SUSTAINABLE SMART CITY ASSISTANT USING IBM GRANITE LLM

Project Documentation done by : Madhumita G

1.Introduction:

•Sustainable Smart City

•Team Leader : MANOGARI V

•Team member : MADHUMITHA G

•Team member : MAHALAKSHIMI K

•Team member : Malar Ranjani M

2.Project overview:

* Core Vision

To create an AI-powered digital assistant that serves as a single point of contact for citizens and city officials to access information, automate tasks, and make data-driven decisions that promote urban sustainability and improve quality of life.

* Key Objectives

· Empower Citizens: Make sustainable living easier and more accessible.

· Optimize Operations: Help city government manage resources more efficiently.

· Improve Decision-Making: Provide data-driven insights for urban planning.

· Increase Engagement: Foster a collaborative relationship between citizens and their city.

* Target Users & Functionality

A. For Citizens (Public Chatbot & Mobile App):

· Waste Management Guide ("Waste Wizard"):

· Answers questions on recycling, compost, and trash rules via chat (e.g., "Can I recycle this plastic wrapper?").

· Sends personalized collection day reminders and alerts for schedule changes.

· Sustainable Mobility Planner:

· Provides integrated, multi-modal travel routes (public transit, bike-share, walking).

· Locates EV charging stations and provides real-time availability and pricing.

· Calculates carbon footprint savings for chosen routes.

· Resource Conservation Helper:

· Analyzes anonymized utility (water, energy) usage to provide personalized conservation tips.

· Connects users to rebate programs for energy-efficient appliances.

· Civic Engagement Portal:

· Reports issues like potholes, broken streetlights, or illegal dumping via chat and image upload.

· Informs users about local community events, farmers' markets, and public meetings.

3.Architecture:

Core Concept: A secure, AI-powered assistant that uses city data to promote sustainability via a conversational interface.

*User Layer:

· Interfaces: Public Web Chat, Mobile App, City Official Dashboard.

*Orchestration Layer:

· Backend Server: Manages user requests, security, and conversation state.

· Key Task: Constructs intelligent prompts for the LLM.

*AI Core (IBM watsonx.ai Platform):

· IBM Granite LLM: The reasoning engine. Its strengths are:

· Code Generation: Excels at translating user requests into API calls and data queries.
· Enterprise Security: Deployed securely on IBM Cloud, ensuring data privacy and compliance.

*Action & Data Layer:
· Action Broker: Executes the API calls decided by the LLM (e.g., fetch transit data, check recycling rules).
· Data Ecosystem: Connects to city APIs (Transport, Waste, Energy IoT sensors) and external services (Maps).
· Vector Database (For Accuracy): Stores official city documents. Used to retrieve facts and ground the LLM's responses, preventing hallucinations.
*How It Works:
• A user asks a question (e.g., "How do I recycle electronics?").
•The backend sends the query + context to Granite.
•Granite decides if it can answer or needs data.
•The Action Broker calls the required API (e.g., waste management database).
•Granite synthesizes the data into a clear, natural language answer.
•The response is delivered to the user.

4.Setup Instruction:
PREREQUISITES
✓Governance & Planning
✓ Technical Prerequisites
INSTALLATION PROCESS
✓Set Up the IBM watsonx.ai Environment
✓Backend Application Setup
✓Data Layer Configuration
✓Deployment
✓Frontend Integration
✓Testing & Validation

5. Folder Structure:
•app.py - Main application file that:
•Initializes the Gradio interface
•Sets up the model and tokenizer
• Defines the application workflow and UI components
*requirements.txt - Ensures consistent environment setup by specifying exact package versions needed
* README.md - Documentation that explains:
.How to install and run the application
· What the application does
· How to use both features (Eco Tips and Policy Analysis)
* models/ - Optional directory to cache the pretrained model locally rather than downloading each time
*utils/ - Modularizes functionality for better code organization:
· pdf_processor.py handles all PDF-related operations
· model_handler.py manages model loading and text generation
*static/ - Contains assets that enhance the UI/UX:

· Custom CSS to style the Gradio interface
· Images for branding and visual appeal

* templates/ - For future expansion if converting to a web framework like Flask/FastAPI

* tests/ - Ensures code reliability through automated testing:
· Verifies PDF text extraction works correctly
· Tests that model generates appropriate responses.

6. Running the Application

1. Python (3.8 or higher): The most common language for these projects.
· Download from python.org.
· Verify installation: python --version or python3 --version
2. Pip (Python Package Manager): Usually comes with Python.
· Verify: pip --version or pip3 --version
3. IBM Cloud Account & API Key:
· Go to IBM Cloud and create a free account.
· Create an API key for yourself (Search for "IBM Cloud API keys" in the console).
· You need the Project ID for your Watsonx.ai service.
· Go to your IBM Cloud Resource List, find your Watsonx.ai service, and copy its GUID (a long unique string). This is often used as the project_id.

4. The Application Code:
· This is likely in a GitHub repository. You need to clone or download the code to your computer.
· Example: git clone <repository-url>

7.API Documentation

- POST /api/chat-To ask questions and get informative, context-aware answer about sustainable urban living.
- POST /api/analyze/policy-To simulate and get a summary of the potential economic, environmental, and social impacts of a proposed city policy.
- POST /api/generate/report-To automatically generate reports (e.g., Annual Sustainability Report, Carbon Footprint Analysis) from structured data.
- POST /api/analyze/sentiment-To process large volumes of text feedback and summarize the main complaints, suggestions, and public sentiment.
- POST /api/optimize/:resource (e.g., /api/optimize/energy)-To get specific, actionable recommendations for optimizing a particular city resource.

8.Authentication

1. Purpose: Secure Role-Based Access
2. Primary Method: JWT (JSON Web Tokens)
3. Key API Endpoints
4. Role-Based Access Control (RBAC)
5. Environment Configuration

6. Integration with IBM Granite

9.User Interface

The UI transforms the complex AI and data capabilities into a simple, actionable, and engaging experience for everyone in the city.

10.Testing

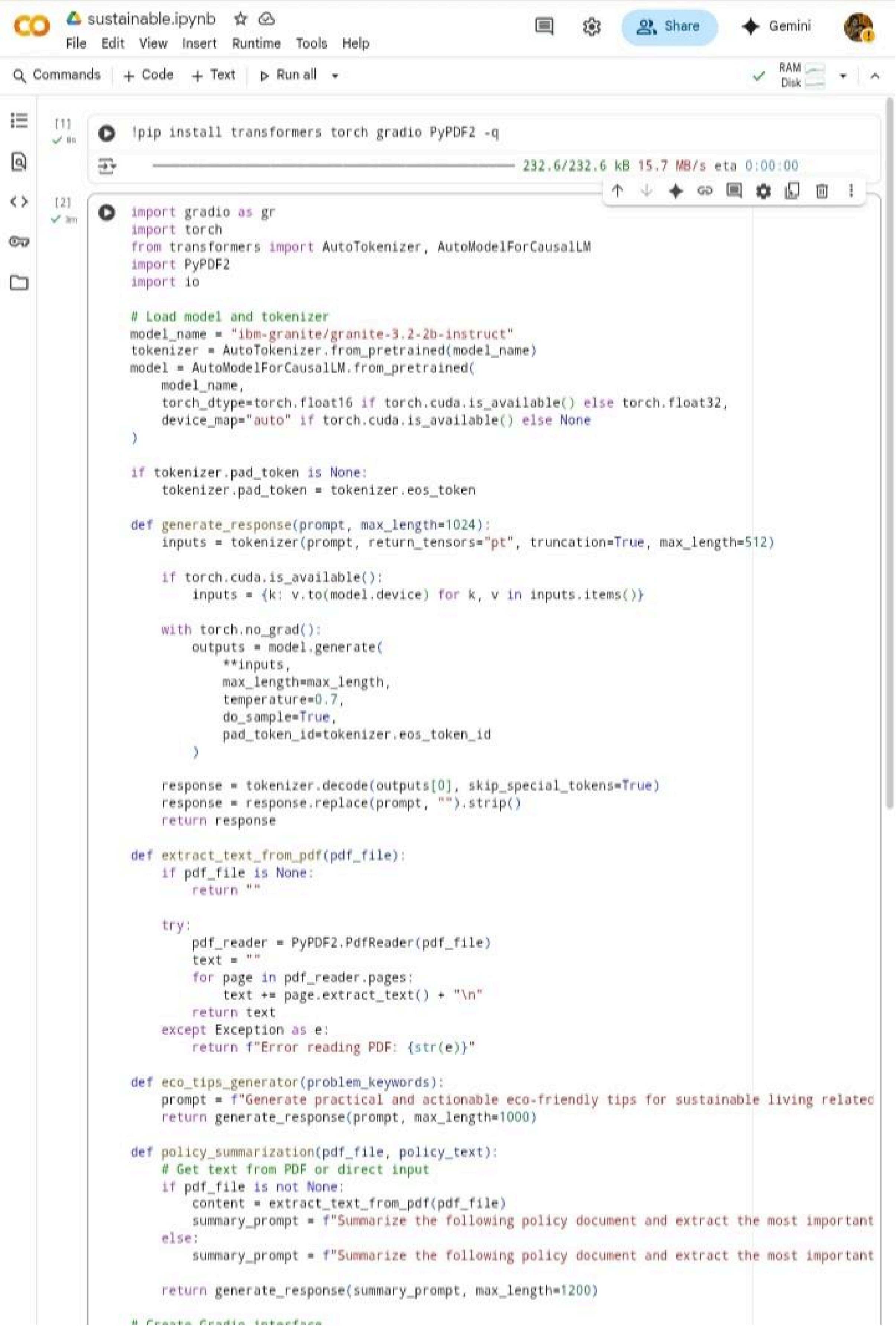
Testing was done in different phases:

Phase 1: Requirements Analysis
Phase 2: Data Collection & Validation
Phase 3: System Integration Testing
Phase 4: Functional Testing
Phase 5: Performance Testing
Phase 6: Security Testing
Phase 7: User Acceptance Testing (UAT)
Phase 8: Pilot Deployment Testing
Phase 9: Sustainability Impact Assessment
Phase 10: Regression & Maintenance testing
Phase 11: Compliance Testing
Phase 12: Disaster Recovery Testing

Screenshot:

Program:

first cell:

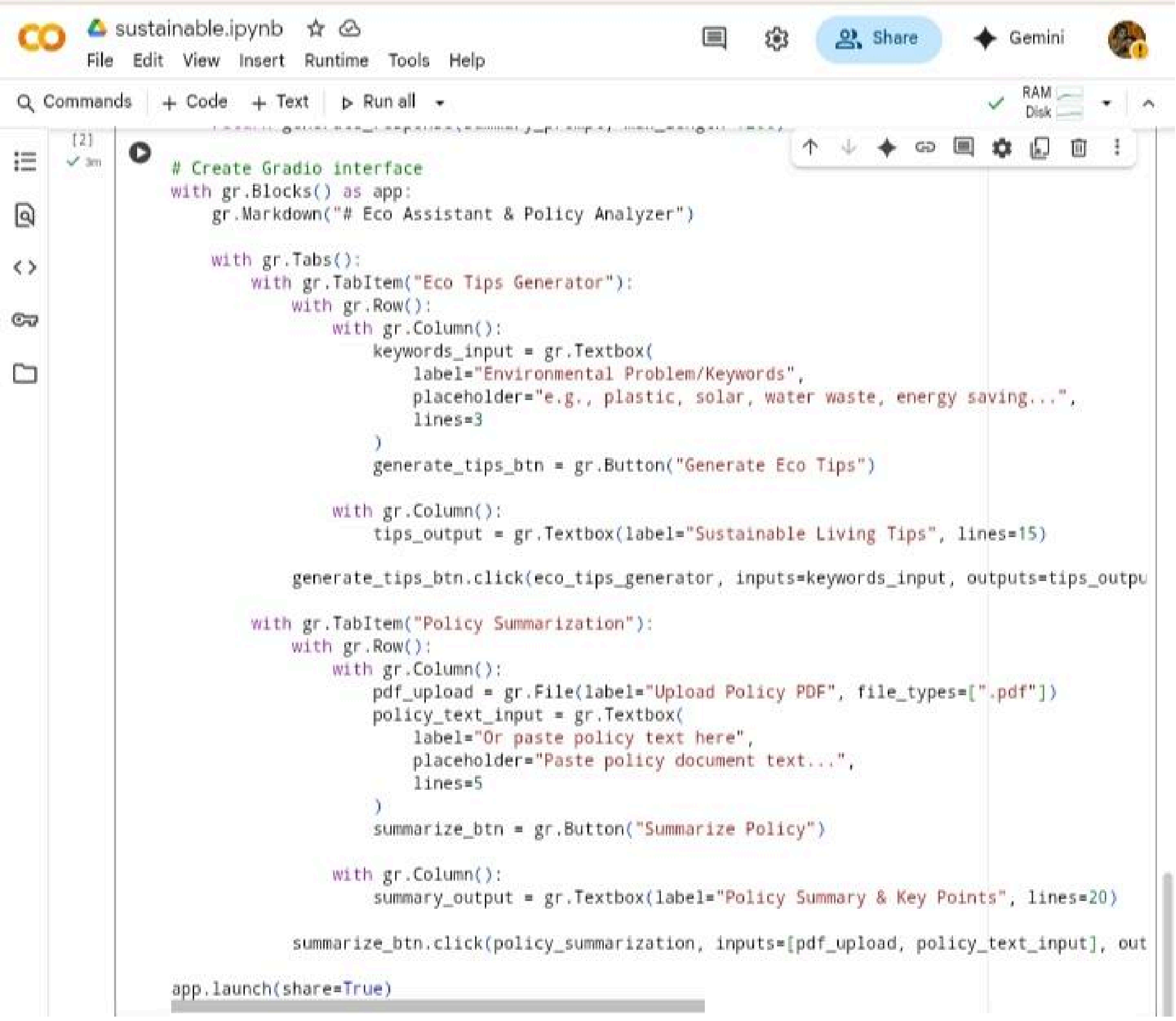


```
!pip install transformers torch gradio PyPDF2 -q
```

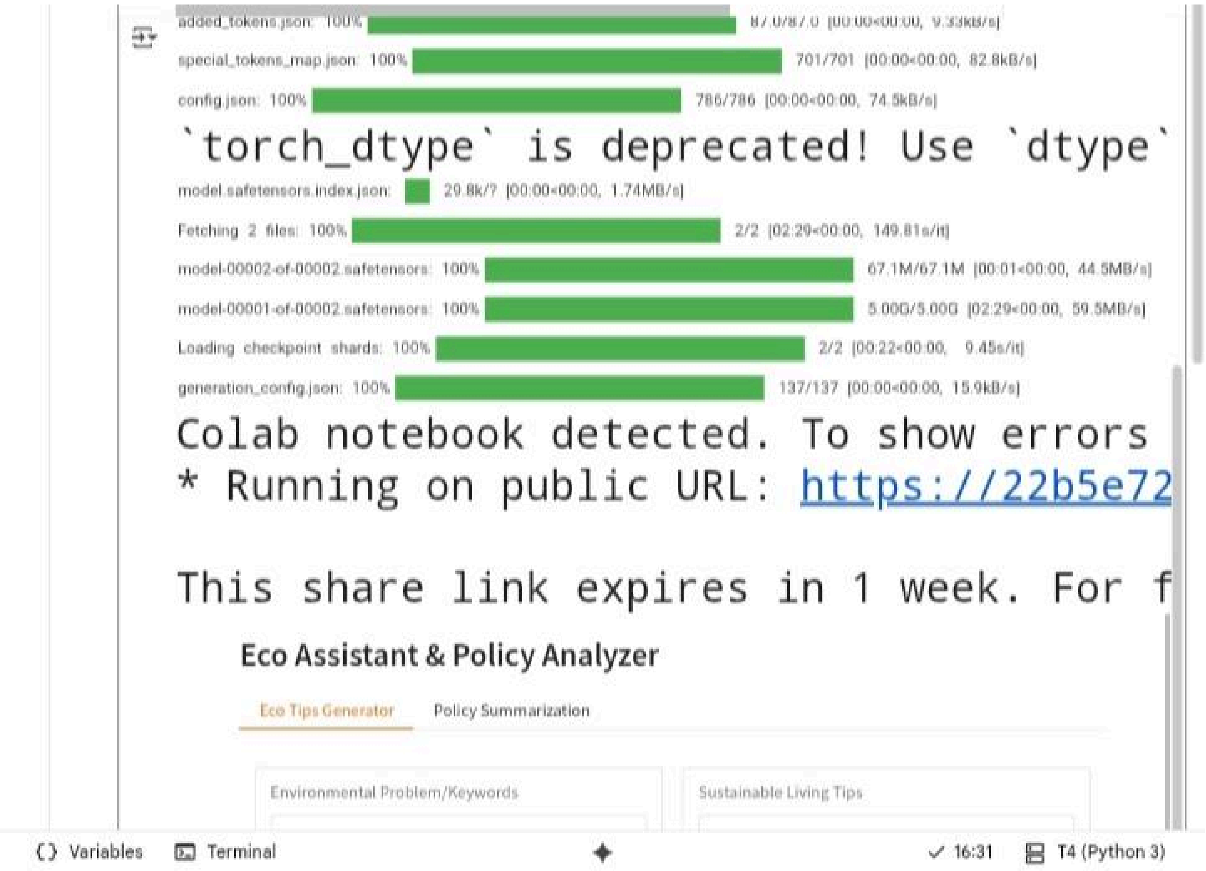
```
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 15.7 MB/s eta 0:00:00
```

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

# 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

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

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 relatec
    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
    else:
        summary_prompt = f"Summarize the following policy document and extract the most important

    return generate_response(summary_prompt, max_length=1200)

# Create Gradio interface
```

Second cell:

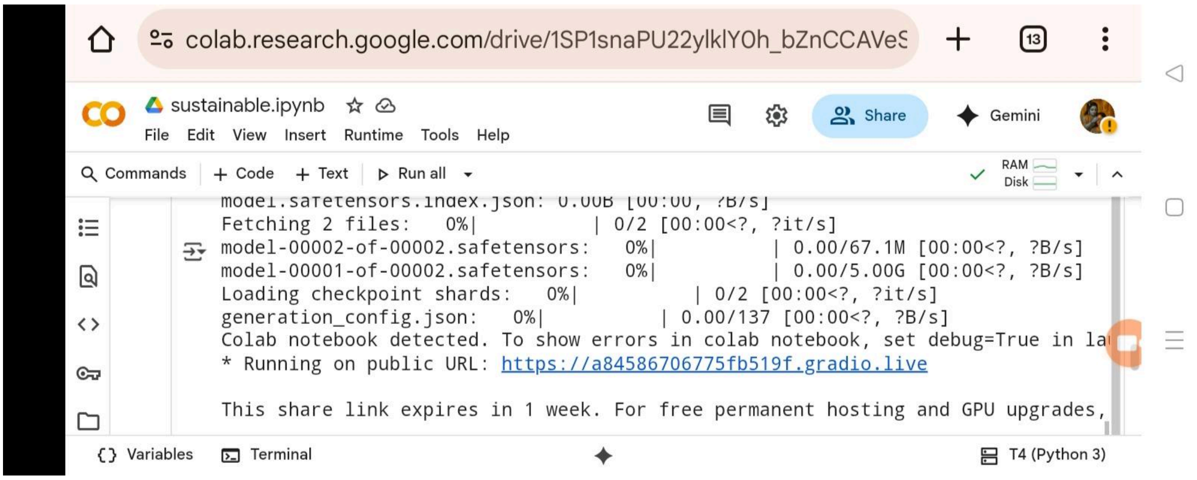


```python
# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# Eco Assistant & Policy Analyzer")

    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
                    )
                    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_outpu

        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], out

app.launch(share=True)
```



```
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 9.33kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 82.8kB/s]
config.json: 100% 786/786 [00:00<00:00, 74.5kB/s]
`torch_dtype` is deprecated! Use `dtype`
model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.74MB/s]
Fetching 2 files: 100% 2/2 [02:29<00:00, 149.81s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 44.5MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [02:29<00:00, 59.5MB/s]
Loading checkpoint shards: 100% 2/2 [00:22<00:00, 9.45s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 15.9kB/s]
Colab notebook detected. To show errors
* Running on public URL: https://22b5e72

This share link expires in 1 week. For f
```

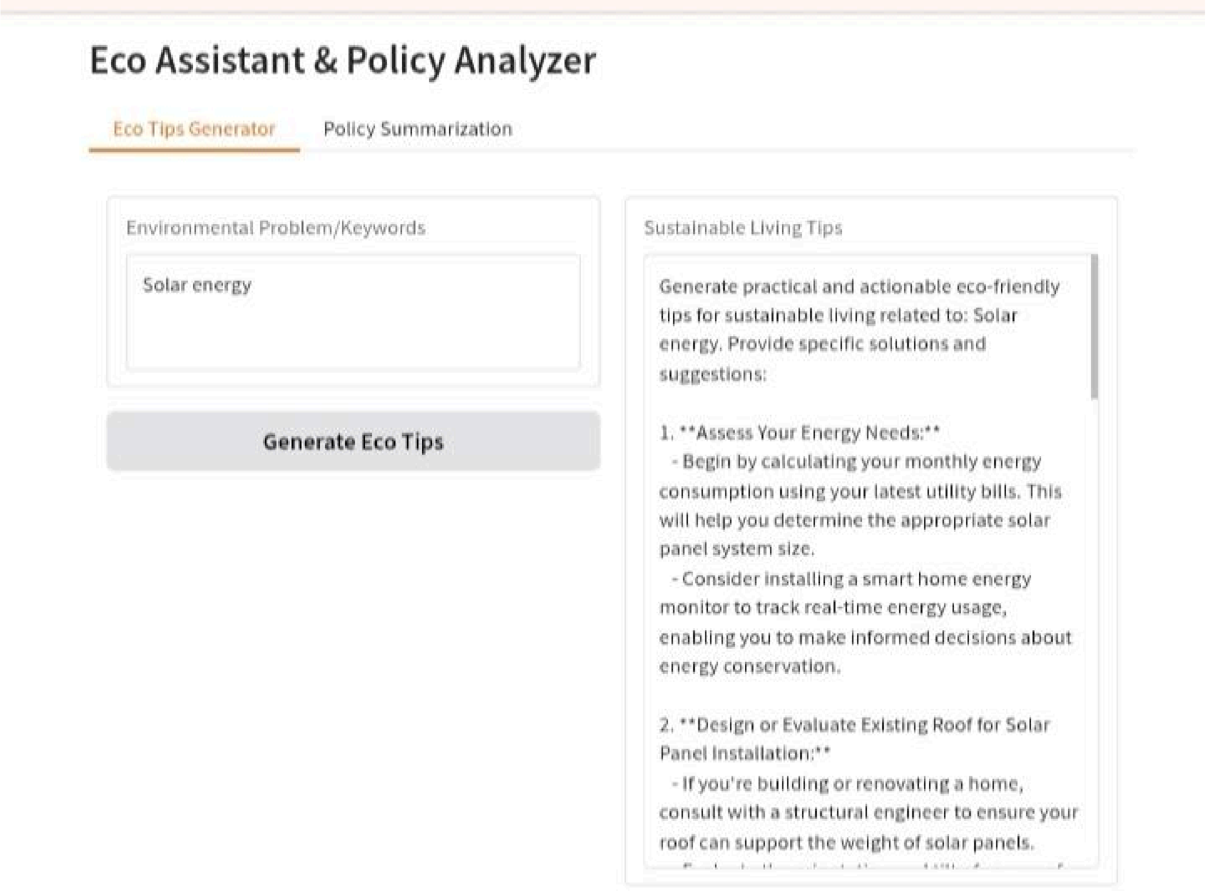
Eco Assistant & Policy Analyzer



```
model.safetensors.index.json: 0.00B [00:00, ?B/s]
Fetching 2 files:   0%|          | 0/2 [00:00<?, ?it/s]
model-00002-of-00002.safetensors:   0%|          | 0.00/67.1M [00:00<?, ?B/s]
model-00001-of-00002.safetensors:   0%|          | 0.00/5.00G [00:00<?, ?B/s]
Loading checkpoint shards:   0%|          | 0/2 [00:00<?, ?it/s]
generation_config.json:   0%|          | 0.00/137 [00:00<?, ?B/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in la
* Running on public URL: https://a84586706775fb519f.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades,
```

Output:

- Eco tips generator:



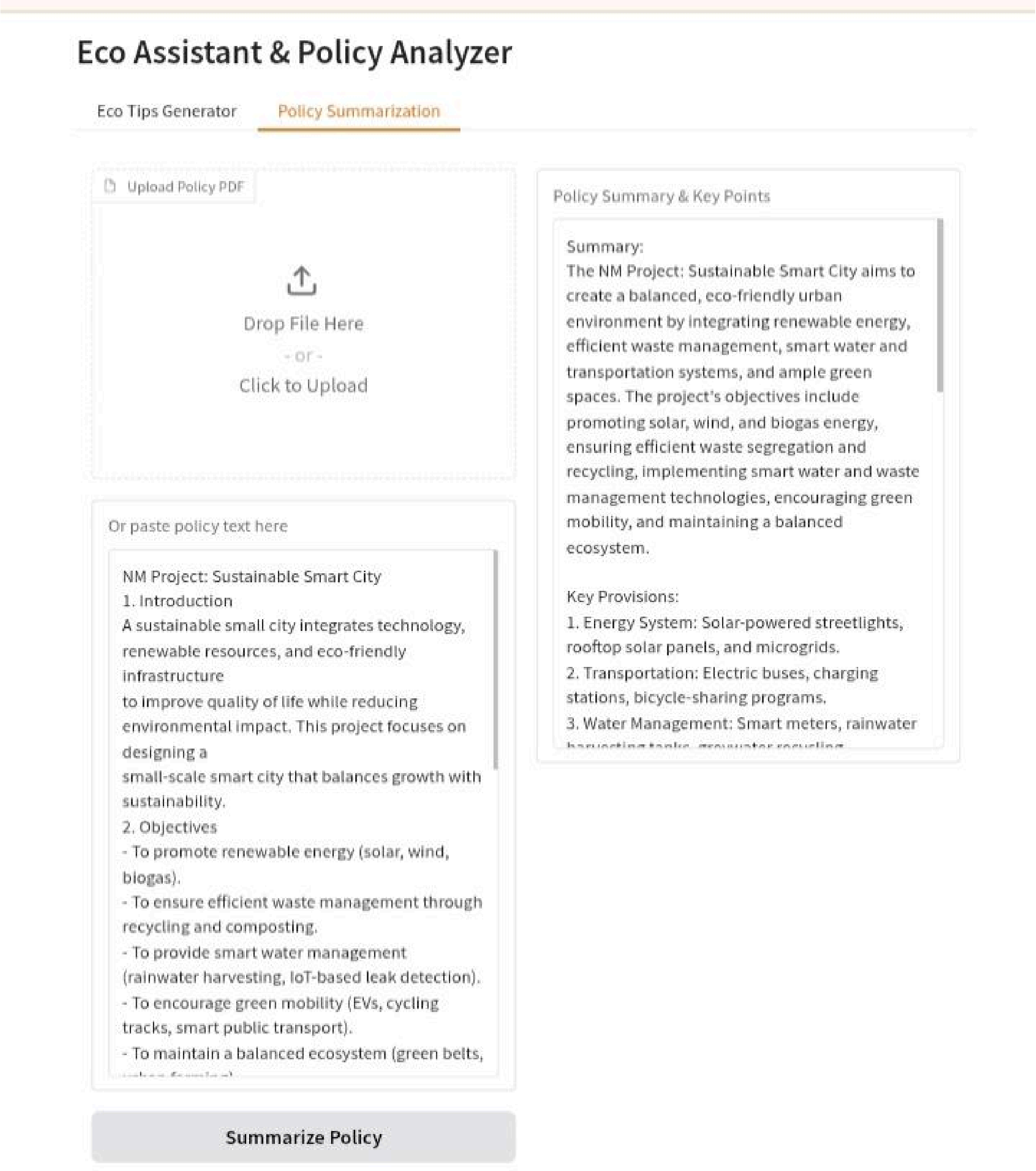
Solar energy

Generate practical and actionable eco-friendly tips for sustainable living related to: Solar energy. Provide specific solutions and suggestions:

1. **Assess Your Energy Needs:**
   - Begin by calculating your monthly energy consumption using your latest utility bills. This will help you determine the appropriate solar panel system size.
   - Consider installing a smart home energy monitor to track real-time energy usage, enabling you to make informed decisions about energy conservation.

2. **Design or Evaluate Existing Roof for Solar Panel Installation:**
   - If you're building or renovating a home, consult with a structural engineer to ensure your roof can support the weight of solar panels.

- Policy Summarisation:



Eco Assistant & Policy Analyzer

Or paste policy text here

NM Project: Sustainable Smart City
1. Introduction
A sustainable small city integrates technology, renewable resources, and eco-friendly infrastructure
to improve quality of life while reducing environmental impact. This project focuses on designing a
small-scale smart city that balances growth with sustainability.
2. Objectives
- To promote renewable energy (solar, wind, biogas).
- To ensure efficient waste management through recycling and composting.
- To provide smart water management (rainwater harvesting, IoT-based leak detection).
- To encourage green mobility (EVs, cycling tracks, smart public transport).
- To maintain a balanced ecosystem (green belts, urban farming).

Policy Summary & Key Points

Summary:
The NM Project: Sustainable Smart City aims to create a balanced, eco-friendly urban environment by integrating renewable energy, efficient waste management, smart water and transportation systems, and ample green spaces. The project's objectives include promoting solar, wind, and biogas energy, ensuring efficient waste segregation and recycling, implementing smart water and waste management technologies, encouraging green mobility, and maintaining a balanced ecosystem.

Key Provisions:
1. Energy System: Solar-powered streetlights, rooftop solar panels, and microgrids.
2. Transportation: Electric buses, charging stations, bicycle-sharing programs.
3. Water Management: Smart meters, rainwater harvesting tanks, greywater recycling