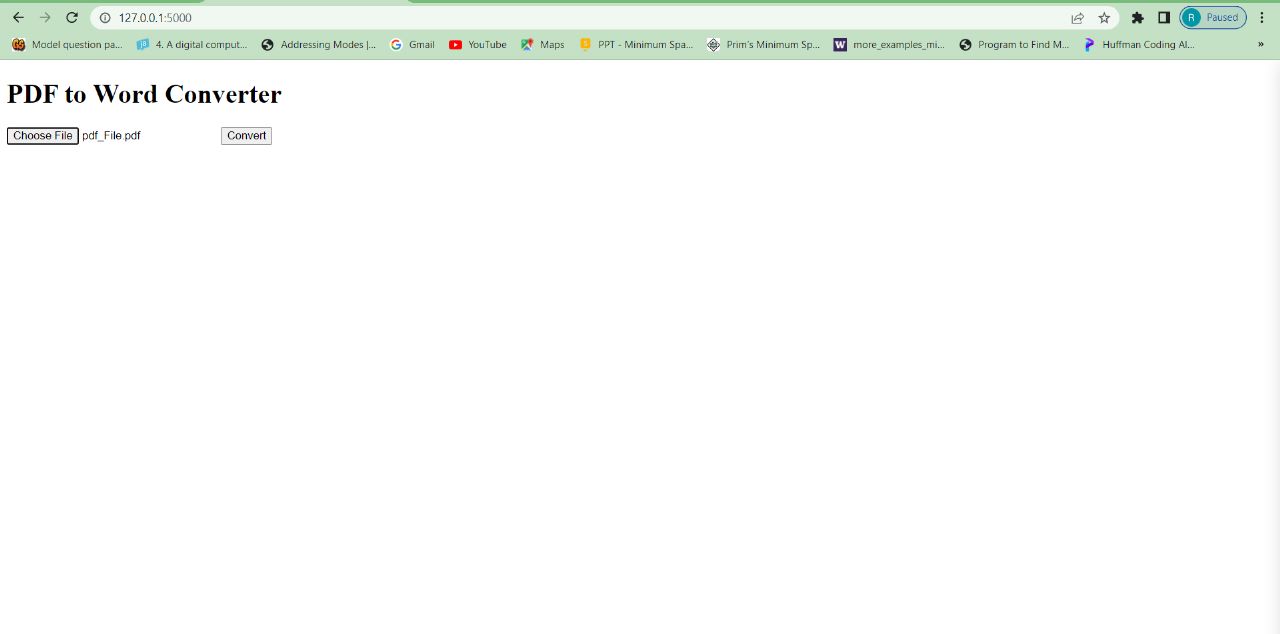
    margin-top: 10px;

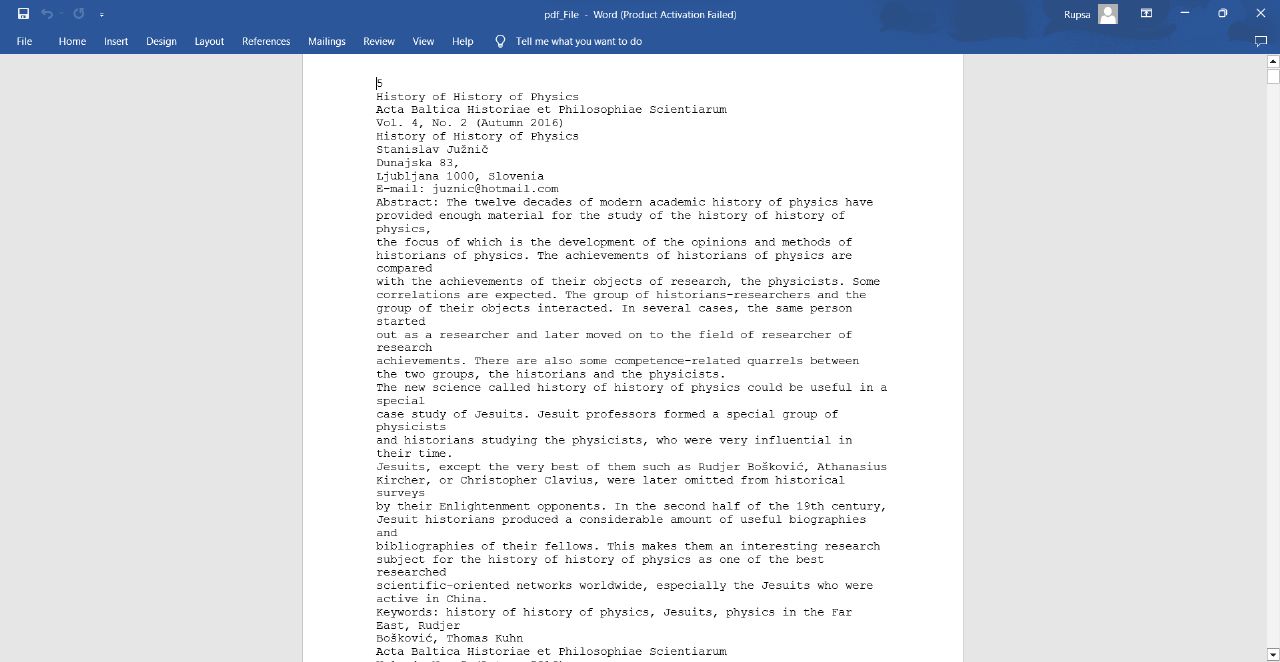
  }

**Working of the pdf to Wordfile (Screenshots)**







****

**CONCLUSION**

The PDF to Word project in Python enables the conversion of PDF documents into editable Word files using the Python programming language. This project offers a convenient solution for extracting text and other content from PDF files and saving them in a format that can be easily modified and manipulated using word processing software. To accomplish this task, several Python libraries, such as PyPDF2, textract, or pdf2docx, can be utilized. These libraries provide functionalities that allow the parsing of PDF files, extraction of text and images, and conversion into usable formats. It is important to note that the accuracy of the conversion largely depends on the complexity and formatting of the original PDF document. Simple text-based PDFs tend to convert more accurately compared to documents with intricate layouts, extensive images, or tables.