Document Summarizer - Project Documentation

Tech Stack

Frontend:

- HTML5, CSS3, JavaScript

- Marked.js (for Markdown rendering)

Backend:

- Flask (Python Web Framework)

- REST API using Flask Routes

LLM Integration:

- Groq APIs (e.g., Claude, LLaMA)

File Processing:

- PyMuPDF (PDFs)

- python-docx (DOCX files)

Project Flow

1. User opens the homepage where they can either upload a document or paste raw text.

2. Upon submission, a POST request is sent to the /summarize endpoint.

3. Backend Flask route handles the request:

   - If a file is uploaded, it's saved and processed using PyMuPDF or python-docx.

   - If raw text is provided, it's sent directly to the summarizer.

4. The extracted text is sent to the Groq LLM API via utils/summarizer.py.

5. The summary (in Markdown format) is returned as a JSON response.

6. Frontend receives the response and renders it using marked.js.

API Overview

POST /summarize

- Accepts: multipart/form-data with either a document or raw text

- Returns: JSON with summarized Markdown text

Example Request:

- document: uploaded file (.pdf, .docx, .txt)

- raw_text: user-pasted text

Example Response:


```
{
  "summary": "# Summary\n- Key points\n- Bullet list..."
}
```