

Sustainable Smart City Assistance

project documentation

1.Introduction

- Project title : Sustainable Smart City Assistance
- Team member: kiruthika M
- Team member: Kaviya M
- Team member: Nathiya S
- Team member: Jeevitha M

2.project overview

- Purpose :

The purpose of a Sustainable Smart City Assistant is to empower cities and their residents to thrive in a more eco-conscious and connected urban environment. By leveraging AI and real-time data, the assistant helps optimize essential resources like energy, water, and waste, while also guiding sustainable behaviors among citizens through personalized tips and services. For city officials, it serves as a decision-making partner—offering clear insights, forecasting tools, and summarizations of complex policies to support strategic planning. Ultimately, this assistant bridges technology, governance, and community engagement to foster greener cities that are more efficient, inclusive, and resilient.
- Features:

Policy Summarization

Key Point: Simplified policy understanding

Functionality: Converts lengthy government documents into concise, actionable summaries.

Eco-Tip Generator

Key Point: Personalized sustainability advice

Functionality: Recommends daily actions to reduce environmental impact based on user behavior.

3. Architecture

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.

LLM Integration (IBM Watsonx Granite):

Granite LLM models from IBM Watsonx are used for natural language understanding and generation. Prompts are carefully designed to generate summaries, sustainability tips, and reports.

Vector Search (Pinecone):

Uploaded policy documents are embedded using Sentence Transformers and stored in Pinecone. Semantic search is implemented using cosine similarity to allow users to search documents using natural language queries.

ML Modules (Forecasting and Anomaly Detection):

Lightweight ML models are used for forecasting and anomaly detection using Scikit-learn. Time-series data is parsed, modeled, and visualized using pandas and matplotlib.

4. Setup Instructions

Prerequisites:

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

Installation Process:

- o Clone the repository
- o Install dependencies from requirements.txt
- o Create a .env file and configure credentials
- o Run the backend server using Fast API
- o Launch the frontend via Stream lit
- o Upload data and interact with the modules

5. Folder Structure

app/ – Contains all Fast API backend logic including routers, models, and integration modules.

app/api/ – Subdirectory for modular API routes like chat, feedback, report, and document vectorization.

ui/ – Contains frontend components for Stream lit pages, card layouts, and form UIs.

smart_dashboard.py – Entry script for launching the main Stream lit dashboard.

granite_llm.py – Handles all communication with IBM Watsonx Granite model including summarization and chat.

document_embedder.py – Converts documents to embeddings and stores in Pinecone.

kpi_file_forecaster.py – Forecasts future energy/water trends using regression.

anomaly_file_checker.py – Flags unusual values in uploaded KPI data.

report_generator.py – Constructs AI-generated 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

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

This version of the project runs in an open environment for demonstration. However, secure deployments can integrate:

- Token-based authentication (JWT or API keys)
- OAuth2 with IBM Cloud credentials
- Role-based access (admin, citizen, researcher)
- Planned enhancements include user sessions and history tracking.⁸

Authentication

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.Add screen shots

Eco Assistant & Policy Analyzer

Eco Tips Generator



Upload Policy PDF



9af8f-govern... .pdf

3.4 MB ↓

Or paste policy text here

Paste policy document text...

Summarize Policy

Policy Summary & Key Points

1.1. Process of Public Policy Formation

The process of public policy formation involves several stages, each requiring careful consideration and collaboration among various stakeholders. These stages typically include:

- Identification of policy issues or problems: This involves recognizing and acknowledging specific

Eco Assistant & Policy Analyzer

Eco Tips Generator



Environmental Problem/Keywords

How to save oceans from plastic waste

Generate Eco Tips

Sustainable Living Tips

1. Reduce single-use plastics:
 - Replace disposable plastic items with reusable alternatives, such as cloth shopping bags, water bottles, and straws.
 - Carry a reusable container for leftovers and unpackaged food purchases.
 - Opt for products with minimal or no plastic packaging.
2. Proper waste disposal:
 - Familiarize yourself with local recycling guidelines and ensure your plastic waste is sorted correctly.
 - Participate in community clean-up events or organize your own, focusing on plastic pollution.

12.known issues

- Data Reliability – Forecasting accuracy depends heavily on the quality of real-time and historical data; poor sensor data or incomplete datasets may reduce effectiveness.
- Scalability Challenges – Current prototype may struggle with large-scale deployment across multiple cities without infrastructure upgrades.
- Latency in Processing – Real-time responses may experience delays when handling very large documents or simultaneous requests.
- Limited Policy Coverage – Policy summarization might miss nuances in lengthy government regulations.
- Anomaly Detection Accuracy – False positives/negatives can occur in anomaly detection due to noisy sensor data.
- Basic UI Design – The current Streamlit/Gradio interface is functional but lacks advanced user experience design.
- Authentication Gaps – The demo runs in an open environment; full security (role-based access, encryption) is not yet implemented.
- Offline Limitations – Features that rely on cloud APIs (IBM Watsonx, Pinecone) may not work in environments with poor internet connectivity.

13.Future enhancement

- Advanced Security Integration – Add multi-factor authentication, role-based permissions, and data encryption for secure deployments.
- Mobile App Support – Extend the system to Android/iOS apps for citizen accessibility on the go.
- Multi-Language Support – Provide translations and natural language interaction in regional languages.
- IoT Device Integration – Directly connect with smart meters, waste bins, and water sensors for more granular real-time monitoring.
- Predictive City Planning – Use AI simulations to model the impact of policy changes on traffic, pollution, and resource usage.
- AI-Powered Visualizations – Introduce dashboards with advanced charts, geospatial maps, and interactive scenario analysis.
- Citizen Reward System – Implement gamification (reward points for eco-friendly actions) to encourage participation.
- Offline Mode with Sync – Allow data collection offline and auto-sync when internet connectivity is restored.
- Integration with External APIs – Connect to weather APIs, transport data, and energy providers for richer forecasting.
- Automated Reporting – Monthly sustainability reports auto-generated and emailed to city officials.