# SUSTAINABLE SMART CITY ASSISTANT USING IBM GRANITE LLM

## 1.Introduction


- Team Member : Padma Priya S
- Team Member : Prabhavathi K
- Team Member : Priyanka P
- Team Member : Rafia Begum N


## 2.Project Overview

### Purpose :

The purpose of a sustainable smart city assistant using IBM Granite LLM is to support cities in achieving their sustainability goals through intelligent, data-driven, and citizen-centric solutions. By leveraging the advanced natural language understanding and reasoning capabilities of IBM Granite, the assistant can help manage urban resources more efficiently, reduce environmental impact, and enhance the quality of life for residents. It enables real-time interaction with citizens, providing personalized insights on energy usage, waste management, transportation, and policy updates. Additionally, the assistant can assist city officials in analyzing complex datasets, generating reports, and making informed decisions to drive sustainability initiatives. Integrated with IoT and city infrastructure, it also interprets live sensor data to optimize operations and respond proactively to urban challenges. Built on IBM's enterprise-grade AI platform, this solution ensures security, scalability, and ethical AI governance, making it a powerful tool for building smarter, greener, and more resilient cities.

## Conversational Interface

- Key Point: Natural language interaction
- Simplified policy understanding

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

## Resource Forecasting

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

## Policy Summarization

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

## 1. AI Assistant Layer (IBM Granite LLM Integration)

This is the core intelligence layer where the **IBM Granite LLM** processes user inputs (in natural language) and generates context-aware responses.

- **Key Functions**:
  - Natural language understanding (NLU) for user queries.
  - Answer generation using city-specific data and sustainability goals.
  - Multilingual and domain-specific support for urban topics (e.g., energy, waste, transport).
- **Tools**:
  - IBM Granite LLM via **watsonx.ai**
  - RAG (Retrieval-Augmented Generation) for incorporating live city data
  - Prompt engineering for sustainability and smart city contexts

## 2. Data Integration Layer

This layer connects various smart city data sources, both **real-time** and **historical**, to provide context for the assistant's responses.

- **Data Sources**:
    - IoT devices (air quality sensors, traffic monitors, smart meters)
    - City open data portals (transport schedules, weather, utilities)
    - GIS systems and urban planning datasets
- **Purpose**:
    - Supply relevant data for analysis, forecasting, and personalized recommendations.
    - Enable intelligent decision-making for citizens and officials.

## 4. Setup Instructions

**Prerequisites:**

o 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

## 8. Authentication

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

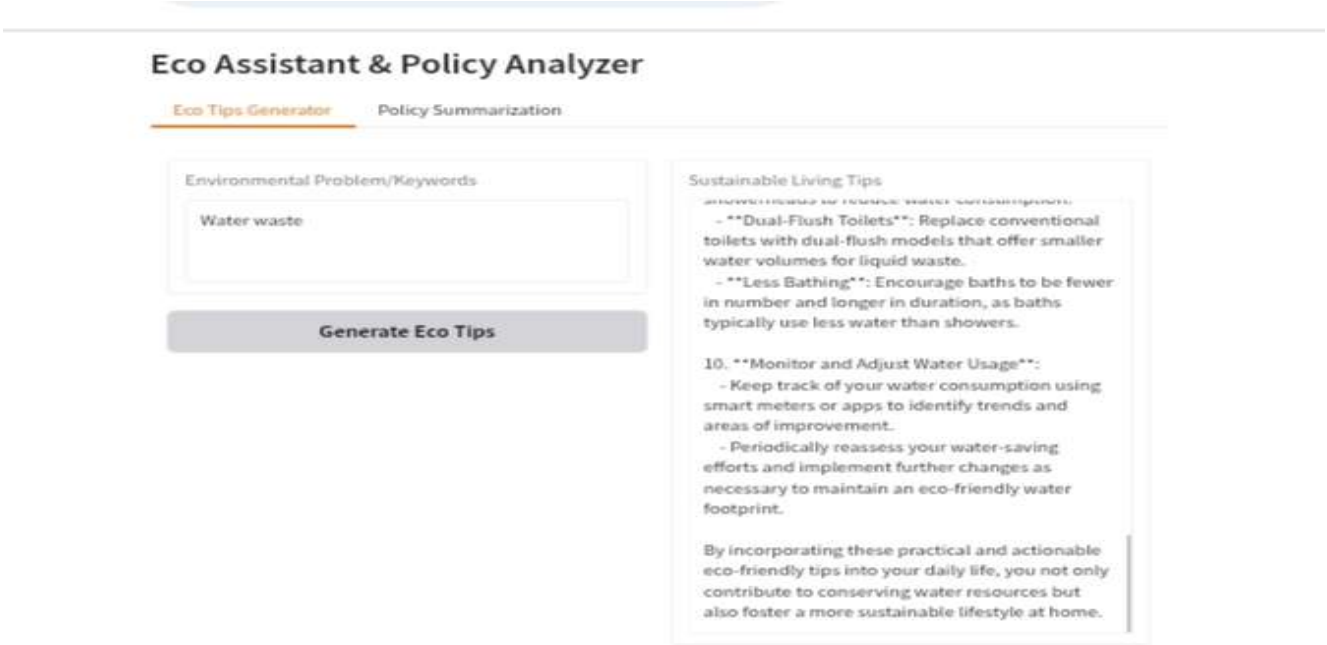
**Testing was done in multiple phases:**

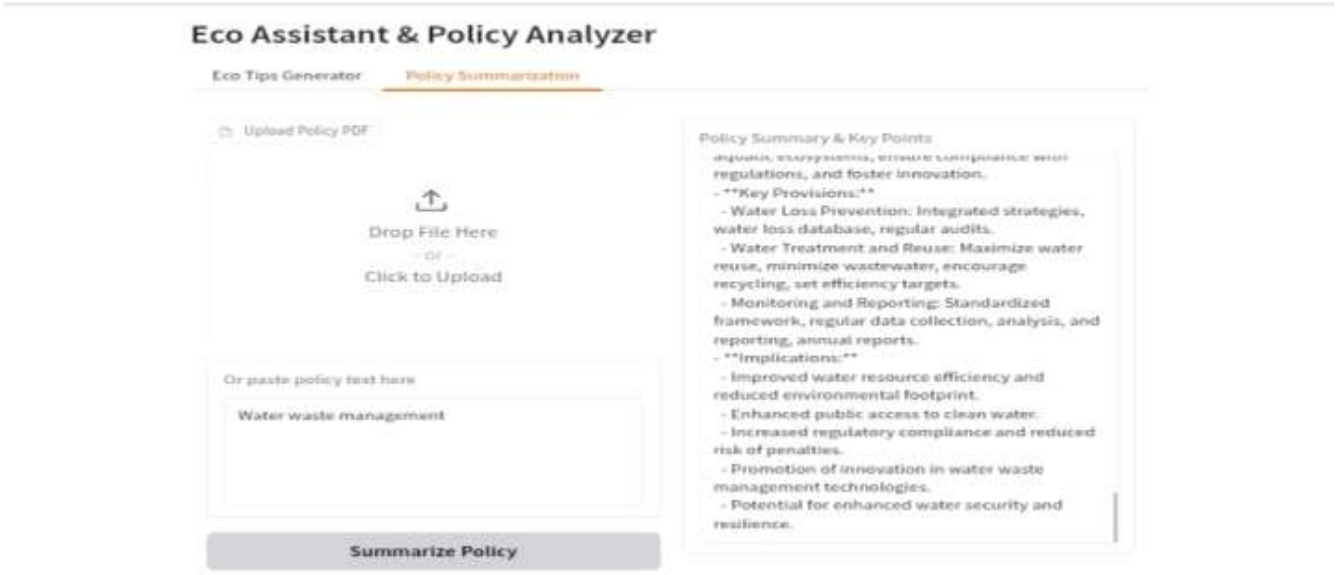
Unit Testing: For prompt engineering functions and utility scripts API Testing: Via Swagger UI, Postman, and test scripts

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.Screenshorts






## 12. Known Issues

- **Hallucinations and Factual Inaccuracy**
    - The model may generate false or misleading information, even when it sounds confident.
    - This can be dangerous in areas like environmental advice, city policy, or emergency communication.

- **High Sensitivity to Prompt Phrasing**
    - Output quality varies significantly depending on how questions or prompts are worded.
    - Makes it difficult to ensure consistent performance across diverse user inputs (e.g., citizens vs. officials).

## 13. Future enhancement

- **Integration with Real-Time IoT Data**

Connect the assistant to live city sensors (e.g., air quality, traffic flow, energy meters) to provide accurate, real-time recommendations.

- **Fine-Tuning with Localized Sustainability Data**

Train the model using city-specific data like municipal policies, climate reports, and urban planning documents to improve contextual understanding and relevance.

- **Multilingual and Inclusive Communication**

Enable support for multiple languages and regional dialects to make the assistant accessible to a broader range of citizens.