**PDF to Excel Converter Report**

**1. Introduction**

In many industries, data is often stored in PDF format, making it difficult to extract structured information for further analysis. The **PDF to Excel Converter** is a tool designed to automate this process by extracting tables from PDF files and converting them into Excel spreadsheets. This project was implemented using **Python**, utilizing the **pdfplumber** library for extracting text and tables, and **pandas** for handling structured data. Additionally, a **Streamlit web application** was developed to provide a user-friendly interface.

**2. Objectives**

The main objectives of this project are:

* To efficiently extract tabular data from PDF documents.
* To convert extracted tables into an Excel file for easy accessibility.
* To provide a simple and interactive interface using Streamlit.
* To automate data extraction, reducing manual effort and errors.

**3. Methodology**

The project follows these key steps:

**a. PDF Parsing:**

* The application uses the **pdfplumber** library to read and extract data from PDF files.
* The text is processed to identify tabular structures.

**b. Data Structuring:**

* Extracted text is organized into rows and columns.
* A mapping function ensures proper alignment of data.

**c. Excel Conversion:**

* The structured data is stored in a **pandas DataFrame**.
* The data is written to an **Excel file** using **pandas' ExcelWriter**.

**d. Streamlit Integration:**

* A **Streamlit web application** was created for easy user interaction.
* Users can upload PDF files and receive an Excel file as output.

**4. Implementation**

The main components of the implementation include:

**a. Python Script for PDF Processing:**

* The script reads PDF files and processes tables using **pdfplumber**.
* Data is structured into a proper tabular format.

**b. Web Application (app.py):**

* A **Streamlit** interface allows users to upload PDF files.
* The application processes the file and generates a downloadable Excel file.

**c. Running the Application:**

* The application can be run using the command:

streamlit run app.py

* Users can upload a PDF, and the tool will generate an Excel file with extracted tables.

**5. Results and Benefits**

* The application successfully extracts tabular data from PDFs and converts it into Excel format.
* Users can save time and effort compared to manually copying data.
* The tool provides **accuracy** in data extraction and prevents errors.
* The **Streamlit interface** makes it easy to use, even for non-programmers.

**6. Conclusion**

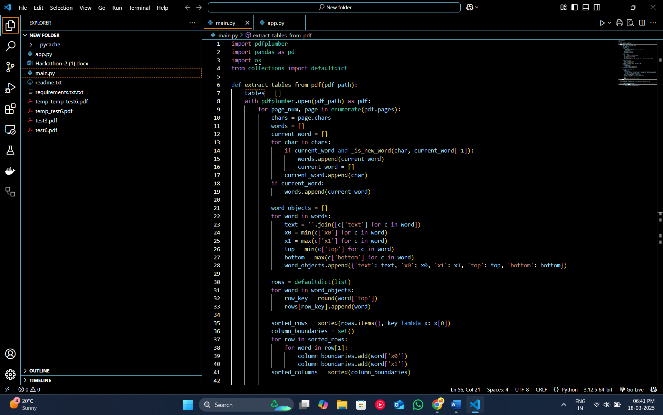
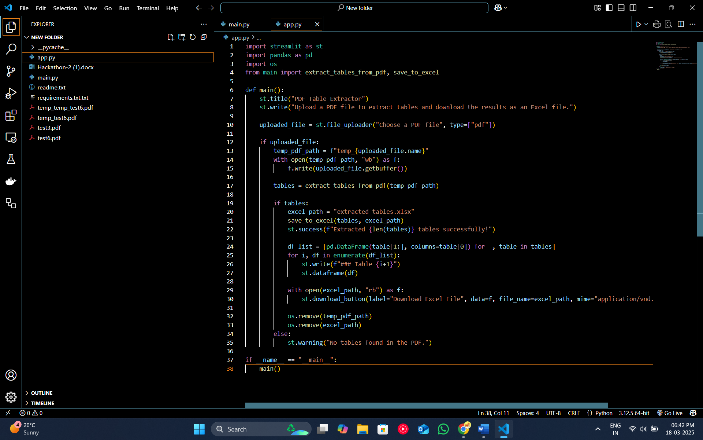
The **PDF to Excel Converter** is a valuable tool for professionals working with tabular data in PDF documents. By automating the extraction and conversion process, it enhances efficiency and minimizes manual effort. The integration of **Streamlit** makes the application accessible and easy to use. Future improvements could include support for **image-based PDFs using OCR** and **enhanced table detection algorithms**.

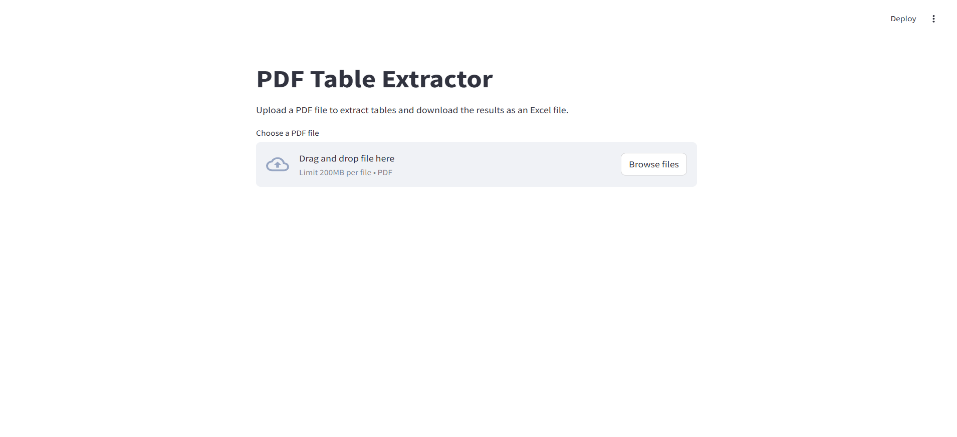
**7. Future Enhancements**

* Adding **OCR support** for scanned PDFs using Tesseract OCR.
* Improving **table detection** for more complex document layouts.
* Enhancing **user interface** with additional customization options.

This project successfully demonstrates the power of **Python** in automating data extraction and making information more accessible. It is a useful tool for professionals dealing with data extraction challenges in PDFs.

**Screenshots**

**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

Main.py Code

def extract\_tables\_from\_pdf(pdf\_path):

    tables = []

    with pdfplumber.open(pdf\_path) as pdf:

        for page\_num, page in enumerate(pdf.pages):

            chars = page.chars

            words = []

            current\_word = []

            for char in chars:

                if current\_word and \_is\_new\_word(char, current\_word[-1]):

                    words.append(current\_word)

                    current\_word = []

                current\_word.append(char)

            if current\_word:

                words.append(current\_word)

            word\_objects = []

            for word in words:

                text = ''.join([c['text'] for c in word])

                x0 = min(c['x0'] for c in word)

                x1 = max(c['x1'] for c in word)

                top = min(c['top'] for c in word)

                bottom = max(c['bottom'] for c in word)

                word\_objects.append({'text': text, 'x0': x0, 'x1': x1, 'top': top, 'bottom': bottom})

            rows = defaultdict(list)

            for word in word\_objects:

                row\_key = round(word['top'])

                rows[row\_key].append(word)

            sorted\_rows = sorted(rows.items(), key=lambda x: x[0])

            column\_boundaries = set()

            for row in sorted\_rows:

                for word in row[1]:

                    column\_boundaries.add(word['x0'])

                    column\_boundaries.add(word['x1'])

            sorted\_columns = sorted(column\_boundaries)

            table = []

            for row\_key, words in sorted\_rows:

                row\_dict = {}

                for word in words:

                    col\_start = max([c for c in sorted\_columns if c <= word['x0']], default=sorted\_columns[0])

                    col\_end = min([c for c in sorted\_columns if c >= word['x1']], default=sorted\_columns[-1])

                    col\_index = sorted\_columns.index(col\_start)

                    row\_dict[col\_index] = word['text']

                max\_col = max(row\_dict.keys()) if row\_dict else 0

                row\_data = [row\_dict.get(i, '') for i in range(max\_col+1)]

                table.append(row\_data)

            if table:

                tables.append((f"Page\_{page\_num+1}", table))

    return tables

def \_is\_new\_word(new\_char, prev\_char):

    return (new\_char['x0'] - prev\_char['x1'] > 3 or new\_char['top'] - prev\_char['top'] > 5)

def save\_to\_excel(tables, output\_path):

    with pd.ExcelWriter(output\_path) as writer:

        for name, table in tables:

            df = pd.DataFrame(table[1:], columns=table[0])

            df.to\_excel(writer, sheet\_name=name, index=False)