Project Title

Project Documentation

Sustainable Smart City Assistant Using IBM Granite LLM

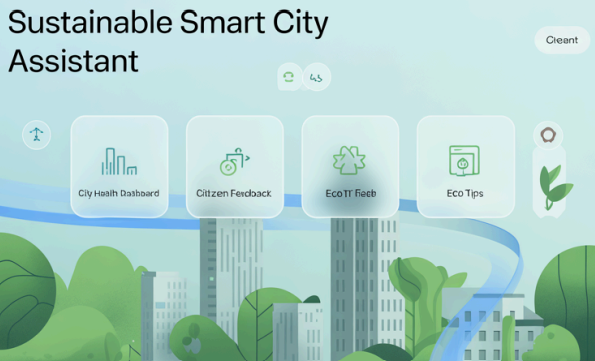
**1.Introduction**

• Team Leader : MANOJ M

• Team Member : MUTHUSELVAN R

• Team Member : SATHIRIYAN B

• Team Member : RAVEENDRAN R



**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 decisionmaking 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.

**Conversational Interface**

*Key Point:* Natural language interaction

*Functionality:* Allows citizens and officials to ask questions, get updates, and receive guidance in plain language

**Policy Summarization**

*Key Point:* Simplified policy understanding

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

**Resource Forecasting**

*Key Point:* Predictive analytics

*Functionality:* Estimates future energy, water, and waste usage using historical and real-time data.

**Eco-Tip Generator**

*Key Point:* Personalized sustainability advice

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

**Citizen Feedback Loop**

*Key Point:* Community engagement

*Functionality:* Collects and analyzes public input to inform city planning and service improvements.

**KPI Forecasting**

*Key Point:* Strategic planning support

*Functionality:* Projects key performance indicators to help officials track progress and plan ahead.

**Anomaly Detection**

*Key Point:* Early warning system

*Functionality:* Identifies unusual patterns in sensor or usage data to flag potential issues.

**Multimodal Input Support**

*Key Point:* Flexible data handling

*Functionality:* Accepts text, PDFs, and CSVs for document analysis and forecasting.

**Streamlit or Gradio UI**

*Key Point:* User-friendly interface

*Functionality:* Provides an intuitive dashboard for both citizens and city officials to interact with the assistant.

# 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:**

* 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 nontechnical 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

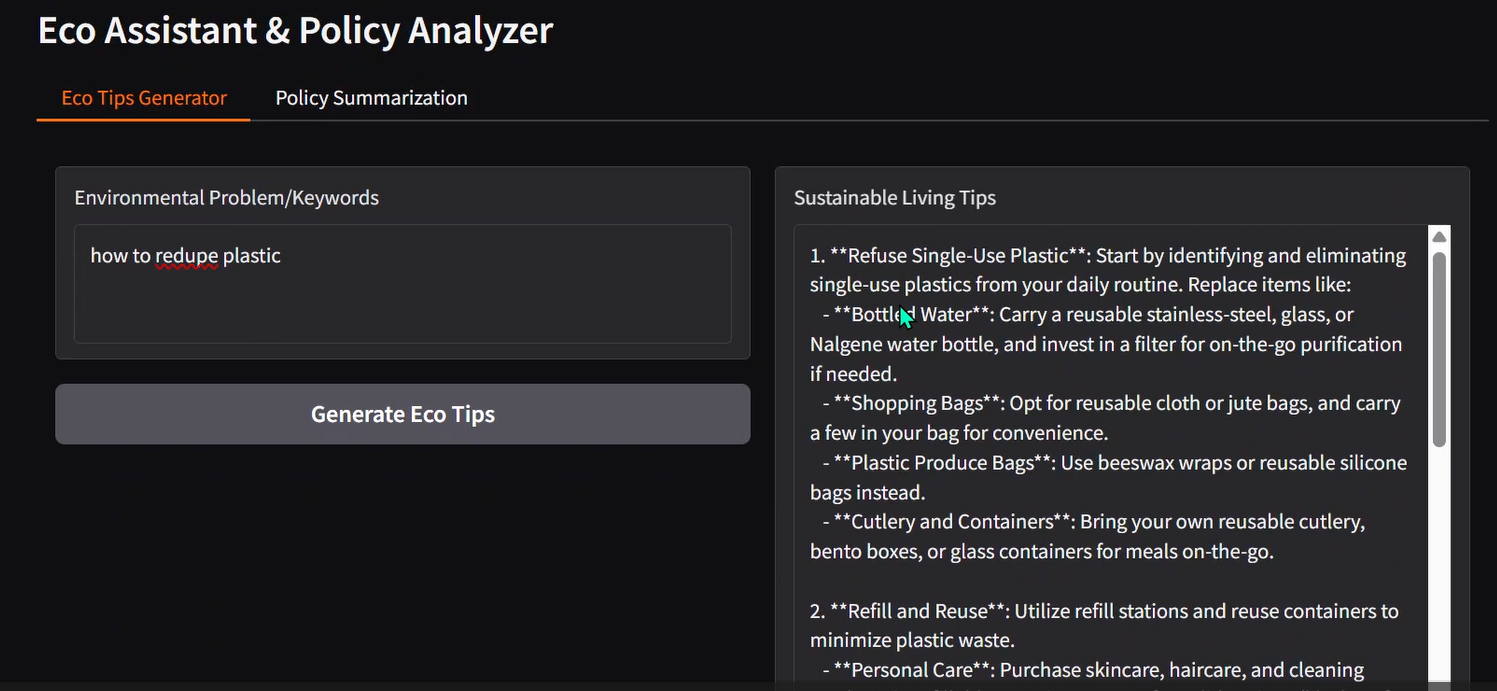
API Testing: Via Swagger UI, Postman, and test scripts

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 APIconnected modes.

**11.screen shots**



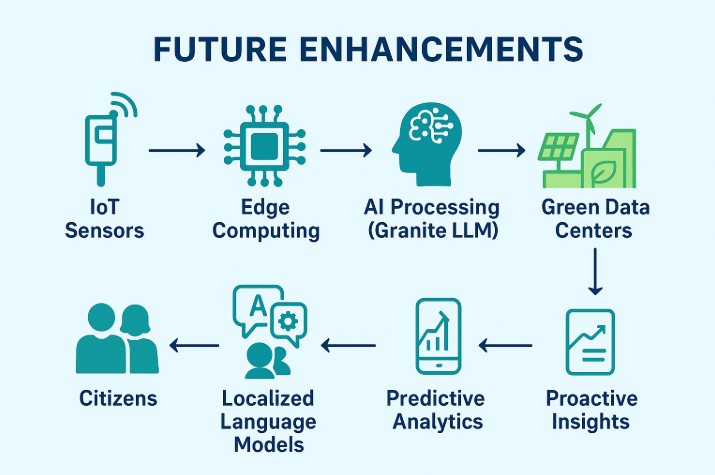
1. **Known Issues**

Although the Sustainable Smart City Assistant using IBM Granite LLM has significant potential, there are several challenges that limit its effectiveness. One of the primary issues lies in data availability and accuracy. The system depends heavily on real-time information collected from IoT devices, city sensors, and existing digital platforms. However, in many urban environments, these infrastructures are either incomplete, outdated, or inconsistent. As a result, the assistant may sometimes provide inaccurate or delayed insights, particularly if traffic sensors fail, pollution monitors stop transmitting, or energy consumption data is missing.

Scalability is another major concern. In large metropolitan areas, enormous volumes of data are generated every second. Processing this data in real time requires advanced infrastructure with powerful servers and optimized algorithms. Without such resources, the system may suffer from latency, delayed responses, or even downtime, which reduces its reliability for citizens and city officials.

Language and dialect variations also pose a challenge. While IBM Granite LLM supports multiple languages, urban populations are highly diverse and often communicate in regional dialects, slang, or even mixed-language queries. In such cases, the assistant may misinterpret user intent and fail to provide relevant answers, thereby limiting accessibility for non-standard language users.

1. **Future enhancement**



The Sustainable Smart City Assistant has strong potential for future growth by overcoming its present challenges. A key direction for enhancement is the integration of **edge computing** alongside cloud-based processing. Instead of sending all raw data to central servers, small processors attached to traffic lights, energy meters, and waste bins could analyze information locally. This would allow the system to make **real-time decisions** such as adjusting traffic signals during peak hours without waiting for cloud instructions, thereby reducing congestion and fuel wastage.

Another future development lies in the creation of **green data centers** powered by renewable energy sources like solar and wind. Since large AI models demand high computational power, this step ensures that the assistant itself does not add to the city’s carbon footprint. Additionally, **lightweight AI models** could be deployed on citizen smartphones to offer essential features such as waste segregation tips, water conservation advice, or bus route information even in offline mode.

Enhancements in **language support** will also be critical. By training IBM Granite LLM with regional datasets, the assistant can better handle mixed-language queries. For example, in a city like Mumbai, a citizen might ask: “Next train Churchgate la epdi poganum?” (a mix of English, Hindi, and Tamil). A more context-aware system would recognize this and respond with accurate train timings and directions.