**PROJECT DOCUMENTATION SMARTSDLC**

**TEAM ID: NM2025TMID02757**

**Team Leader**

**V.Roshini Priya**

**Gmail:** [**roshinijancy0510@gmail.com**](mailto:roshinijancy0510@gmail.com)

**Team Members**

**1.Janani.M (gmail:** [**jananisri1818@gmail.com**](mailto:jananisri1818@gmail.com)**)**

**2.Shalini.G-(gmail: shalini27n05@gmail.com)**

**3.Yogavani.P- (gmail:** [**abinaya4222@gmail.com**](mailto:abinaya4222@gmail.com)**)**

**Project Overview**

**1. Uploading PDF in Gradio**

pdf\_upload = gr.File(label="upload policy PDF", file\_types=[".pdf"])

* In the **Policy Summarization** tab, the user can **upload a PDF**.
* This file is passed to the function policy\_summarization.

**2. Extracting Text from PDF**

def extract\_text\_from\_pdf(pdf\_file):

if pdf\_file is None:

return ""

try:

pdf\_reader = pyPDF2.PdfReader(pdf\_file)

text = ""

for page in pdf\_reader.pages:

text += page.extract\_text() + "\n"

return text

except Exception as e:

return f"Error reading PDF: {str(e)}"

* Uses **PyPDF2** to read the uploaded file.
* Loops through all pages, extracts text, and concatenates them.
* If PDF is empty or encrypted → returns an error message.

**3. Policy Summarization**

def policy\_summarization(pdf\_file, policy\_text):

if pdf\_file is not None:

content = extract\_text\_from\_pdf(pdf\_file)

summry\_prompt = f"Summarize the following policy document and extract the most important points,key provisions,and implications:\n\n{content}"

else:

summry\_prompt = f"Summarize the following policy document and extract the most important points,key provisions,and implications:\n\n{policy\_text}"

return generate\_response(summry\_prompt, max\_length=1200)

* If a **PDF is uploaded** → extract text using extract\_text\_from\_pdf().
* If no PDF is uploaded → take raw text from the textbox.
* Creates a **prompt** for the AI model:

"Summarize the following policy document and extract key points, provisions, implications…"

* Passes this to generate\_response() for summarization.

**4. Generate Response**

def generate\_response(prompt, max\_length=1024):

inputs = tokenizer(prompt, return\_tensors="pt", truncation=True, max\_length=512)

if torch.cuda.is\_available():

inputs = {k: v.to(model.device) for k, v in inputs.items()}

with torch.no\_grad():

outputs = model.generate(

\*\*inputs,

max\_length=max\_length,

temperature=0.7,

do\_sample=True,

pad\_token\_id=tokenizer.eos\_token\_id

)

response = tokenizer.decode(outputs[0], skip\_special\_tokens=True)

response = response.replace(prompt, "").strip()

return response

* Takes the **prompt** from summarization.
* Tokenizes it → feeds to **Granite 3.2 model**.
* Model generates summarized text.
* Cleans output (removes prompt text).

**5. Display in Gradio**

summarize\_btn.click(policy\_summarization, inputs=[pdf\_upload, policy\_text\_input], outputs=summary\_output)

* When user clicks **Summarize Policy**,
  + PDF (or typed text) → policy\_summarization()
  + Summarized response → shown in summary\_output textbox.

Top of Form

Bottom of Form