Sustainable Smart City Assistant Using IBM Granite LLM
Project Documentation

Introduction

Project Title:Sustainable Smart City Assistant Using IBM Granite LLM
Team member: Hrithik P
Team member: Gogulnath V
Team member: Hari Babu C H
Team member: Jackdivyan J

Eco Assistant & Policy Analyzer is an AI-powered tool designed to simplify the way users engage with sustainability information. It helps generate quick, one-line eco tips and summarizes lengthy policy documents into clear, plain-language text. The solution uses a simple Gradio-based interface and integrates with Google Gemini 1.5 Flash to provide fast and accurate responses. Users can enter any topic related to sustainability, such as plastic waste, and instantly receive concise, actionable tips. They can also upload a policy document or paste its content to obtain a short, three- to four-sentence summary, reducing the time and effort needed to understand complex information.

Project Overview

The purpose of this project is to encourage sustainable living and promote environmental awareness by giving individuals, students, and organizations an easy way to access eco-friendly actions and policy insights. The scope of the project includes both individuals seeking quick suggestions for a greener lifestyle and organizations such as NGOs or eco clubs that need to create awareness materials. The application currently supports generating eco tips based on a keyword entered by the user and summarizing policy documents uploaded in PDF format or pasted as plain text. The output is intentionally kept short, simple, and easy to understand, allowing users to apply the information quickly. The interface is clean, runs locally using Python, and does not require complex setup. At present, the application does not provide user accounts, role-based access, or vector search capabilities, but these are considered for future development.

Architecture

Eco Assistant & Policy Analyzer is structured with a simple yet effective architecture. The frontend is built using Gradio and consists of two main workflows: the eco tips generator and the policy summarization tool. When a user provides input, whether it is a keyword or a document, the system assembles a prompt and sends it to the Gemini 1.5 Flash API. The API returns a concise response which is then displayed in the Gradio interface. The entire application is contained in a single Python file, which makes it lightweight and easy to deploy.

Setup Instructions

The project can be set up by first ensuring that Python 3.9 or above is installed. The repository can then be cloned from GitHub, and the required dependencies can be installed using the provided requirements file. The user must set their Google Gemini API key as an environment variable before running the application. Once these steps are completed, the application can be started using the Python command, and Gradio will provide a local URL for the user to access the interface.

Folder Structure

The repository contains a small set of files, including the main Python script that defines the Gradio interface and logic, a README file with instructions, a license file specifying the MIT license, and a requirements file listing the dependencies. As the project evolves, the folder structure can be expanded to include organized directories for prompts, utility scripts, user interface components, tests, and documentation.

Running the Application

After launching the Gradio app, the user is greeted with a simple interface. They can choose to either generate eco tips or summarize a policy. In the eco tips tab, a keyword such as “plastic” can be entered to generate three short, actionable suggestions. In the policy summary tab, a PDF can be uploaded or text can be pasted, and the system will produce a three- to four-sentence summary that captures the essence of the policy. The results can then be copied and used directly in reports or presentations.

API Documentation

At this stage, the application does not expose public REST endpoints and operates as a local interface only. In future versions, a FastAPI layer can be introduced to expose endpoints for tasks such as policy summarization, eco tip generation, and service health checks.

Authentication and Security

The current version uses the Google Gemini API key stored as an environment variable to connect to the language model. Since this is a single-user local application, there are no authentication layers or role-based permissions. Best practices recommend keeping the API key in a secure .env file and never committing it to version control. Future versions can include a FastAPI server with JWT-based authentication or API key validation for multi-user scenarios.

User Interface

The interface has been designed for simplicity, with two clearly defined sections for generating eco tips and summarizing policies. The results are presented in plain English with short, easy-to-read sentences, making it suitable for all users. The interface is intuitive and allows quick interaction without the need for technical knowledge.

Testing

Testing for the application can include checking that the eco tips generator produces exactly three concise tips, verifying that the summaries are three to four sentences long and relevant to the content, and ensuring that the application gracefully handles empty inputs, non-text PDFs, and large documents. Latency tests can be performed to confirm that responses are generated quickly, and a simple smoke test can be run to verify that the application launches and both main features work correctly.

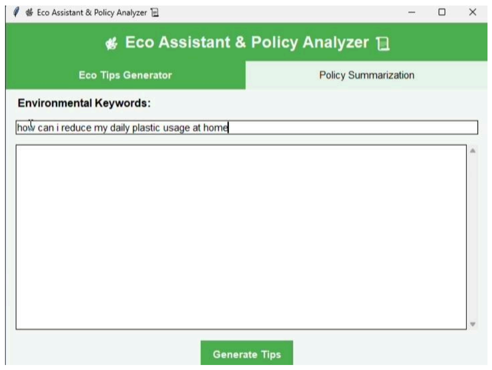
Known Issues

Some limitations include dependency on the quality of input documents, meaning scanned PDFs without text recognition may fail to summarize properly. Additionally, there is no export option, so users must copy results manually. API or network errors currently display generic messages, which can be improved with more descriptive feedback.

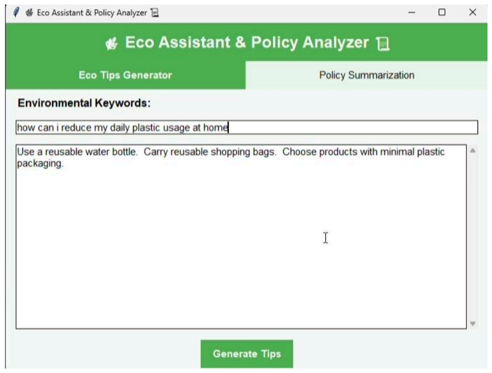
Future Enhancements

Future development could include adding options to save outputs as PDF or DOCX files and providing a history of generated results. The application could be extended with a FastAPI backend to allow REST API integration, enabling use in other applications. A vector search feature could be implemented to store multiple policy documents and answer questions across them. A dashboard could be added to display key metrics such as energy, water, and waste data with visual charts, and basic forecasting could be used to project trends or detect anomalies. Role-based views could be introduced to offer tailored outputs for citizens, researchers, and policymakers. Finally, multilingual support could be added to make the tool accessible to a wider audience.

Screenshots

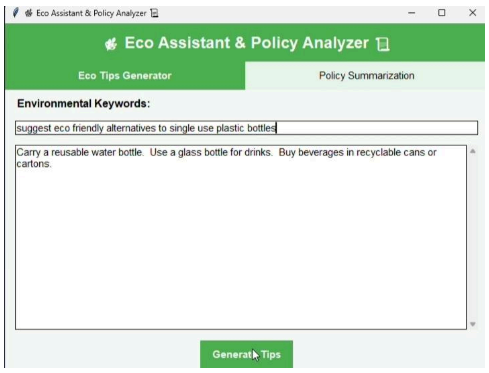


how can i reduce my daily plastic usage at home



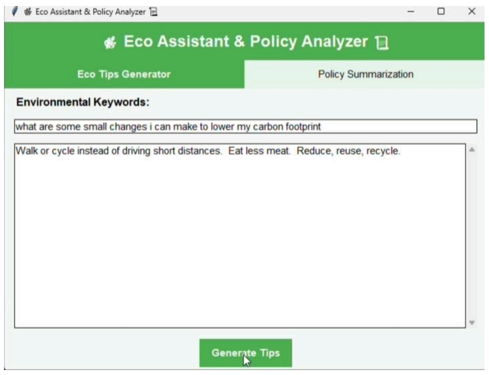
how can i reduce my daily plastic usage at home

Use a reusable water bottle. Carry reusable shopping bags. Choose products with minimal plastic packaging.



suggest eco friendly alternatives to single use plastic bottles

Carry a reusable water bottle. Use a glass bottle for drinks. Buy beverages in recyclable cans or cartons.



what are some small changes i can make to lower my carbon footprint

Walk or cycle instead of driving short distances. Eat less meat. Reduce, reuse, recycle.

Conclusion

Eco Assistant & Policy Analyzer offers a practical and lightweight way to spread environmental awareness and simplify policy understanding. By combining AI-powered summarization with a user-friendly interface, it empowers individuals and organizations to make informed, eco-friendly choices quickly and efficiently. With future improvements, the project has the potential to grow into a more comprehensive sustainability assistant and policy analysis platform.