Document summarization system

Directory Navigation

```
document_summarizer/
│
├── static/          # Folder for CSS (optional, for styling)
│   └── style.css    # CSS file for styling (optional)
│
├── templates/       # Folder for HTML templates
│   └── index.html   # HTML frontend for file upload
│
├── app.py           # Main Flask application (Backend)
└── summarizer.py    # PDF summarization logic (Backend logic)
```

app.py

```
from flask import Flask, render_template, request, redirect, url_for, flash
from werkzeug.utils import secure_filename
import os
from summarizer import summarize_pdf

app = Flask(__name__)

# Folder to store uploaded files
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Ensure the upload folder exists
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

# Allowed extensions for file upload
ALLOWED_EXTENSIONS = {'pdf'}

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

# Route to render the main page
@app.route('/')
def index():
    return render_template('index.html')

# Route to handle file upload and summarization
@app.route('/upload', methods=['POST'])
def upload_file():
    # Check if a file is uploaded
    if 'file' not in request.files:
        flash('No file part')
        return redirect(request.url)

    file = request.files['file']

    # Check if file has a valid filename and is a PDF
    if file.filename == '':
        flash('No selected file')
        return redirect(request.url)

    if file and allowed_file(file.filename):
        # Save the uploaded file
        filename = secure_filename(file.filename)
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)

        # Summarize the PDF
        summary = summarize_pdf(file_path, word_limit=200)

        # Return the summary to the frontend
        return render_template('index.html', summary=summary)
    flash('Allowed file type is PDF')
    return redirect(request.url)

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

summarize.py

```
from PyPDF2 import PdfReader
from transformers import T5ForConditionalGeneration, T5Tokenizer

# Function to extract text from PDF
def extract_text_from_pdf(pdf_file):
    reader = PdfReader(pdf_file)
    text = ""

    # Iterate through all the pages and extract text
    for page_num in range(len(reader.pages)):
        page = reader.pages[page_num]
        text += page.extract_text()

    return text

# Load the pre-trained T5 model and tokenizer
model = T5ForConditionalGeneration.from_pretrained('t5-small')
tokenizer = T5Tokenizer.from_pretrained('t5-small')

# Function to summarize text with a word limit (approx. 200 words)
def summarize(text, max_words=200):
    # Tokenize the input text
    inputs = tokenizer.encode("summarize: " + text, return_tensors="pt", max_length=512, truncation=True)

    # Generate summary with beam search
    summary_ids = model.generate(
        inputs,
        max_length=max_words,      # Control the maximum length of the summary (200 words approx.)
        min_length=50,             # Minimum length of the summary
        length_penalty=2.0,        # Controls summary length preference
        num_beams=4,               # Beam search for generating higher quality summaries
        early_stopping=True        # Stops generation when optimal
    )

    # Decode and return the summary
    summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
    return summary

# Function to extract and summarize PDF content
def summarize_pdf(pdf_file, word_limit=200):
    # Extract text from the PDF
    text = extract_text_from_pdf(pdf_file)

    # Summarize the extracted text
    summary = summarize(text, max_words=word_limit)

    return summary
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document Summarization System</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
```

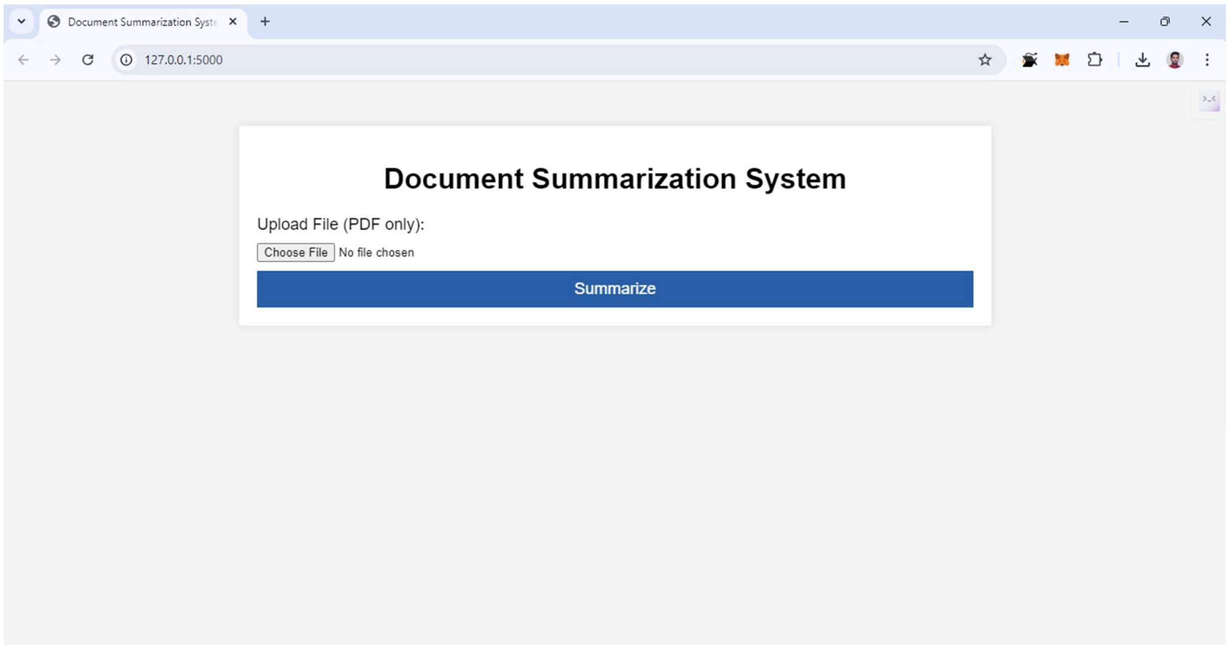
```
        <h1>Document Summarization System</h1>
        <form method="POST" action="{{ url_for('upload_file') }}" enctype="multipart/form-data">
            <label for="pdf_file">Upload File (PDF only):</label>
            <input type="file" id="file" name="file" accept="application/pdf" required>
            <button type="submit">Summarize</button>
        </form>

        {% if summary %}
        <div class="summary-section">
            <h2>Generated Summary:</h2>
            <p>{{ summary }}</p>
        </div>
        {% endif %}
    </div>
</body>
</html>
```

style.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
}

.container {
    max-width: 800px;
    margin: 50px auto;
    background: white;
    padding: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
    text-align: center;
}

label {
    font-size: 18px;
}

input[type="file"] {
    display: block;
    margin: 10px 0;



}

button {
    display: block;
    width: 100%;
    padding: 10px;
    background-color: #285fa7;
    color: white;
    border: none;
    cursor: pointer;
    font-size: 18px;
}

button:hover {
    background-color: #218838;
}

.summary-section {
    margin-top: 20px;
    background-color: #f9f9f9;
    padding: 15px;
    border-radius: 5px;
```

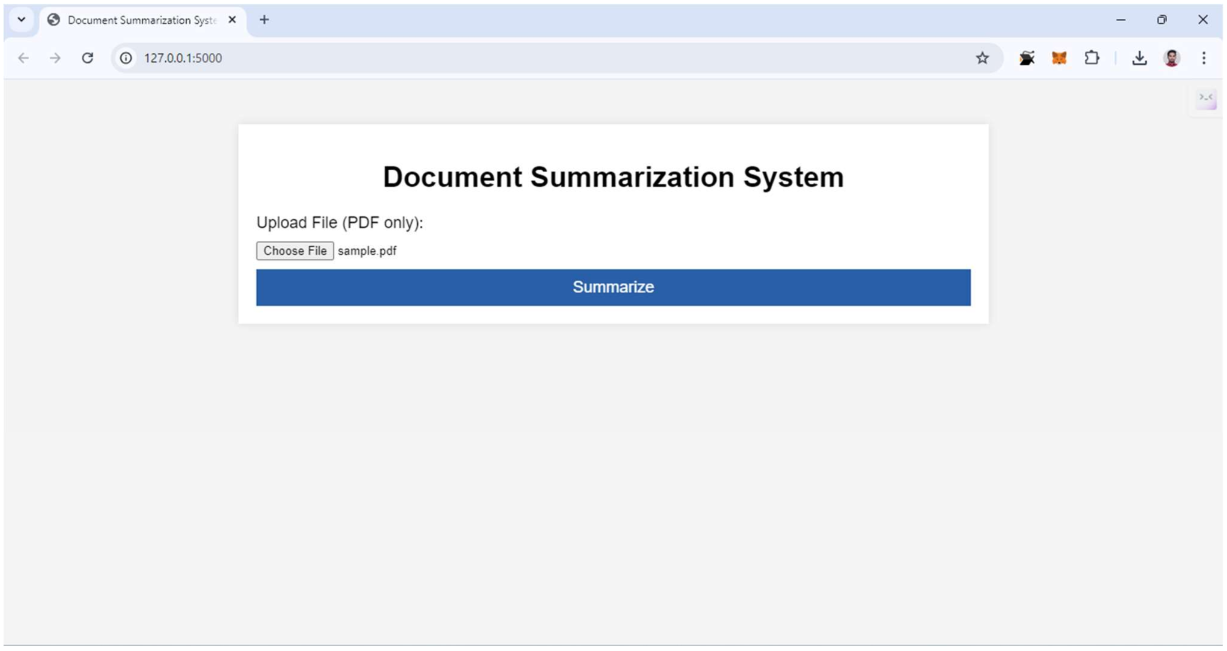
```
}

.summary h2 {
    margin-top: 0;
}
```

Output Screenshot



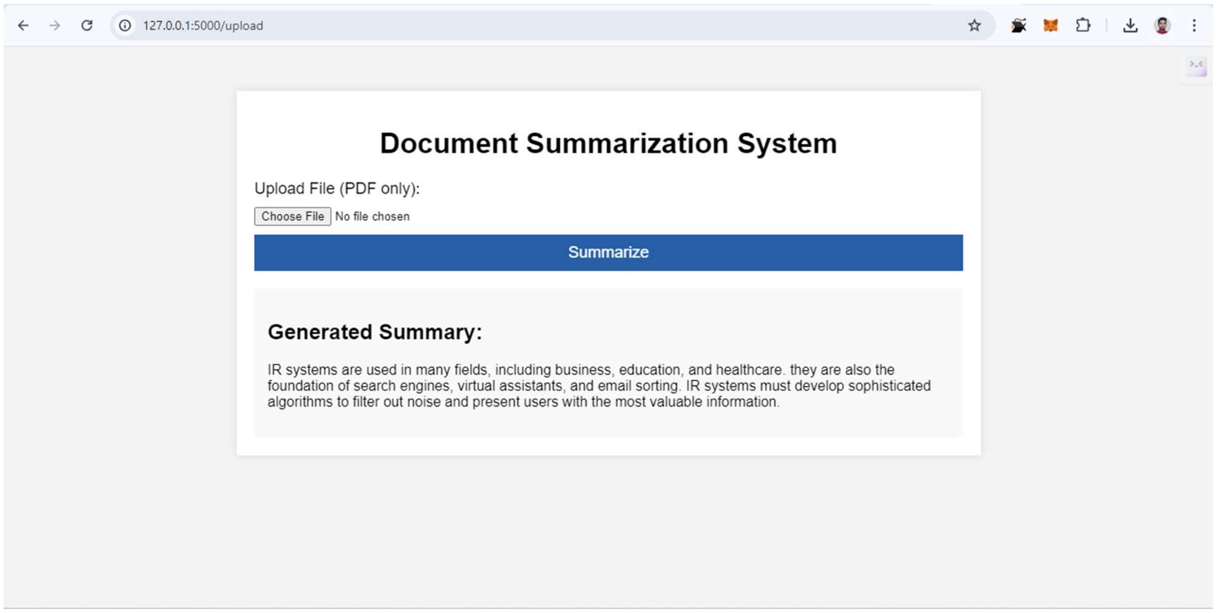
Document Summarization System

Upload File (PDF only):



Document Summarization System

Upload File (PDF only):



Document Summarization System

Upload File (PDF only):

Generated Summary:

IR systems are used in many fields, including business, education, and healthcare. they are also the foundation of search engines, virtual assistants, and email sorting. IR systems must develop sophisticated algorithms to filter out noise and present users with the most valuable information.