Sustainable Smart City Assistant Using IBM Granite LLM

Project Documentation

1. Introduction

- Project Title : Sustainable Smart City Assistant Using IBM Granite LLM
- Team Member 1 : Noor Mohamed.J
- Team Member 2 : Praseeman.S
- Team Member 3 : Arsath Mohammed.B
- Team Member 4 : Abdul Rahman.J.B
- Team Member 5 : Darwin.

2. Project Overview

Purpose:

The Sustainable Smart City Assistant is designed to empower cities and their residents to thrive in a more eco-conscious and connected urban environment. By leveraging **IBM Granite LLM, AI, and real-time data**, the assistant helps optimize essential resources like energy, water, and waste while guiding citizens with sustainable habits.

For **citizens**, it acts as a digital guide that provides personalized eco-tips, policy explanations, and sustainability recommendations. For **officials**, it serves as a decision-making partner by summarizing policies, forecasting resources, detecting anomalies, and analyzing citizen feedback.

Key Features:

- **Conversational Interface –** Interact with the assistant in natural language.
- **Policy Summarization –** Converts lengthy documents into short, clear summaries.

- **Resource Forecasting –** Predicts energy, water, and waste usage.
- **Eco-Tip Generator –** Provides practical and daily eco-friendly suggestions.
- **Citizen Feedback Loop –** Collects and analyzes public inputs.
- **KPI Forecasting –** Helps in long-term planning for sustainable growth.
- **Anomaly Detection –** Flags unusual usage patterns for early warning.
- **Multimodal Input Support –** Handles text, PDFs, and CSVs.
- **Streamlit/Gradio UI –** A clean dashboard for citizens and officials.

3. System Architecture

- **Frontend (Streamlit/Gradio):**
    - User-friendly interface with tabs for Eco Tips, Policy Summarization, Reports, and Forecasts.
    - File upload support (PDF/CSV).
    - Real-time response display.
- **Backend (FastAPI):**
    - REST APIs to process queries, documents, and feedback.
    - Scalable, asynchronous, and easy integration with Swagger UI.
- **LLM Integration (IBM Granite LLM):**
    - Used for natural language summarization, eco-tip generation, and conversational responses.
    - Optimized for policy document analysis and smart recommendations.
- **Vector Search (Pinecone):**
    - Semantic search on uploaded policies using embeddings.
    - Allows users to query policies in plain language.

- **ML Modules (Scikit-learn):**
    - Forecasting energy/water usage.
    - Detecting anomalies in KPI data.

4. Setup Instructions

Prerequisites:

- Python 3.9+
- pip and venv tools
- API keys for IBM Watsonx & Pinecone
- Internet access

Installation:

1. Clone the repository
2. Install dependencies:
3. pip install -r requirements.txt
4. Create a .env file and add credentials (IBM API key, Pinecone key).
5. Run backend:
6. uvicorn app.main:app --reload
7. Run frontend:
8. streamlit run smart_dashboard.py

5. Folder Structure

```
project/
|—— app/                    # Backend with FastAPI
```

```
│   ├── api/                  # API routes
│   ├── models/               # Data models
│   └── services/             # Integration with LLM, Pinecone
├── ui/                       # Streamlit/Gradio interface
├── granite_llm.py            # IBM Granite integration
├── document_embedder.py      # Vector embeddings
├── kpi_file_forecaster.py    # Forecasting module
├── anomaly_file_checker.py   # Anomaly detection
├── report_generator.py       # Report creation
├── smart_dashboard.py        # Entry dashboard script
└── requirements.txt          # Dependencies
```

6. Running the Application

1. Start **FastAPI server**
2. Run **Streamlit/Gradio UI**
3. Navigate via sidebar tabs:
    - Upload PDF (Policy Summarization)
    - Enter keywords (Eco Tips Generator)
    - Check anomaly/forecast outputs
4. View reports & download results

7. API Documentation

- **POST /chat/ask** → AI-powered Q&A

- **POST /upload-doc** → Upload + Embed policy documents
- **GET /search-docs** → Search policy documents by query
- **GET /get-eco-tips** → Generate eco-friendly tips
- **POST /submit-feedback** → Store citizen feedback

APIs are tested with Swagger UI.

8. Authentication

- Supports **JWT tokens**, **OAuth2 with IBM Cloud**, and **role-based access**.
- Demo version runs open-access for testing.

9. User Interface

- **Sidebar Navigation** (Eco Tips, Policy Summarization, Forecasting, Reports).
- **Tabbed Layouts** for better organization.
- **Real-time Outputs** – Policy summaries, eco-tips, and anomaly flags.
- **Report Download** (PDF).

10. Testing

- **Unit Testing** – Prompt responses and ML models.
- **API Testing** – Swagger & Postman.
- **Manual Testing** – File uploads, summarization, anomaly detection.
- **Edge Cases** – Invalid inputs, empty PDFs, missing API keys.

11. Screenshots

SmartCity.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Run all

RAM Disk

[1] 12s

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

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.56.1)
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.8.0+cu126)
Requirement already satisfied: gradio in /usr/local/lib/python3.12/dist-packages (5.44.1)
Collecting PyPDF2
  Downloading pypdf2-3.0.1-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (3.19.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.34.4)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.22.0)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch) (12.5.4.2)
```

Variables Terminal 6:30 PM T4 (Python 3)

SmartCity.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Run all

RAM Disk

[2] 2m

```
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 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}"
```

Variables Terminal 6:30 PM T4 (Python 3)

SmartCity.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Run all

RAM Disk

[2] 2m

```
        )

    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):
    # 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}"
```

Variables Terminal 6:30 PM T4 (Python 3)

SmartCity.ipynb

File Edit View Insert Runtime Tools Help

Commands | + Code | + Text | Run all

RAM Disk

Share | Gemini

```
[2]
✓ 2m
    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
                    )
                    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.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
                    )
```

Variables | Terminal | 6:30 PM | T4 (Python 3)

SmartCity.ipynb

File Edit View Insert Runtime Tools Help

Commands | + Code | + Text | Run all

RAM Disk

Share | Gemini

```
[2]
✓ 2m
                    )
                    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)
```

```
/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
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, 161kB/s]
vocab.json: 777k/? [00:00<00:00, 7.85MB/s]
merges.txt: 442k/? [00:00<00:00, 15.0MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 28.6MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 3.78kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 16.7kB/s]
config.json: 100% 786/786 [00:00<00:00, 24.2kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.74MB/s]
```

Variables | Terminal | 6:30 PM | T4 (Python 3)

Eco Assistant & Policy Analyzer

Eco Tips Generator | Policy Summarization

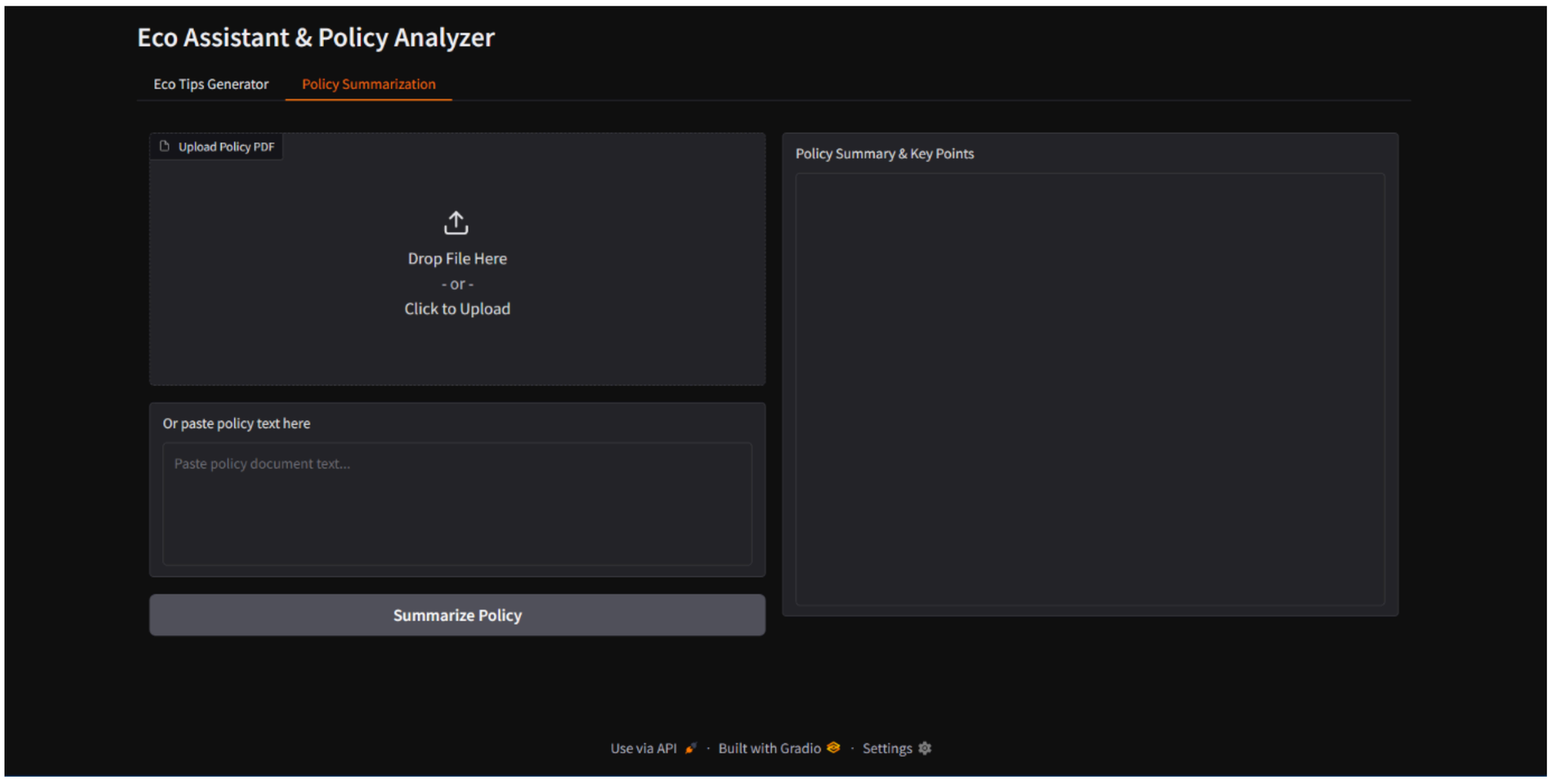
Environmental Problem/Keywords

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

Generate Eco Tips

Sustainable Living Tips

Use via API · Built with Gradio · Settings



12. Known Issues

- Occasional long response time for large PDFs.
- Forecasting limited to structured CSV data.
- Requires stable internet for IBM API access.

13. Future Enhancements

- Add **voice-based interaction**.
- Expand forecasting to include **traffic & pollution data**.
- Develop a **mobile app version**.
- Integrate with **IoT smart sensors**.
- Support **multi-language outputs** for local communities.