BLACKBOXAI

Generated on: 16/8/2025 6:25:22 pm

User

Got it ✅ — you’re describing something more advanced:

Upload a PDF on a website.

The system extracts structured fields (like name, email, invoice no., etc.) automatically and fills them in a form on the webpage.

If the PDF has images, they should be extracted and displayed inside the image section of the form.

This is essentially a PDF Form Extractor + Autofill Web Form. Let’s break it down.

BLACKBOXAI

I'll create a PDF form extractor and autofill web application that extracts structured fields from uploaded PDFs and populates a web form. This will require:

1. A PDF parsing library (pdf.js)
2. File upload handling
3. Form field extraction logic
4. Automatic form population
5. Image extraction and display

Here's the complete implementation:

`pdf-extractor.html`

```
html
```

PDF Form Extractor

Upload your PDF form and we'll automatically extract and fill the fields

Drag & drop your PDF file here

or click to browse files

Process PDF

User

I want it on Python

BLACKBOXAI

I'll create a Python version of the PDF form extractor application using Flask for the web framework and PyPDF2 for PDF processing. This will be a complete application with a similar UI but running on a Python backend.

Here's the implementation:

1. First, the main application file `app.py`:

```
 python
from flask import Flask, render_template, request, jsonify, send_from_directory
from werkzeug.utils import secure_filename
import os
import PyPDF2
from io import BytesIO
import re
from PIL import Image
import base64
from pdf2image import convert_from_bytes

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'
app.config['ALLOWED_EXTENSIONS'] = {'pdf'}
app.config['MAX_CONTENT_LENGTH'] = 16  1024  1024  # 16MB
```

Ensure upload folder exists

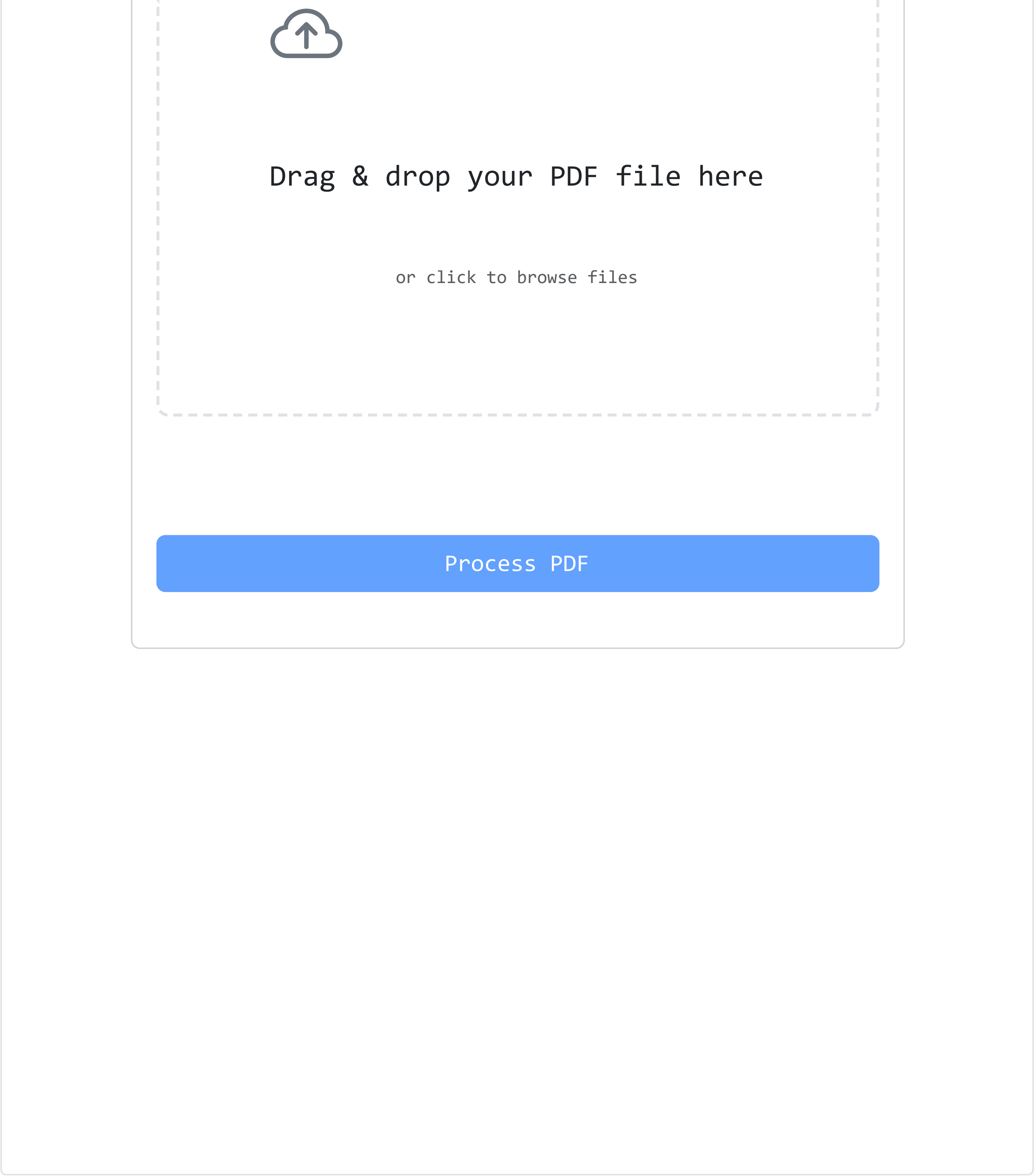
```
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)

def allowed_file(filename):
    return '.' in filename and
           filename.rsplit('.', 1)[1].lower() in app.config['ALLOWED_EXTENSIONS']

def extract_fields_from_text(text):
    """Extract common form fields patterns from text"""
    fields = {}
    patterns = [
        ('name', r'name[: ]+([^n]+)', False),
        ('email', r'email[: ]+([^n]+)', False),
        ('phone', r'phone[: ]+([^n]+)', False),
        ('address', r'address[: ]+([^n]+)', False),
        ('invoice_number', r'invoice #?[: ]+([^n]+)', False),
        ('date', r'date[: ]+([^n]+)', False),
        ('amount', r'amount[: ]+([^n]+)', False),
        ('description', r'description[: ]+([^n]+)', False)
    ]

    for name, pattern, _ in patterns:
        match = re.search(pattern, text, re.IGNORECASE)
        if match and match.group(1):
            fields[name] = match.group(1).strip()

    return fields

def process_pdf(pdf_bytes):
    """Process PDF to extract text fields and images"""
    fields = {}
    images = []

    try:
        # Extract text fields
        pdf_reader = PyPDF2.PdfReader(BytesIO(pdf_bytes))

        for page_num in range(len(pdf_reader.pages)):
            page = pdf_reader.pages[page_num]
            page_text = page.extract_text()
            page_fields = extract_fields_from_text(page_text)
            fields.update(page_fields)

            # Try to extract images (this is limited - PyPDF2 doesn't have good image 
extraction)
            if '/XObject' in page['/Resources']:
                x_object = page['/Resources']['/XObject'].get_object()
                for obj in x_object:
                    if x_object[obj]['/Subtype'] == '/Image':
                        try:
                            image_data = x_object[obj].get_data()
                            img_format = x_object[obj]['/Filter']

                            if img_format == '/FlateDecode':
                                im = Image.open(BytesIO(image_data))
                                buffer = BytesIO()
                                im.save(buffer, format="PNG")
                                img_data = buffer.getvalue()
                            else:
                                img_data = image_data

                            images.append({
                                'page': page_num + 1,
                                'data': base64.b64encode(img_data).decode('utf-8'),
                                'format': 'png' if img_format == '/FlateDecode' else img_format[1:].lower()
                            })
                        except Exception as e:
                            print(f"Error extracting image: {e}")
                            continue

    except Exception as e:
        print(f"Error processing PDF: {e}")

    return fields, images

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400

    file = request.files['file']

    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400

    if not allowed_file(file.filename):
        return jsonify({'error': 'File type not allowed'}), 400

    try:
        # Save the file temporarily
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)

        # Read the file
        with open(filepath, 'rb') as f:
            pdf_bytes = f.read()

        # Process the PDF
        fields, images = process_pdf(pdf_bytes)

        # Clean up
        os.remove(filepath)

        return jsonify({
            'success': True,
            'fields': fields,
            'images': images
        })
    except Exception as e:
        if os.path.exists(filepath):
            os.remove(filepath)
        return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

2. Now the template file `templates/index.html`:

html

PDF Form Extractor

Upload your PDF form and we'll automatically extract and fill the fields



Drag & drop your PDF file here

or click to browse files

3. Requirements file `requirements.txt` :

```
 Flask==2.3.2
PyPDF2==3.0.1
pdf2image==1.16.3
Pillow==10.0.0
```

To run this application:

1. Install the dependencies:

```
 bash
pip install -r requirements.txt
```

2. Run the application:

```
 bash
python app.py
```

3. Open your browser to http://localhost:5000

Key features of this implementation:

1. PDF text extraction using PyPDF2
2. Basic form field detection using regex patterns
3. Image extraction (though limited by PyPDF2's capabilities)
4. Modern UI with Bootstrap
5. Drag and drop file upload
6. Responsive design for all screen sizes
7. Error handling and user feedback

Note that this is a simplified implementation. In a production environment, you might want to:

- Use a more robust PDF parsing library (like pdfminer.six)
- Implement better field detection logic
- Add more error handling
- Implement user authentication
- Add database storage for processed documents
- Deploy with a production WSGI server (like Gunicorn)