SUSTAINABLE SMART CITY ASSISTANT USING IBM GRANITE LLM

Project Documentation

1.INTRODUCTION

• Project title : SUSTAINABLE SMART CITY ASSISTANT USING IBM GRANITE LLM

• Team member : VARSHA .R(TEAM LEADER)

• Team member : DISHA . A

• Team member : KAJAL MANDAL

• Team member : SHEETALPUROHIT

2.PROJECT OVERVIEW:

• Purpose :

The Sustainable Smart City Assistant is designed to empower urban communities and local governments to build greener, more connected cities. By leveraging AI and real-time data, it optimizes critical resources such as energy, water, and waste management. It encourages sustainable habits among residents through personalized guidance and equips city planners with clear insights, forecasts, and simplified policy summaries to make informed decisions. This assistant acts as a bridge across technology, governance, and civic participation to foster cities that are efficient, inclusive, and environmentally resilient.

• Features:

Conversational Interface:

- Key Point: Natural language interaction

- Functionality: Enables residents and officials to ask questions, receive updates, and get clear guidance in everyday language.

Policy Summarization:

- Key Point: Simplified policy comprehension

- Functionality: Converts long government documents into brief, actionable summaries.

Resource Forecasting:

- Key Point: Predictive analytics- Functionality: Uses historical and real-time data to estimate future energy, water, and waste usage trends.

Eco-Tip Generator:

- Key Point: Customized sustainability advice

- Functionality: Offers daily personalized tips encouraging eco-friendly behaviors based on user habits.

Citizen Feedback Loop:

- Key Point: Community engagement

- Functionality: Collects and analyzes public feedback to improve city planning and services.

KPI Forecasting:

- Key Point: Strategic planning support

- Functionality: Projects key performance indicators to help officials track progress and plan effectively.

Anomaly Detection:

- Key Point: Early-warning mechanism

- Functionality: Detects unusual patterns in sensor or usage data to flag potential issues early.

Multimodal Input Support:

- Key Point: Versatile data handling

- Functionality: Accepts inputs in text, PDF, and CSV formats for document analysis and forecasting.

User Interface:

- Key Point: Intuitive design

- Functionality: Provides a user-friendly dashboard via Streamlit or Gradio for easy interaction by citizens and officials.

TECHNOLOGY STACK

AI Engine: IBM Watsonx Granite LLM models (granite-13b-instruct-v2, granite-3-8binstruct)

Vector Database: Pinecone for semantic search and document retrieval

Backend Framework: FastAPI for high-performance API services

Frontend Framework: Streamlit for interactive data visualization and user interface

ML Libraries: Scikit-learn for predictive modeling and anomaly detection

Data Processing: Pandas for data manipulation and CSV processing

3. **ARCHITECTURE:**

Frontend:The frontend is built with Streamlit, delivering an interactive web interface with multiple pages including dashboards, file uploaders, chat modules, feedback forms, and report viewers. Navigation is streamlined using a sidebar powered by the streamlit-option-menu library. The modular design ensures scalability.

Backend:Powered by FastAPI, the backend provides RESTful endpoints for document processing, chat services, eco tip generation, report creation, and vector embedding. It supports asynchronous processing and integrates Swagger for API documentation.

LLM Integration: IBM Watsonx Granite Large Language Models are utilized for natural language understanding and generation with expertly crafted prompts designed for summarizations, sustainability advice, and report production.

Vector Search: Policy documents are embedded into vector space via Sentence Transformers and indexed in Pinecone, enabling semantic searches using cosine similarity for natural language queries.

ML Modules:Lightweight machine learning models implemented with Scikit-learn manage forecasting and anomaly detection tasks. Time-series data is handled using pandas, with visualizations created using matplotlib.

4. SETUP INSTRUCTIONS:

Prerequisites:

- Python 3.9 or later
- pip and virtual environment management tools
- API keys for IBM Watsonx and Pinecone
- Internet access for cloud services

Installation Steps:

- Clone the project repository

- Install dependencies from requirements.txt

- Set up credentials in a .env file

- Start the FastAPI backend server

- Launch the frontend application via Streamlit

- Upload datasets and begin interacting with the system

5. FOLDER STRUCTURE:

The app/ directory contains all backend logic for **FastAPI**, including routers, models, and integrations. Subfolder **app/api/** organizes modular API routes for chat, feedback, reports, and document vectorization. The ui/ directory houses frontend components such as **Streamlit** pages, card layouts, and forms. The main application entry point is **smart_dashboard.py**. Other key modules include **granite_llm.py** for interfacing with IBM Watsonx Granite, **document_embedder.py** for creating embeddings stored in Pinecone, **kpi_file_forecaster.py** for predicting energy and water trends, **anomaly_file_checker.py** for detecting anomalies in KPI data, and **report_generator.py** for generating AI-driven sustainability reports.

6. RUNNING THE APPLICATION:

To start the project:

➢ Launch the FastAPI server to expose backend endpoints.

➢ Run the Streamlit dashboard to access the web interface.

➢ Navigate through pages via the sidebar.

➢ Upload documents or CSVs, interact with the chat assistant, and view

outputs like reports, summaries, and predictions.

➢ All interactions are real-time and use backend APIs to dynamically

update the frontend.

Frontend (Stream lit):

The frontend is built with Stream lit, offering an interactive web UI with

multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for document processing, chat interactions, eco tip generation, report creation, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

7. API DOCUMENTATION

Backend APIs available include:

POST /chat/ask – Accepts a user query and responds with an AI-generated message

POST /upload-doc – Uploads and embeds documents in Pinecone

GET /search-docs – Returns semantically similar policies to the input query

GET /get-eco-tips – Provides sustainability tips for selected topics like energy, water, or waste

POST /submit-feedback – Stores citizen feedback for later review or analytics

Each endpoint is tested and documented in Swagger UI for quick inspection and trial during development.

8. AUTHENTICATION:

Currently, the system runs openly for demonstration purposes. Future improvements will include secure authentication methods such as JWT or API key-based token authentication, OAuth2 integration using IBM Cloud credentials, role-based access control among user types like admin, citizen, and researcher, along with session and history tracking features.

9. USER INTERFACE

The interface is minimalist and functional, focusing on accessibility for non technical users.

It includes:

Sidebar with navigation

KPI visualizations with summary cards

Tabbed layouts for chat, eco tips, and forecasting

Real-time form handling

PDF report download capability

The design prioritizes clarity, speed, and user guidance with help texts and intuitive flows.

10. TESTING

Testing was done in multiple phases:

Unit Testing: For prompt engineering functions and utility scripts

API Testing: Via Swagger UI, Postman, and test scripts

Manual Testing: For file uploads, chat responses, and output consistency

Edge Case Handling: Malformed inputs, large files, invalid API keys

Each function was validated to ensure reliability in both offline and API connected modes

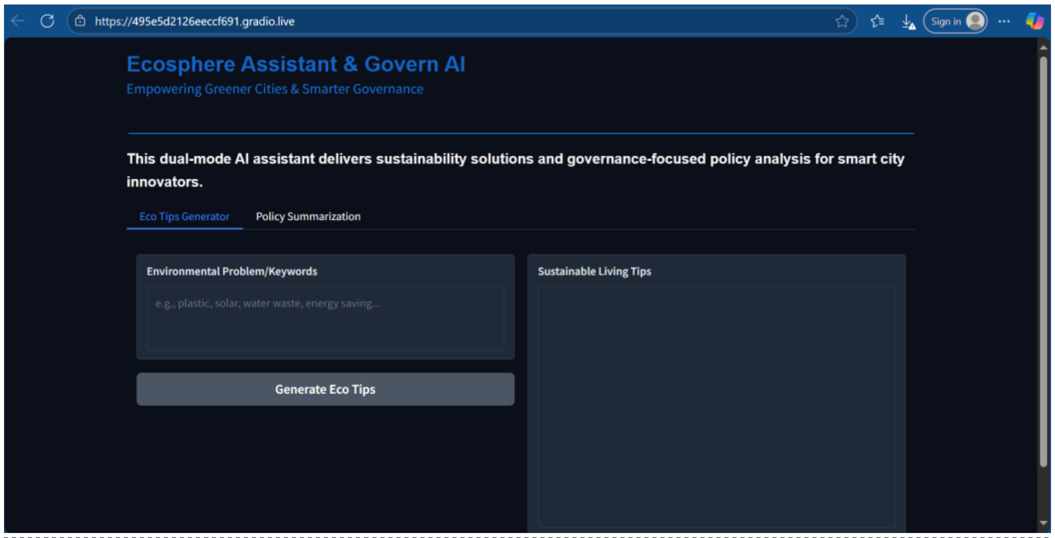
11.SOURCE CODE:

```
# -*- coding: utf-8 -*-
"""sustainble-smart-city.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1bXKnE_Atut4kHoF4Vbge7di8v9ipU52O
"""

!pip install transformers torch gradio PyPDF2 -q

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

# ---- INSTALL REQUIREMENTS ----
!pip install transformers torch gradio PyPDF2 -q

import gradio as gr
import torch
```

```
from transformers import AutoTokenizer, AutoModelForCausalLM
import PyPDF2

# ---- LOAD IBM GRANITE MODEL ----
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 related to: {problem_keywords}. Provide specific solutions and
suggestions:"
    return generate_response(prompt, max_length=1000)

def policy_summarization(pdf_file, policy_text):
    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)

# ---- THEMING (Blue, Modern) ----
theme = gr.themes.Default(
    primary_hue=gr.themes.colors.blue,
    secondary_hue=gr.themes.colors.sky,
    neutral_hue=gr.themes.colors.gray
).set(
    body_background_fill="linear-gradient(135deg, #e3f2fd 0%, #bbdefb 100%)",
    block_title_text_weight="700",
    block_label_text_size="1.15rem"
)

# ---- UI ----
with gr.Blocks(theme=theme) as app:
    gr.Markdown(
        "<h1 style='color: #1565c0; font-family: Quicksand, ui-sans-serif, sans-serif; margin-bottom: 0;'>Ecosphere Assistant & Govern AI</h1>" +
        "<span style='color: #1976d2; font-size: 1.08rem;'>Empowering Greener Cities & Smarter Governance</span><hr style='border: 1px solid #1976d2; margin-bottom: 16px;'>"
    )
    gr.Markdown(
        "<p style='font-family:Quicksand,sans-serif; color: #1976d2; font-size:1.08rem;'><b>This dual-mode AI assistant delivers sustainability solutions and governance-focused policy analysis for smart city innovators.</b></p>"
    )
    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=14)
            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=18)
            summarize_btn.click(policy_summarization, inputs=[pdf_upload, policy_text_input], outputs=summary_output)

app.launch(share=True)
```

GITHUB PROFILE LINK: https://github.com/varsha-web567/IBM-Project

12.SCREENSHOT:



https://495e5d2126eeccf691.gradio.live

Ecosphere Assistant & Govern AI

Empowering Greener Cities & Smarter Governance

This dual-mode AI assistant delivers sustainability solutions and governance-focused policy analysis for smart city innovators.

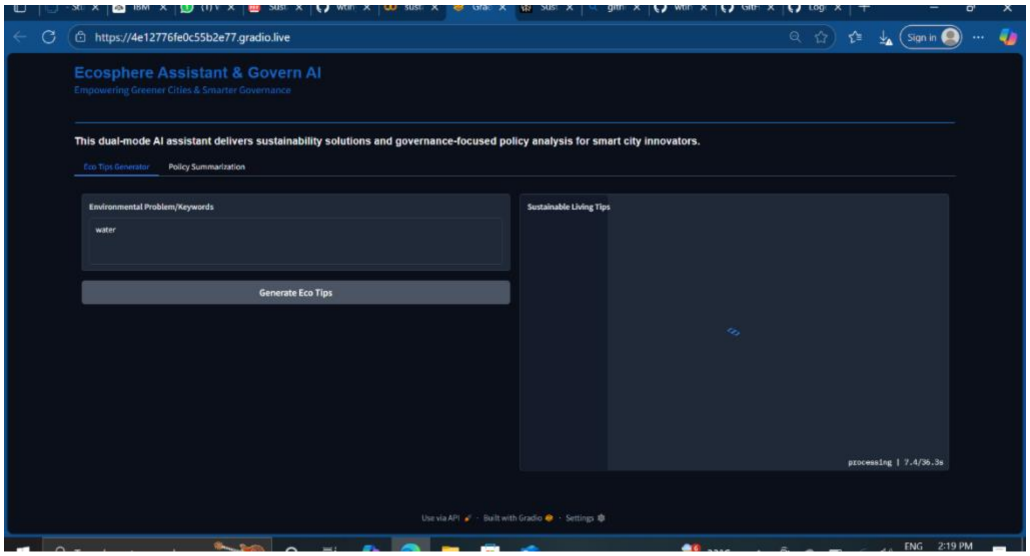
Eco Tips Generator | Policy Summarization

Environmental Problem/Keywords

e.g., plastic, solar, water waste, energy saving...

Generate Eco Tips

Sustainable Living Tips



https://4e12776fe0c55b2e77.gradio.live

Ecosphere Assistant & Govern AI

Empowering Greener Cities & Smarter Governance

This dual-mode AI assistant delivers sustainability solutions and governance-focused policy analysis for smart city innovators.

Eco Tips Generator | Policy Summarization

Environmental Problem/Keywords

water

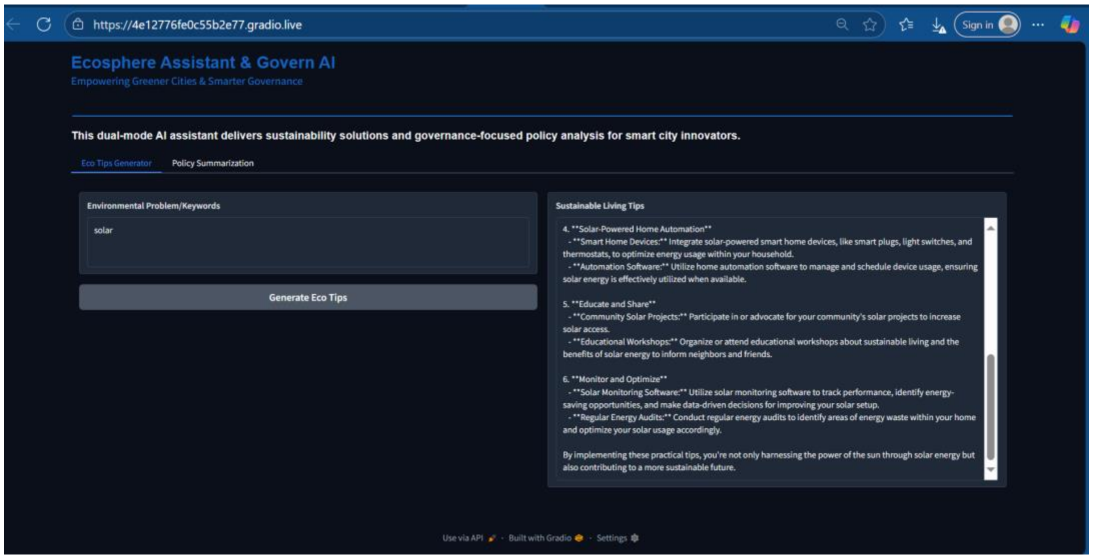
Generate Eco Tips

Sustainable Living Tips

```
processing | 7.4/36.3s
```

Use via API · Built with Gradio · Settings

ENG 2:19 PM


https://4e12776fe0c55b2e77.gradio.live
Sign in
Use via API · Built with Gradio · Settings

Ecosphere Assistant & Govern AI

Empowering Greener Cities & Smarter Governance

This dual-mode AI assistant delivers sustainability solutions and governance-focused policy analysis for smart city innovators.

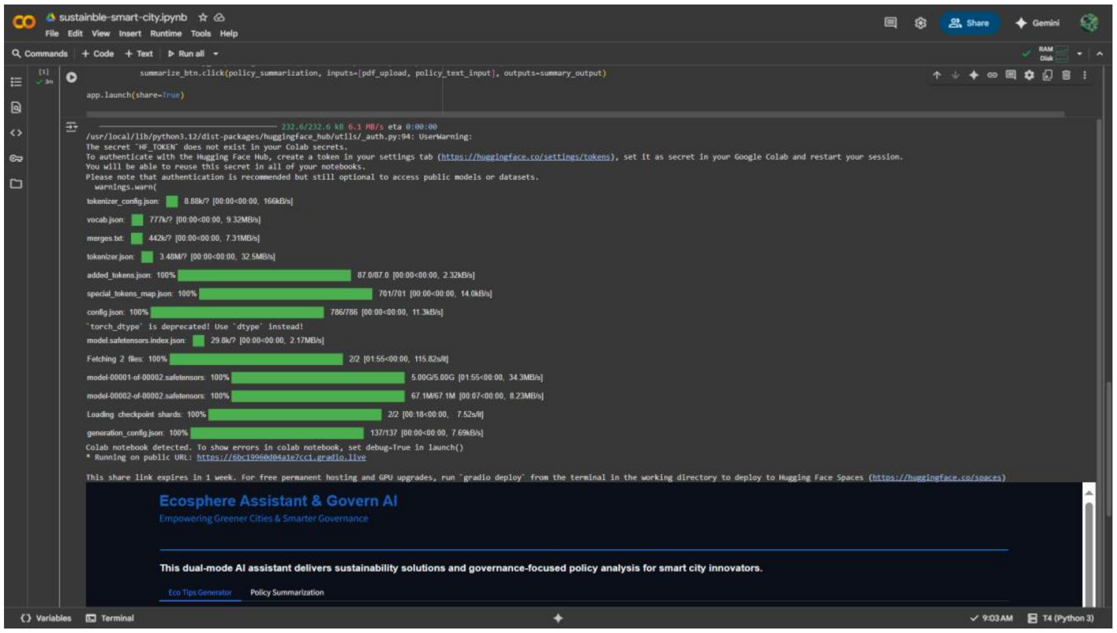
Eco Tips Generator | Policy Summarization

Environmental Problem/Keywords: solar

Generate Eco Tips

Sustainable Living Tips

```
4. **Solar-Powered Home Automation**
  - **Smart Home Devices:** Integrate solar-powered smart home devices, like smart plugs, light switches, and thermostats, to optimize energy usage within your household.
  - **Automation Software:** Utilize home automation software to manage and schedule device usage, ensuring solar energy is effectively utilized when available.

5. **Educate and Share**
  - **Community Solar Projects:** Participate in or advocate for your community's solar projects to increase solar access.
  - **Educational Workshops:** Organize or attend educational workshops about sustainable living and the benefits of solar energy to inform neighbors and friends.

6. **Monitor and Optimize**
  - **Solar Monitoring Software:** Utilize solar monitoring software to track performance, identify energy-saving opportunities, and make data-driven decisions for improving your solar setup.
  - **Regular Energy Audits:** Conduct regular energy audits to identify areas of energy waste within your home and optimize your solar usage accordingly.

By implementing these practical tips, you're not only harnessing the power of the sun through solar energy but also contributing to a more sustainable future.
```


sustainble-smart-city.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
RAM Disk
Variables Terminal
9:03 AM
T4 (Python 3)

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

app.launch(share=True)
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 6.1 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, 166kB/s]
vocab.json:  777k/? [00:00<00:00, 9.32MB/s]
merges.txt:  442k/? [00:00<00:00, 7.31MB/s]
tokenizer.json:  3.48M/? [00:00<00:00, 32.5MB/s]
added_tokens.json: 100%  87.0/87.0 [00:00<00:00, 2.32kB/s]
special_tokens_map.json: 100%  701/701 [00:00<00:00, 14.0kB/s]
config.json: 100%  786/786 [00:00<00:00, 11.3kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:  29.8k/? [00:00<00:00, 2.17MB/s]
Fetching 2 files: 100%  2/2 [01:55<00:00, 115.82s/it]
model-00001-of-00002.safetensors: 100%  5.00G/5.00G [01:55<00:00, 34.3MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:07<00:00, 8.23MB/s]
Loading checkpoint shards: 100%  2/2 [00:18<00:00,  7.52s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 7.69kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://6bc19960d04a1e7cc1.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)
```

Ecosphere Assistant & Govern AI

Empowering Greener Cities & Smarter Governance

This dual-mode AI assistant delivers sustainability solutions and governance-focused policy analysis for smart city innovators.

Eco Tips Generator | Policy Summarization

13. KNOWN ISSUES:

Known IssuesData Privacy Concerns: Handling vast amounts of personal and urban data raises significant privacy and data ownership issues, requiring robust protection and clear regulations.

Integration Challenges: Integrating diverse technologies, systems, and legacy infrastructure is complex and may encounter compatibility issues.

Digital Divide: Not all citizens have equal access to digital tools and skills, which may lead to inequality in benefiting from smart city services.

System Reliability: Ensuring real-time responsiveness, network reliability, and robust operation during outages or cyber threats remains a persistent challenge.

Technological Reductionism: Smart solutions risk focusing too much on technology rather than holistic urban development and citizen participation, potentially depoliticizing city planning.

14.FUTURE ENHANCEMENT:

Future EnhancementsAdvanced Data Security: Enhance encryption, implement decentralized data storage, and enforce stricter privacy policies.

AI-Driven Participation: Introduce more participatory features, such as real-time citizen engagement platforms, and inclusive decision-making tools.

Edge and Cloud Integration: Leverage edge computing for faster processing, reduced latency, and improved privacy, while maintaining scalable cloud analytics.

Adaptable Policy Engines: Enable automatic policy adaptation in response to evolving urban data, including environmental and social changes.

Inclusive Design: Design user interfaces and services that address barriers faced by underserved or digitally less literate populations.

Predictive Urban Management: Expand AI models for forecasting emergencies, optimizing public resources, and enhancing urban resilience (e.g., climate adaptation, resource allocation).

Transparent Governance: Enhance transparency and trust by making algorithmic decisions, data usage, and city operations more open to public understanding and audit.