Sustainable Smart City Assistant

- Project title : Sustainable Smart City Assistant

- Team member : Kevin K J NM ID: D1D9950377121098E51488473EF9D80F
- Team member : Jaylyn Jude NM ID: 6CECD7A31D0CA4DBA85407ED80603E82
- Team member : Nitesh Y NM ID: EC335324A48FBD78D674A475599354E2
- Team member : Santosh M NM ID: 8EC9F3CFD27ABC09D5043DB8E7950F11

1. Introduction

- The Sustainable Smart City Assistant is a project aimed at creating an intelligent, eco-friendly, and interactive platform to support both citizens and city officials in making informed decisions for sustainable living
- Unlike traditional static dashboards, this assistant leverages artificial intelligence and real-time data processing to provide meaningful insights and actionable recommendations
- The goal of the assistant is to serve as a bridge between technology, governance, and community engagement while fostering urban environments that are cleaner, greener, and more efficient
- It offers a conversational interface where users can interact naturally in plain language, lowering the barrier to accessing complex sustainability knowledge
- Furthermore, it brings together advanced features such as policy summarization, eco-friendly tip generation, and forecasting modules that make sustainability more practical and accessible
- The system not only encourages individuals to adopt eco-conscious behaviors but also equips city officials with decision-support tools that ensure efficient governance and planning
- With its scalable design and AI-driven backend, this assistant can be deployed across different cities and communities, adapting to the unique needs of each region while maintaining its focus on long-term environmental resilience.

2. Project Overview

- The Sustainable Smart City Assistant is built to address pressing urban challenges such as excessive energy consumption, water wastage, poor waste management, and limited public participation in environmental planning
- Its primary purpose is to empower cities and their residents with digital tools that simplify sustainability practices and policies
- Using artificial intelligence, the system can generate personalized eco-tips, summarize government policies into concise and actionable insights, and even detect anomalies in city resource usage

-

- One of its most distinctive features is the conversational interface, which allows both citizens and officials to ask questions and receive clear, plain-language responses without needing technical expertise
- Another major component is its integration with IBM's Granite large language model, which provides natural language understanding and high- quality generative capabilities
- The system is designed to forecast trends in energy and water consumption, generate key performance indicator (KPI) projections, and help stakeholders plan for future needs
- By providing this level of forecasting and anomaly detection, the project ensures proactive decision-making rather than reactive responses
- Together, these features create a holistic urban management tool that balances sustainability, technology, and citizen well-being in one unified platform.

3. Folder Structure

- app/ – Contains all backend logic including AI model integration, policy summarization, and eco-tip generation.

- app/api/ – Modular API routes (future extension if integrated with FastAPI).

- ui/ – Frontend components (Gradio interface with eco-tips and policy summarization tabs).

- eco_tips_generator.py – Generates AI-powered eco-friendly tips for sustainable living.

- policy_summarization.py – Summarizes uploaded policy PDFs or pasted text.

- pdf_text_extractor.py – Extracts raw text from PDF documents using PyPDF2.

- main.py – Entry point to launch the Gradio dashboard with multiple tabs.

- requirements.txt – Lists project dependencies (Transformers, Torch, Gradio, PyPDF2).

4.Features

- The assistant integrates a wide range of features designed for both individuals and city-level governance
- First, the conversational interface ensures that all users can communicate naturally and obtain relevant insights instantly

-

  Second, the policy summarization tool reduces the complexity of lengthy government documents by generating clear, concise summaries that highlight key provisions and implications
- Third, the eco-tip generator provides practical sustainability advice tailored to specific environmental concerns such as water conservation, waste reduction, or renewable energy usage
- In addition, the system supports resource forecasting to estimate energy, water, and waste usage using historical and real-time data, ensuring proactive management
- The platform also includes anomaly detection, which flags unusual patterns in usage or sensor data, serving as an early warning system for city officials
- Citizen engagement is strengthened through a feedback loop that collects public input, analyzes responses, and integrates this knowledge into future planning
- Multimodal input support allows the assistant to process and analyze multiple file formats such as PDFs and CSVs for broader usability
- Finally, the user interface, built with Gradio, is designed to be visually appealing, interactive, and user-friendly, ensuring accessibility for both citizens and administrators alike.

5.Architecture

- The architecture of the Sustainable Smart City Assistant is designed to ensure modularity, scalability, and real-time interaction
- The frontend of the application is built with Gradio, a lightweight but powerful library for building interactive machine learning interfaces
- Gradio provides a clean dashboard with tabs for eco-tip generation and policy summarization, both styled with modern CSS for accessibility and aesthetics
- The backend integrates IBM’s Granite large language model for natural language understanding and response generation
- These models are loaded using Hugging Face’s Transformers library and optimized to run on GPU when available
- PyPDF2 is used to extract text from uploaded PDF documents, allowing policy analysis and summarization
- For forecasting and anomaly detection, the design can be extended with machine learning modules such as scikit-learn or pandas-based processing
- This layered architecture ensures that the system can evolve with additional features such as resource forecasting, anomaly detection, and KPI tracking
- Furthermore, the backend is designed to support both local and cloud deployments, making the system flexible for research demonstrations, pilot city deployments, or enterprise- level scaling in urban planning projects.

6.Setup Instructions

- To run the Sustainable Smart City Assistant, users must first install the necessary dependencies, including Transformers, Torch, Gradio, and PyPDF2
- Python 3.9 or later is recommended for compatibility
- The process begins by cloning the repository and creating a virtual environment for package isolation
- Next, the dependencies listed in requirements.txt should be installed

-

- Once the environment is ready, the Gradio-based application can be launched directly
- For cloud-based or GPU-enabled systems, the backend will automatically detect and leverage CUDA resources to accelerate model inference
- Additional configuration steps involve ensuring that internet access is available for downloading model weights from Hugging Face
- Once launched, the application opens an interactive dashboard where users can input keywords for eco-tips or upload policy PDFs for summarization
  The assistant processes these inputs in real-time and provides actionable outputs
- Advanced users may also configure API keys for cloud-based IBM Granite deployments to extend capabilities
- Overall, the setup process is designed to be accessible and straightforward while supporting advanced customization for larger-scale deployments.

7.API Documentation

- The project exposes multiple functionalities through well-defined modules rather than REST APIs
- The eco-tip generator accepts environmental keywords and generates actionable sustainability advice using the LLM backend
- The policy summarization function allows users to either upload PDF files or paste raw policy text for AI-powered summarization
- The extract-text module uses PyPDF2 to retrieve textual content from PDFs, which is then fed into the LLM for summarization
- Each of these modules is connected to Gradio buttons and text fields to ensure seamless interaction
- In an extended version, the project could be integrated with FastAPI to expose endpoints such as /chat/ask for conversational queries, /upload-doc for policy embedding, /get-eco-tips for sustainability advice, and /submit-feedback for community engagement
- Such extensions would enable multi-user deployments where citizens, researchers, and city officials can all interact with the assistant via authenticated APIs
- Currently, the system focuses on local interactivity but is architected in such a way that scaling into a full API-driven service is straightforward.

8.User Interface

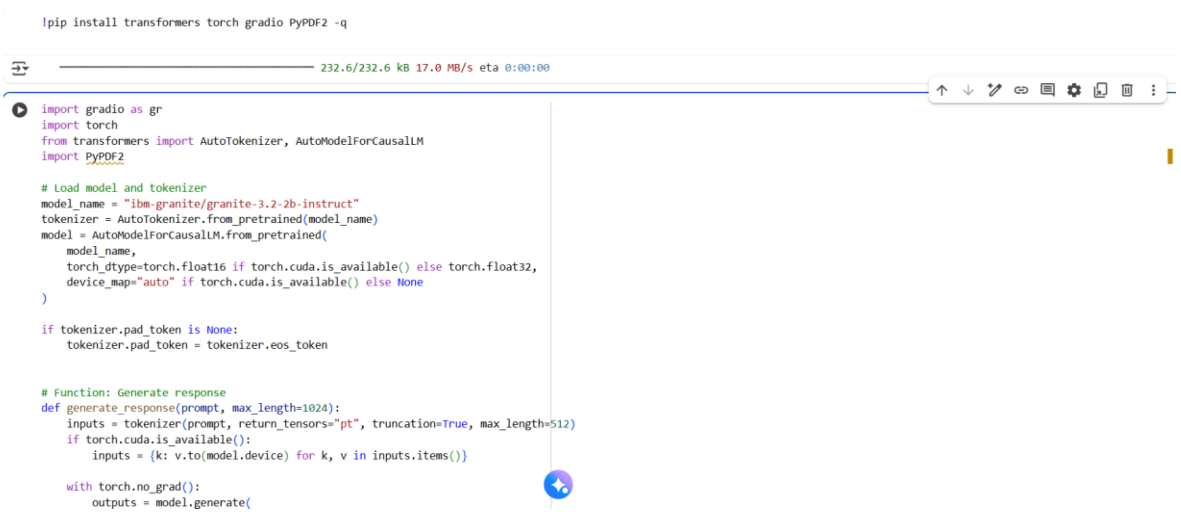
- The user interface of the Sustainable Smart City Assistant is built using Gradio, which provides an accessible and lightweight framework for interactive dashboards
- The design includes two primary tabs: the Eco Tips Generator and the Policy Summarization module
- Each tab is styled with custom CSS to create a visually appealing layout featuring modern gradients, rounded cards, and interactive textboxes
- Users can easily type in environmental issues such as “plastic waste” or “energy saving” to instantly receive detailed, AI-generated tips
- The policy summarization tab allows users to upload PDF documents or paste policy text, which is then processed into a structured summary with key points

-

- Buttons and interactive fields are styled for responsiveness, and hover effects ensure a smooth experience
- The design focuses on accessibility for non-technical users, ensuring that both citizens and officials can make use of the assistant with minimal training
- A footer is also included to highlight the project's mission of building a greener future powered by AI
- This minimalist yet engaging interface helps users focus on the content rather than navigating complex menus or tools.

9.Testing

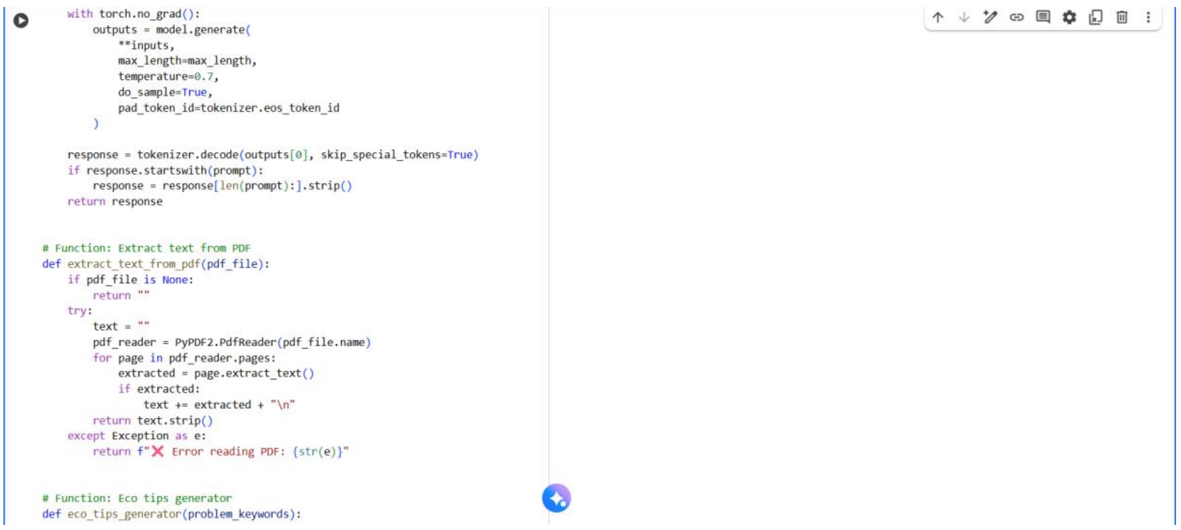
- The system underwent multiple layers of testing to validate its reliability, usability, and accuracy
- Unit testing was conducted for core modules such as eco-tip generation, PDF text extraction, and policy summarization to ensure that outputs aligned with expected results
  Manual testing involved interacting with the Gradio interface to verify responsiveness, layout integrity, and correctness of generated outputs
- Edge case testing was also performed with empty inputs, very large PDFs, and malformed text to ensure the assistant could handle unexpected inputs gracefully
- Performance testing measured response times on CPU versus GPU deployments, confirming that CUDA acceleration improves inference speed significantly
- Additionally, user testing was conducted with small groups to validate the clarity and usefulness of eco-tips and summaries
- Feedback from these tests was incorporated to improve prompt design and user interface styling
- Future testing phases will involve stress-testing the application in multi-user environments and integrating automated test suites for continuous validation
- Together, these tests ensure that the assistant is robust, reliable, and ready for broader adoption.

- 

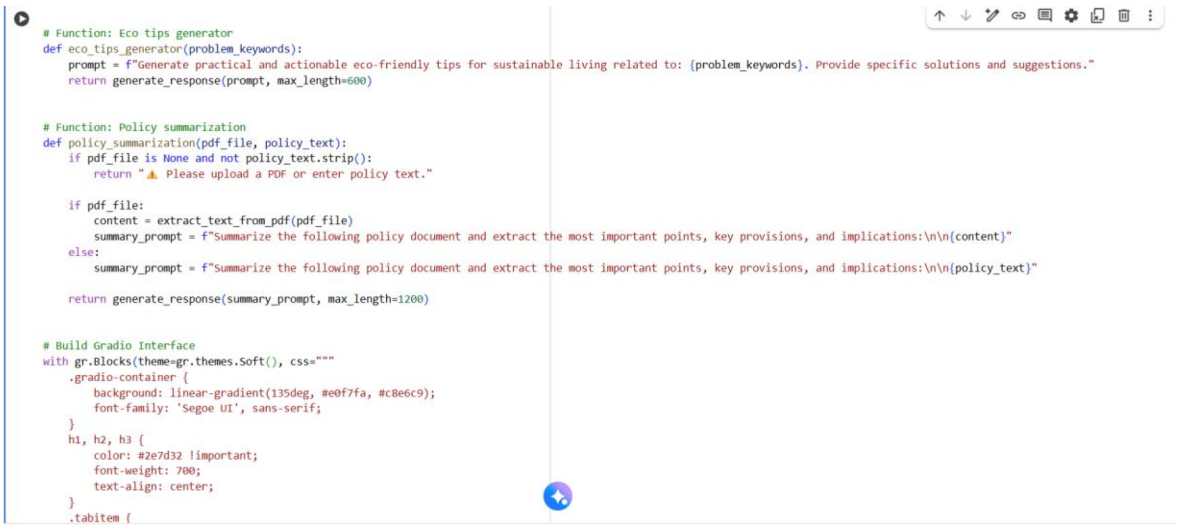
```
!pip install transformers torch gradio PyPDF2 -q
```

```
───────────────────────────────── 232.6/232.6 kB 17.0 MB/s eta 0:00:00
```

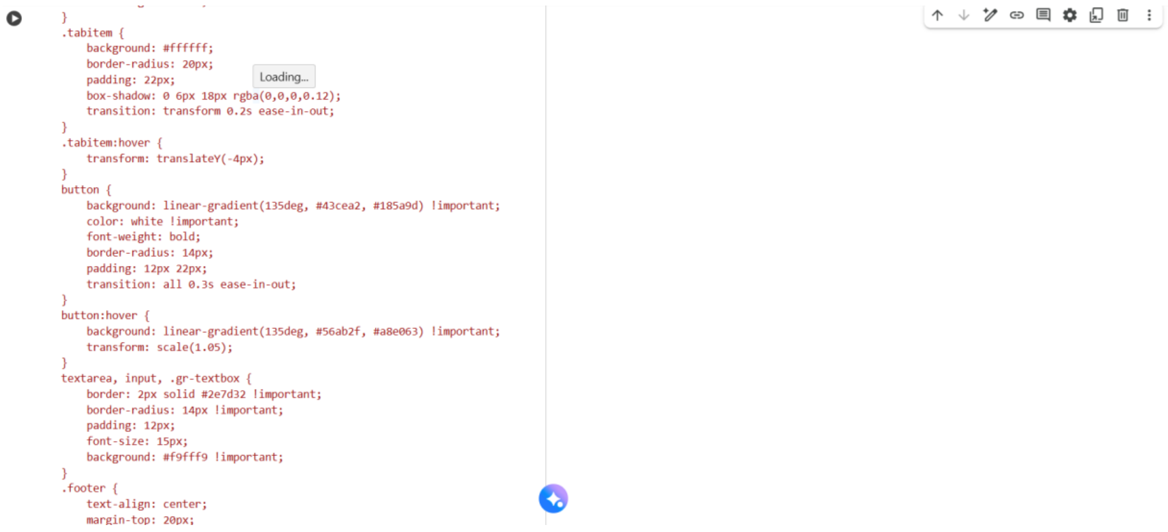
```python
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
import PyPDF2

# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token


# Function: 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(
```



```python
    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)
    if response.startswith(prompt):
        response = response[len(prompt):].strip()
    return response


# Function: Extract text from PDF
def extract_text_from_pdf(pdf_file):
    if pdf_file is None:
        return ""
    try:
        text = ""
        pdf_reader = PyPDF2.PdfReader(pdf_file.name)
        for page in pdf_reader.pages:
            extracted = page.extract_text()
            if extracted:
                text += extracted + "\n"
        return text.strip()
    except Exception as e:
        return f"❌ Error reading PDF: {str(e)}"


# Function: Eco tips generator
def eco_tips_generator(problem_keywords):
```

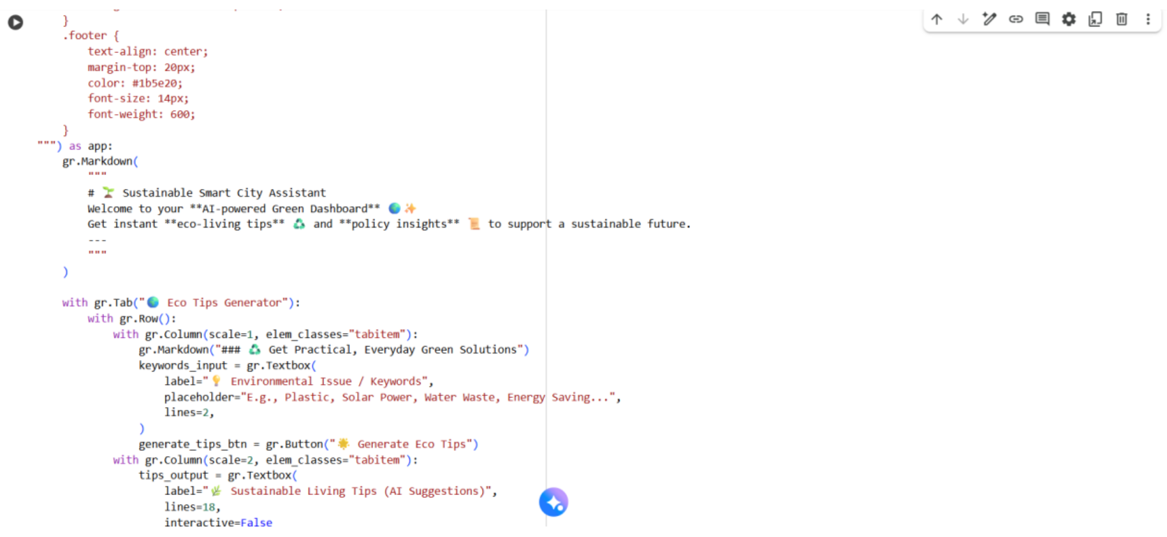


```python
# Function: Eco tips generator
def eco_tips_generator(problem_keywords):
    prompt = f"Generate practical and actionable eco-friendly tips for sustainable living related to: {problem_keywords}. Provide specific solutions and suggestions."
    return generate_response(prompt, max_length=600)


# Function: Policy summarization
def policy_summarization(pdf_file, policy_text):
    if pdf_file is None and not policy_text.strip():
        return "⚠️ Please upload a PDF or enter policy text."

    if pdf_file:
        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)


# Build Gradio Interface
with gr.Blocks(theme=gr.themes.Soft(), css="""
    .gradio-container {
        background: linear-gradient(135deg, #e0f7fa, #c8e6c9);
        font-family: 'Segoe UI', sans-serif;
    }
    h1, h2, h3 {
        color: #2e7d32 !important;
        font-weight: 700;
        text-align: center;
    }
    .tabitem {
```



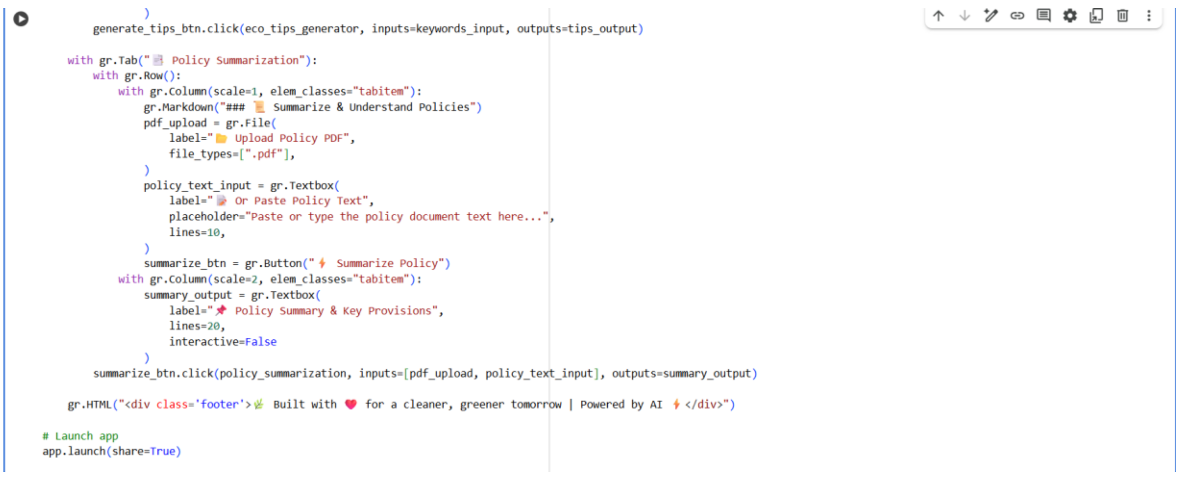
```css
    }
    .tabitem {
        background: #ffffff;
        border-radius: 20px;
        padding: 22px;
        box-shadow: 0 6px 18px rgba(0,0,0,0.12);
        transition: transform 0.2s ease-in-out;
    }
    .tabitem:hover {
        transform: translateY(-4px);
    }
    button {
        background: linear-gradient(135deg, #43cea2, #185a9d) !important;
        color: white !important;
        font-weight: bold;
        border-radius: 14px;
        padding: 12px 22px;
        transition: all 0.3s ease-in-out;
    }
    button:hover {
        background: linear-gradient(135deg, #56ab2f, #a8e063) !important;
        transform: scale(1.05);
    }
    textarea, input, .gr-textbox {
        border: 2px solid #2e7d32 !important;
        border-radius: 14px !important;
        padding: 12px;
        font-size: 15px;
        background: #f9fff9 !important;
    }
    .footer {
        text-align: center;
        margin-top: 20px;
```

•



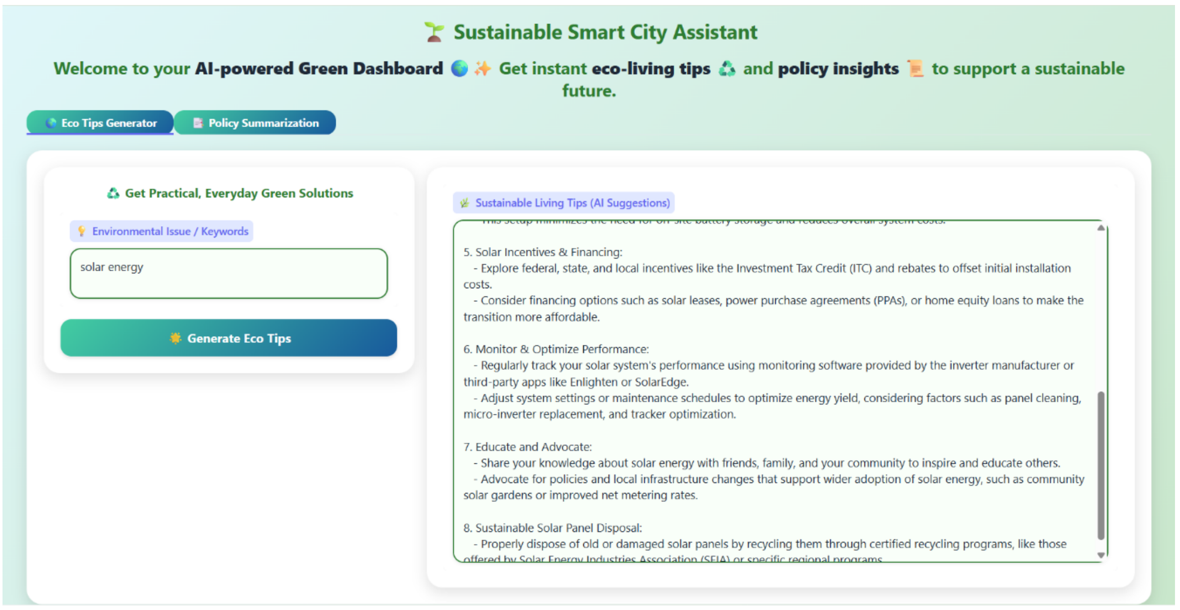
```
    }
    .footer {
        text-align: center;
        margin-top: 20px;
        color: #1b5e20;
        font-size: 14px;
        font-weight: 600;
    }
""") as app:
    gr.Markdown(
        """
        # 🌱 Sustainable Smart City Assistant
        Welcome to your **AI-powered Green Dashboard** 🌍✨
        Get instant **eco-living tips** ♻️ and **policy insights** 📜 to support a sustainable future.
        ---
        """
    )

    with gr.Tab("🌍 Eco Tips Generator"):
        with gr.Row():
            with gr.Column(scale=1, elem_classes="tabitem"):
                gr.Markdown("### ♻️ Get Practical, Everyday Green Solutions")
                keywords_input = gr.Textbox(
                    label="💡 Environmental Issue / Keywords",
                    placeholder="E.g., Plastic, Solar Power, Water Waste, Energy Saving...",
                    lines=2,
                )
                generate_tips_btn = gr.Button("🌟 Generate Eco Tips")
            with gr.Column(scale=2, elem_classes="tabitem"):
                tips_output = gr.Textbox(
                    label="🌿 Sustainable Living Tips (AI Suggestions)",
                    lines=18,
                    interactive=False
```

•



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

    with gr.Tab("📄 Policy Summarization"):
        with gr.Row():
            with gr.Column(scale=1, elem_classes="tabitem"):
                gr.Markdown("### 📜 Summarize & Understand Policies")
                pdf_upload = gr.File(
                    label="📁 Upload Policy PDF",
                    file_types=[".pdf"],
                )
                policy_text_input = gr.Textbox(
                    label="📝 Or Paste Policy Text",
                    placeholder="Paste or type the policy document text here...",
                    lines=10,
                )
                summarize_btn = gr.Button("⚡ Summarize Policy")
            with gr.Column(scale=2, elem_classes="tabitem"):
                summary_output = gr.Textbox(
                    label="📌 Policy Summary & Key Provisions",
                    lines=20,
                    interactive=False
                )
        summarize_btn.click(policy_summarization, inputs=[pdf_upload, policy_text_input], outputs=summary_output)

    gr.HTML("<div class='footer'>🌿 Built with ❤️ for a cleaner, greener tomorrow | Powered by AI ⚡</div>")

# Launch app
app.launch(share=True)
```

•



🌱 **Sustainable Smart City Assistant**

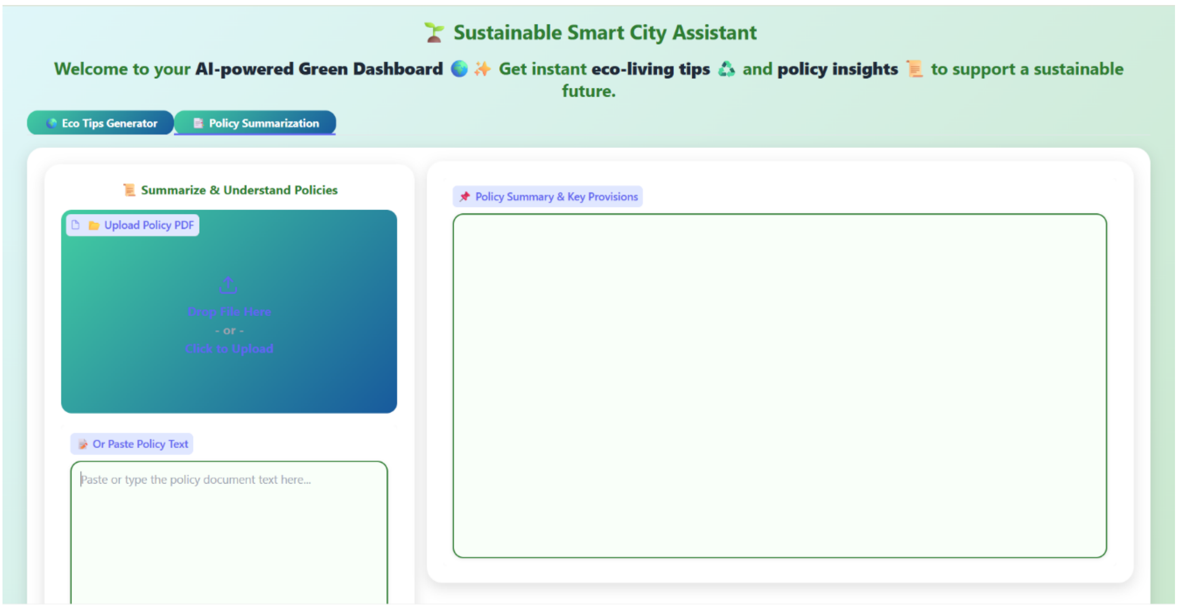
Welcome to your **AI-powered Green Dashboard** 🌍✨ Get instant **eco-living tips** ♻️ and **policy insights** 📜 to support a sustainable future.

🌍 Eco Tips Generator
📄 Policy Summarization
♻️ Get Practical, Everyday Green Solutions
💡 Environmental Issue / Keywords
solar energy
🌟 Generate Eco Tips
🌿 Sustainable Living Tips (AI Suggestions)

```
   This setup minimizes the need for on-site battery storage and reduces overall system costs.

5. Solar Incentives & Financing:
   - Explore federal, state, and local incentives like the Investment Tax Credit (ITC) and rebates to offset initial installation costs.
   - Consider financing options such as solar leases, power purchase agreements (PPAs), or home equity loans to make the transition more affordable.

6. Monitor & Optimize Performance:
   - Regularly track your solar system's performance using monitoring software provided by the inverter manufacturer or third-party apps like Enlighten or SolarEdge.
   - Adjust system settings or maintenance schedules to optimize energy yield, considering factors such as panel cleaning, micro-inverter replacement, and tracker optimization.

7. Educate and Advocate:
   - Share your knowledge about solar energy with friends, family, and your community to inspire and educate others.
   - Advocate for policies and local infrastructure changes that support wider adoption of solar energy, such as community solar gardens or improved net metering rates.

8. Sustainable Solar Panel Disposal:
   - Properly dispose of old or damaged solar panels by recycling them through certified recycling programs, like those offered by Solar Energy Industries Association (SEIA) or specific regional programs.
```

•

•

• 

🌱 Sustainable Smart City Assistant

Welcome to your **AI-powered Green Dashboard** 🌍 ✨ Get instant **eco-living tips** ♻️ and **policy insights** 📜 to support a sustainable future.

10.Future Enhancements

- While the current version of the Sustainable Smart City Assistant provides powerful features such as eco-tip generation and policy summarization, several future enhancements are envisioned
- First, deeper integration with machine learning forecasting models will allow accurate projections of city-level KPIs such as energy consumption, waste trends, and water usage
- Second, anomaly detection modules will be fully implemented to provide early warnings on irregular resource usage or infrastructure issues
- Third, citizen engagement will be expanded through a structured feedback system that integrates public opinion directly into policy recommendations
- Multi-language support will also be added to make the assistant accessible to diverse populations
- Secure deployments will include authentication mechanisms such as API keys, OAuth2, and role-based access control to support enterprise and city- wide rollouts
- Finally, the system could be integrated with advanced visualization dashboards built in Streamlit or other frameworks to provide rich analytics and reporting
- These enhancements will expand the assistant from a demonstration tool into a comprehensive, city-scale decision support system for sustainability.