Sustainable Smart City Assistant – Project Documentation

---

1. Introduction

Project Title : Sustainable Smart City Assistant

Team Members :

Subha Sree.S

Ravi shankar

Ram Sakthi surya Prakash

Sachien

---

2. Project Overview

Purpose

The purpose of the Sustainable Smart City Assistant is to empower cities and residents to adopt more eco-friendly and data-driven lifestyles. Using AI and real-time policy analysis, the assistant helps optimize resources like energy, water, and waste. It also guides citizens with daily eco-tips and enables city officials to better understand complex policy documents through AI-generated summaries.

This tool bridges technology, governance, and citizen engagement to promote greener, more inclusive, and resilient cities.

Features

1. Eco-Tip Generator

Key Point: Personalized sustainability advice

Functionality: Generates practical eco-friendly living tips based on keywords like water, energy, plastic, solar.

2. Policy Summarization

Key Point: Simplified policy understanding

Functionality: Upload a policy PDF or paste text, and the assistant provides a concise summary with key points and implications.

3. Conversational AI

Key Point: Natural language interaction

Functionality: Uses IBM Granite LLM to generate natural, contextual responses.

4. PDF Text Extraction

Key Point: Flexible document handling

Functionality: Reads and extracts text from PDF policy files using PyPDF2.

5. User-Friendly Interface

Key Point: Gradio Web UI

Functionality: Offers an interactive dashboard with tabs for eco tips and policy analysis.

---

3. Architecture

Frontend (Gradio):

The application uses Gradio as the frontend. It provides a simple web-based interface with two main tabs:

Eco Tips Generator

Policy Summarization

Backend (Transformers + PyTorch):

The IBM Granite LLM is used for natural language processing tasks, running on CPU/GPU.

PDF Processing (PyPDF2):

Extracts raw text from policy documents for summarization.

Workflow:

1. User enters keywords or uploads a PDF.

2. Input is processed through tokenizer → Granite model.

3. AI generates eco tips or summaries.

4. Output is displayed in the Gradio UI.

---

4. Setup Instructions

Prerequisites:

Python 3.9 or later

pip installed

Internet connection

Installation Steps:

1. Clone the project repository.

2. Install required dependencies:

pip install transformers torch gradio PyPDF2

3. Run the script:

python sustainable\_smart\_city\_assistance.py

4. The Gradio app will launch in your browser.

---

5. Folder Structure

project/

│

├── sustainable\_smart\_city\_assistance.py # Main app

├── requirements.txt # Dependencies

├── documents/ # Policy PDFs (optional)

└── outputs/ # Summaries, tips (optional)

---

6. Running the Application

1. Launch the script with Python.

2. Open the Gradio link in your browser.

3. Use the Eco Tips Generator tab:

Enter keywords like plastic, water saving, solar.

Click Generate Eco Tips.

View AI-generated suggestions.

4. Use the Policy Summarization tab:

Upload a policy PDF or paste text.

Click Summarize Policy.

View summarized output.

---

7. Functions & API Documentation

1. Eco Tips Generator

Input: Keywords (string)

Output: Actionable eco tips (string)

Example: Input → plastic, energy saving → Output → Use reusable bags, install LED lights, minimize standby power.

2. Policy Summarization

Input: PDF file or text input

Output: Concise summary with key points and implications

Example: Upload → Water Conservation Policy → Output → The policy focuses on groundwater recharge, rainwater harvesting, and waste-water recycling.

3. Text Extraction (from PDF)

Input: PDF document

Output: Extracted text (string)

---

8. Authentication

This project currently runs in open mode (no authentication).

For production:

Add API key authentication

Add role-based access control (citizen, admin, researcher)

Enable secure sessions

---

9. User Interface

Framework: Gradio Blocks

Tabs:

Eco Tips Generator

Policy Summarization

Widgets: Textbox, File Upload, Button, Output Box

Design: Minimal, user-friendly, suitable for non-technical users

---

10. Testing

Unit Testing: Model prompt responses validated.

PDF Testing: Verified extraction from multiple policy PDFs.

Manual Testing: Checked UI interaction with eco tips and summarization.

Edge Cases:

Empty input

Large PDFs

Unsupported file formats

---

11. Future Enhancements

Add Resource Forecasting (predict energy/water consumption trends)

Add Citizen Feedback Loop (collect public feedback via UI)

Include KPI Monitoring Dashboard

Add Mobile App Integration

Implement Anomaly Detection for resource misuse patterns