SmartSDLC – AI-Enhanced Software Development Lifecycle

Generative AI with IBM

SmartSDLC

1. Introduction

- ***Project Title:*** *SmartSDLC*
- ***Team Leader :*** *Kaviya M S*

- ***Team member :*** *Jency Mary A*
- ***Team member :*** *Sri Sara R*
- ***Team member :*** *Yogeshwari M*

2. Project Overview

The Eco Assistant & Policy Analyzer is an AI-driven application that helps users:

1. *Generate Eco-Friendly Tips: Provides practical and actionable sustainable living suggestions based on environmental problem keywords.*
2. *Summarize Policy Documents: Extracts key points, provisions, and implications from environmental and sustainability-related policy documents (uploaded as PDF or pasted text).*

This project integrates IBM Granite (LLM model) with Gradio UI to provide a simple and interactive interface for both individuals and policymakers.

3. Architecture

Components:

1. ***Model & Tokenizer***
    - *Uses Hugging Face's AutoTokenizer and AutoModelForCausalLM with IBM Granite model (ibm-granite/granite-3.2-2b-instruct).*
    - *Runs on CPU/GPU with PyTorch backend.*
2. ***Core Functions***
    - *generate_response(prompt): Generates AI-powered responses from user prompts.*
    - *extract_text_from_pdf(pdf_file): Reads and extracts text from uploaded PDF files using PyPDF2.*
    - *eco_tips_generator(problem_keywords): Produces eco-friendly living suggestions.*

- *policy_summarization(pdf_file, policy_text): Summarizes policy documents.*

3. ***Gradio Interface***
    - *Tab 1: Eco Tips Generator → User enters keywords (e.g., solar, plastic, energy saving).*
    - *Tab 2: Policy Summarization → User uploads PDF or pastes policy text to get a concise summary.*

4. Setup Instructions

Prerequisites

- *Python ≥ 3.9*
- *pip package manager*
- *GPU with CUDA (optional for faster inference**)***

5. Folder Structure

EduTutor-AI/

│── app.py # Main application script

│── requirements.txt # Dependencies

│── docs/ # Documentation files

│── models/ # Pretrained model references

│── utils/ # Helper functions (if extended later)

6. Running the Application

- Launch the Gradio interface by running app.py

- Navigate between Concept Explanation and Quiz Generator tabs

- Input the desired topic and view the AI-generated output in real time

7. Authentication

Currently, the project runs locally with no authentication.
If deployed (e.g., on cloud or Hugging Face Spaces), authentication can be added using:

- *API Key-based Authentication (restrict model access).*
- *Gradio Auth (username/password login).*
- *OAuth (Google/GitHub login) for enterprise use.*

8. API Documentation

Although the project runs via Gradio UI, backend functions can be exposed as APIs:

POST /generate-tips

Input: Environmental keywords

Output: AI-generated eco tips

POST /summarize-policy

Input: Policy text or PDF

Output: Summary with key points & implications

POST /extract-pdf

Input: PDF file

Output: Extracted plain text

9. User Interface

Built with Gradio Tabs:

Tab 1 – Eco Tips Generator

- *Input: Keywords (e.g., "solar", "plastic", "energy saving").*
    - *Output: Sustainable living tips (bulleted, structured).*
- *Button: Generate Eco Tips.*

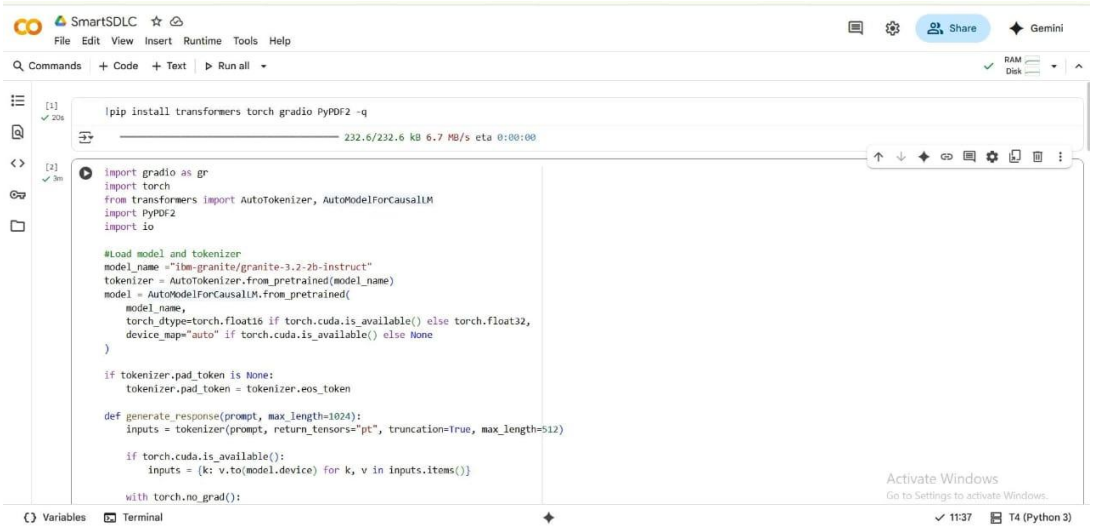
Tab 2 – Policy Summarization

- *Input Option 1: Upload Policy PDF.*
- *Input Option 2: Paste policy text.*
- *Output: Summarized policy with key provisions and implications.*

10. Testing

- *Run the app locally or on Colab.*
- *Test with valid and invalid inputs:*
    - *Empty city name or query should return validation messages.*
    - *Vary temperature and max tokens to observe output diversity.*
- *Verify outputs are relevant and coherent.*

10. Screenshots



```
        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

    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)}"

    def eco_tips_generator(problem_keywords):
        prompt = f"Generate practical and actionable eco-friendly tips for sustainable living related to: {problem_keywords}. Provide specific solutions ans suggestions:"
        return generate_response(prompt, max_length=1000)

    def policy_summarization(pdf_file, policy_text):
```



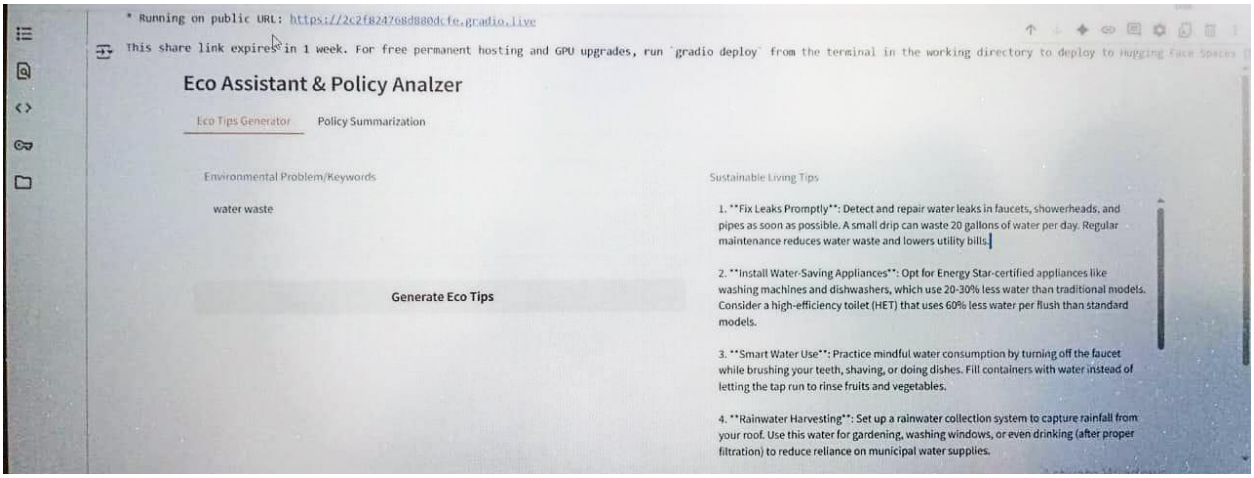
```
    def policy_summarization(pdf_file, policy_text):
        # Get text from PDF or direct input
        if pdf_file is not None:
            content = extract_text_from_pdf(pdf_file)
            summary_prompt = f"Summarize the following policy document and extract the most important points, key provision, and implications:\n\n{content}"
        else:
            summary_prompt = f"Summarize the following policy document and extract the most important points, key provision, and implications:\n\n{policy_text}"

        return generate_response(summary_prompt, max_length=1200)

    # create Gradio interface
    with gr.Blocks() as app:
        gr.Markdown("# Eco Assistant & Policy Analzer")

        with gr.Tabs():
            with gr.TabItem("Eco Tips Generator"):
                with gr.Row():
                    with gr.Column():
                        keywords_input = gr.Textbox(
                            label="Environmental Problem/Keywords",
                            placeholder="e.g., plastic, solar, water waste, energy saving...",
                            lines=3
                        )
                        generate_tips_btn = gr.Button("Generate Eco Tips")

                    with gr.Column():
                        tips_output = gr.Textbox(label="Sustainable Living Tips", lines=15)

                generate_tips_btn.click(eco_tips_generator, inputs=keywords_input, outputs=tips_output)

            with gr.TabItem("Policy Summarization"):
```



```
            with gr.TabItem("Policy Summarization"):
                with gr.Row():
                    with gr.Column():
                        pdf_upload = gr.File(label="Upload Policy PDF", file_types=[".pdf"])
                        policy_text_input = gr.Textbox(
                            label="Or paste policy text here",
                            placeholder="Paste policy document text...",
                            lines=5
                        )
                        summarize_btn = gr.Button("Summarize Policy")

                    with gr.Column():
                        summarize_output = gr.Textbox(label="Policy Summary & Key Points", lines=20)

                summarize_btn.click(policy_summarization, inputs=[pdf_upload, policy_text_input], outputs=summarize_output)

    app.launch(share=True)
```

Output:


/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 255kB/s]
vocab.json: 777k/? [00:00<00:00, 9.69MB/s]
merges.txt: 442k/? [00:00<00:00, 9.38MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 37.3MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 3.70kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 42.9kB/s]
config.json: 100% 786/786 [00:00<00:00, 30.9kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.71MB/s]
Fetching 2 files: 100% 2/2 [02:05<00:00, 125.31s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [02:05<00:00, 41.9MB/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 64.7MB/s]
Loading checkpoint shards: 100% 2/2 [00:19<00:00, 8.27s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 14.6kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://2c2f824768d880dcfe.gradio.live
Activate Windows
Go to Settings to activate Windows.


* Running on public URL: https://2c2f824768d880dcfe.gradio.live
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces
Eco Assistant & Policy Analzer
Eco Tips Generator
Policy Summarization
Environmental Problem/Keywords
water waste
Generate Eco Tips
Sustainable Living Tips
1. **Fix Leaks Promptly**: Detect and repair water leaks in faucets, showerheads, and pipes as soon as possible. A small drip can waste 20 gallons of water per day. Regular maintenance reduces water waste and lowers utility bills.
2. **Install Water-Saving Appliances**: Opt for Energy Star-certified appliances like washing machines and dishwashers, which use 20-30% less water than traditional models. Consider a high-efficiency toilet (HET) that uses 60% less water per flush than standard models.
3. **Smart Water Use**: Practice mindful water consumption by turning off the faucet while brushing your teeth, shaving, or doing dishes. Fill containers with water instead of letting the tap run to rinse fruits and vegetables.
4. **Rainwater Harvesting**: Set up a rainwater collection system to capture rainfall from your roof. Use this water for gardening, washing windows, or even drinking (after proper filtration) to reduce reliance on municipal water supplies.


Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://2c2f824768d880dcfe.gradio.live
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces
Eco Assistant & Policy Analzer
Eco Tips Generator
Policy Summarization
Upload Policy PDF
Drop File Here
- or -
Click to Upload
Policy Summary & Key Points

11.Known Issues:

- *Lower accuracy for rare/niche environmental topics due to dataset limits.*
- *No offline mode; dependent on stable internet and Hugging Face API.*
- *No direct database or EHR/EMR integration for storing or fetching policies.*
- *PDF text extraction may fail with scanned or image-based PDFs.*
- *Limited UI (basic Gradio interface, no advanced visualization).*

12. Future Enhancements

- *Add offline model support to reduce API dependency.*
- *Improve PDF parsing with OCR for image-based files.*
- *Integrate voice input for accessibility.*
- *Build a dashboard for saving, organizing, and comparing summaries.*
- *Expand datasets for rare environmental and policy topics.*
- *Add cloud storage & security (encrypted summaries & tips).*
- *Enable integration with government APIs for live policy updates.*