Project Documentation:

Eco Assistant & Policy Analyzer

Team Member - Akshara UG
Team Member - Harishini R R
Team Member - Deepika P
Team Member - Aishwariya R

Introduction:

Sustainability and policy awareness are two crucial pillars in addressing environmental challenges in the modern world. Individuals often seek actionable eco-friendly practices for daily life, while policymakers, researchers, and students require tools to digest complex policy documents efficiently. The Eco Assistant & Policy Analyzer is an AI-powered application designed to bridge this gap. Built using Gradio, PyTorch, and Transformers, this project provides two key functionalities:

Eco Tips Generator:

It offers practical, sustainable living tips based on user-provided environmental keywords.

Policy Summarization Tool:
It extracts, summarizes, and presents key points from lengthy policy documents, either via direct text input or PDF uploads.

This project aims to leverage the capabilities of natural language processing (NLP) to promote sustainability, simplify policy analysis, and encourage environmentally conscious decision-making.

Objectives:
The main objectives of the project are to:

- **Promote Sustainable Living**
  Provide users with simple, actionable, and personalized eco-friendly tips to tackle everyday environmental challenges.

- **Simplify Policy Understanding**
  Break down complex policy documents into concise summaries highlighting provisions, implications, and key insights.

- **Enhance Accessibility**
  Make policy analysis and sustainability recommendations accessible to students, researchers, organizations, and individuals with little technical expertise.

- **Leverage AI for Social Good**
  Utilize large language models (LLMs) like IBM Granite for generating reliable, human-like responses to real-world challenges.

- **Tools and Technologies Used**
  Python – Core programming language for the application.

- **Gradio** – For building a user-friendly, interactive web interface.

- **Transformers (Hugging Face)** – To load and utilize the IBM Granite LLM for generating eco tips and summarizations.

- **PyTorch** – Backend for running deep learning models.

- **PyPDF2** – For reading and extracting text content from PDF documents.

- **IBM Granite 3.2-2B-Instruct Model –** A fine-tuned causal language model used for text generation tasks.

Methodology:

The application workflow is divided into two modules:

1. Eco Tips Generator

Input: Users provide environmental keywords such as plastic pollution, renewable energy, water conservation, etc.

Process: The system constructs a prompt with the given keywords and feeds it to the IBM Granite model.

Output: The model generates a list of actionable and practical eco-friendly solutions tailored to the user's concern.

Example:

Input – "plastic waste"
Output – "Switch to reusable cloth bags, encourage local stores to adopt biodegradable packaging, and set up household segregation for plastics to enable recycling."

2. Policy Summarization

Input: Users can either upload a PDF file of a policy document or paste the text directly into the input box.

Process: The uploaded PDF is processed using PyPDF2, converting pages into raw text. A summarization prompt is then generated and passed to the language model.

Output: The AI produces a structured summary of the policy, highlighting the main objectives, key provisions, stakeholders involved, and expected implications.

Example:
A 50-page renewable energy policy can be summarized into a one-page report, emphasizing government targets, subsidies, and sustainability goals.

Features:

- **Dual Functionality:** Users can switch between Eco Tips Generator and Policy Summarization seamlessly using tabbed navigation.

- **PDF Upload Support:** Allows direct analysis of official policy documents without manual copy-pasting.

- **AI-Powered Summarization:** Produces concise and human-readable summaries, reducing hours of manual effort.

- **Interactive Interface:** Simple, clean, and responsive Gradio-based UI for ease of use.

- **Cross-Platform Compatibility:** Can be accessed via browser with options for local and public sharing.

Applications:

- **For Individuals:** Offers eco-friendly suggestions for sustainable lifestyles.

- **For Students & Researchers:** Helps in quickly analyzing policy frameworks for academic projects and case studies.

- **For NGOs & Activists:** Assists in simplifying complex policies to communicate with the general public.

- **For Policymakers & Analysts:** Provides quick summaries of lengthy government policies for decision-making support.

Advantages:

- Saves time by reducing lengthy policy reading into concise summaries.
- Encourages eco-conscious behavior with practical, localized tips.
- User-friendly interface makes it accessible to non-technical users.
- Utilizes open-source libraries and models, making it cost-effective.

Limitations:

- Model outputs depend heavily on prompt quality; vague inputs may lead to generic responses.
- Summaries might occasionally omit niche but important legal details.
- Requires internet and sufficient computational resources for smooth model execution.
- Limited handling of scanned PDFs with images (only text-based PDFs are supported).

Future Enhancements:

- Multilingual Support – Extend eco tips and summaries in multiple regional languages.

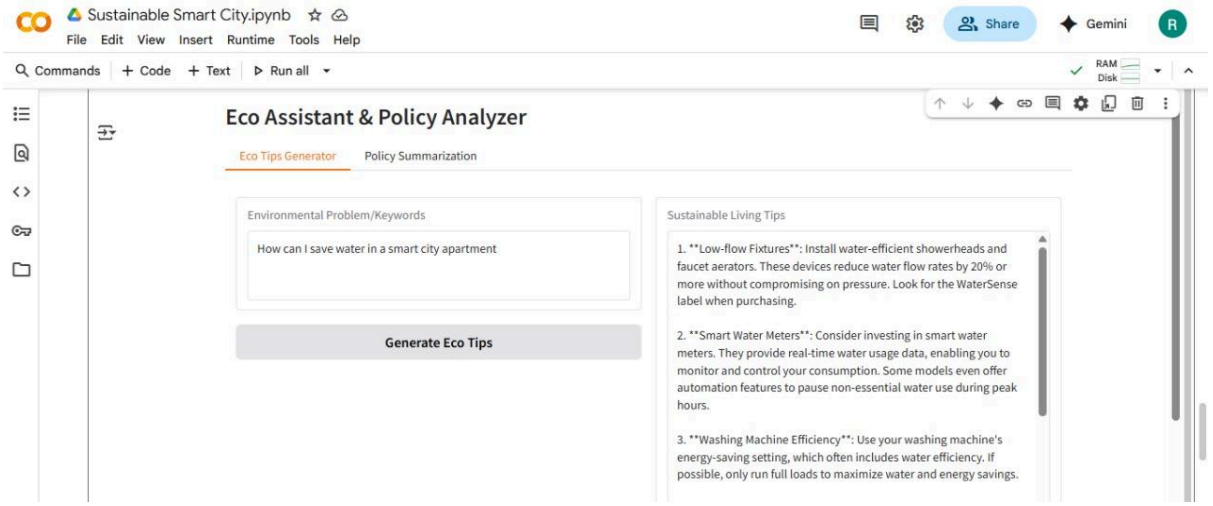
- Image-to-Text OCR Integration – Enable processing of scanned documents using OCR libraries.
- Recommendation Engine – Suggest eco-friendly products or organizations aligned with user's interest.
- Database Integration – Allow saving and retrieving previous summaries and eco tips.
- Mobile App Version – Build an Android/iOS application for broader accessibility.

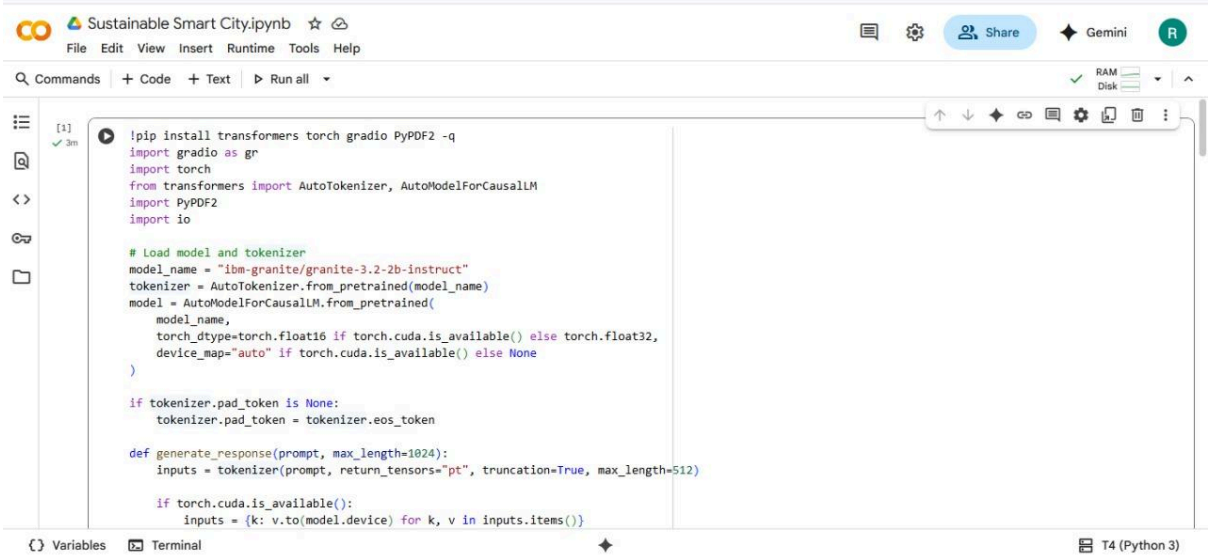
Conclusion:

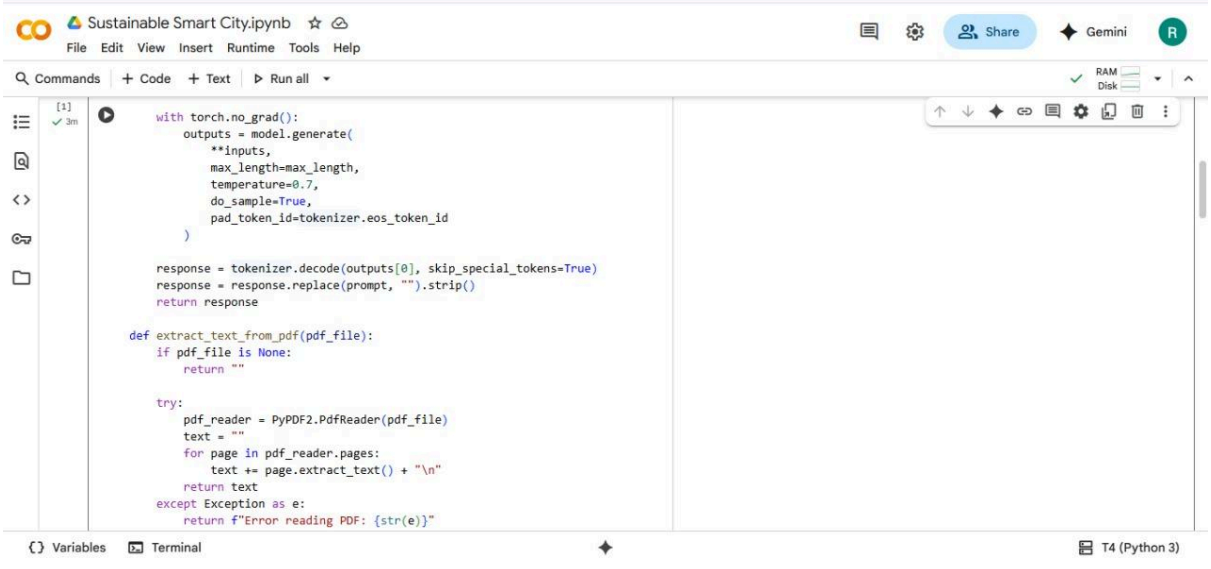
The Eco Assistant & Policy Analyzer demonstrates the potential of artificial intelligence in addressing two vital societal needs: promoting sustainability and making policies more understandable. By merging eco-friendly guidance with AI-powered policy summarization, the project provides a valuable tool for individuals, researchers, NGOs, and policymakers alike.

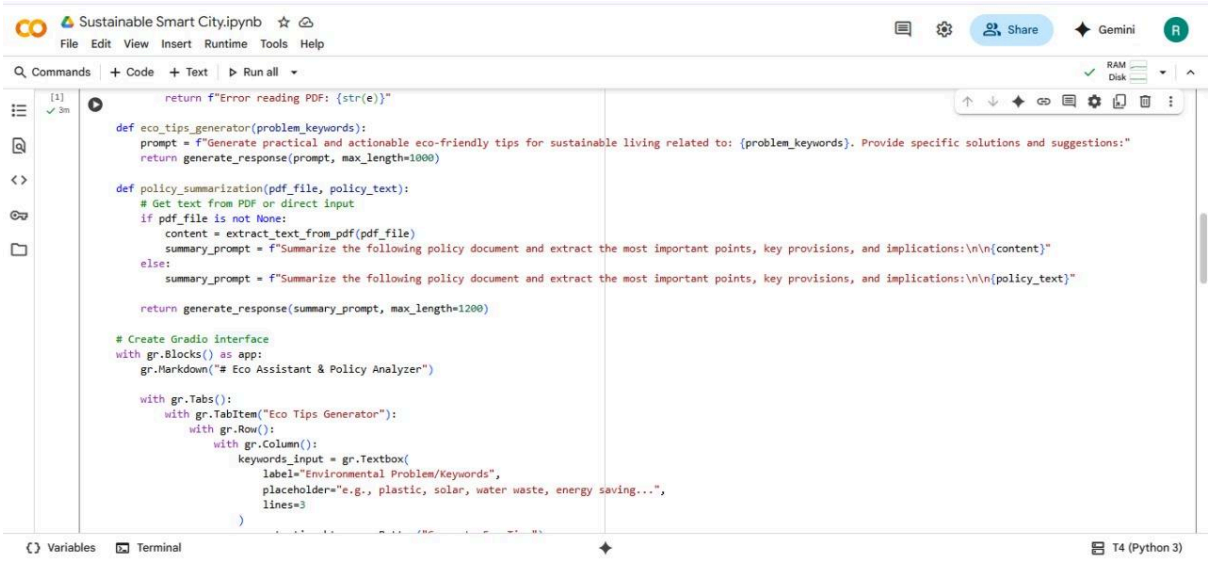
This project is a step toward creating technology that not only solves problems but also contributes to global sustainability goals. With future enhancements, it holds promise as a comprehensive assistant for both environmental awareness and policy understanding.

Screenshots:

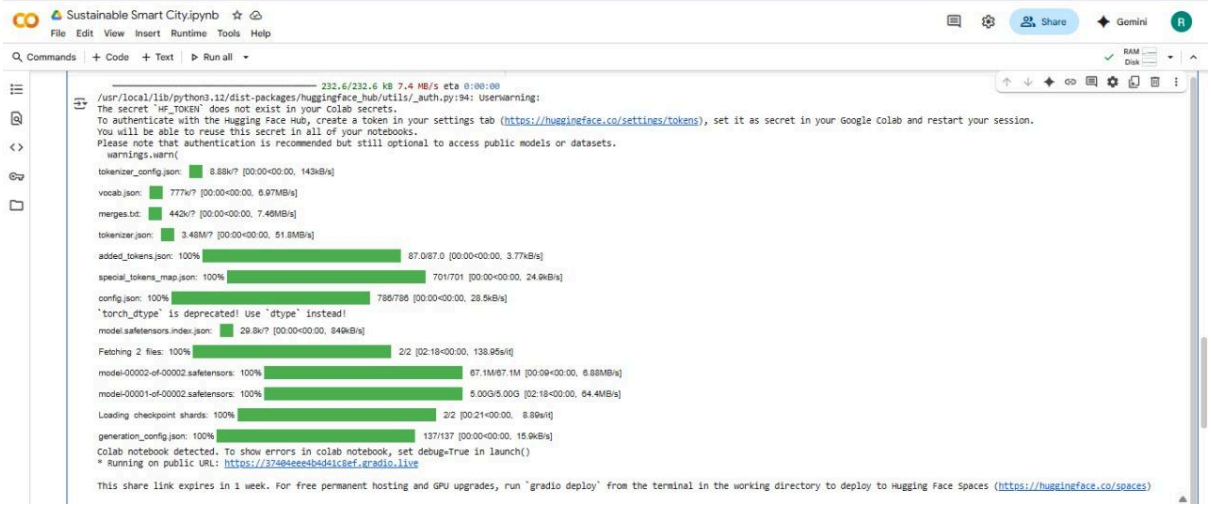





Sustainable Smart City.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[1]
3m
with torch.no_grad():
outputs = model.generate(
**inputs,
max_length=max_length,
temperature=0.7,
do_sample=True,
pad_token_id=tokenizer.eos_token_id
)
response = tokenizer.decode(outputs[0], 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)}"
Variables
Terminal
T4 (Python 3)


Sustainable Smart City.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[1]
3m
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 and suggestions:"
return generate_response(prompt, max_length=1000)
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 provisions, and implications:\n\n{content}"
else:
summary_prompt = f"Summarize the following policy document and extract the most important points, key provisions, 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 Analyzer")
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
)
Variables
Terminal
T4 (Python 3)


Sustainable Smart City.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[1]
3m
generate_tips_btn = gr.Button("Generate Eco Tips")
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.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():
summary_output = gr.Textbox(label="Policy Summary & Key Points", lines=20)
summarize_btn.click(policy_summarization, inputs=[pdf_upload, policy_text_input], outputs=summary_output)
app.launch(share=True)
Variables
Terminal
T4 (Python 3)


Sustainable Smart City.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
232.6/232.6 kB 7.4 MB/s eta 0:00:00
/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 session.
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, 143kB/s]
vocab.json: 777k/? [00:00<00:00, 6.97MB/s]
merges.txt: 442k/? [00:00<00:00, 7.46MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 51.8MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 3.77kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 24.9kB/s]
config.json: 100% 786/786 [00:00<00:00, 28.5kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 849kB/s]
Fetching 2 files: 100% 2/2 [02:18<00:00, 138.95s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:09<00:00, 6.88MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [02:18<00:00, 64.4MB/s]
Loading checkpoint shards: 100% 2/2 [00:21<00:00, 8.89s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 15.9kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://37404eee4b4d41c8ef.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 (https://huggingface.co/spaces)