# Sustainable Smart City Assistant Using IBM Granite LLM

## Project Documentation

Team ID: LTVIP2025TMID32000

Team Size: 4

Team Leader: K Sai Charan

Team Member: Inkollu Sai Bhanu Koushik

Team Member: Gunisetti Rakesh

Team Member: Gousia Begum

### 1. Introduction

The Sustainable Smart City Assistant is an AI-powered application designed to support urban sustainability, governance, and citizen engagement. Built using Python, Streamlit, and integrated with IBM Watsonx's Granite LLM, it provides AI-driven policy summarization, eco-advice, KPI forecasting, anomaly detection, and citizen support through natural language interaction.

### 2. Project Overview

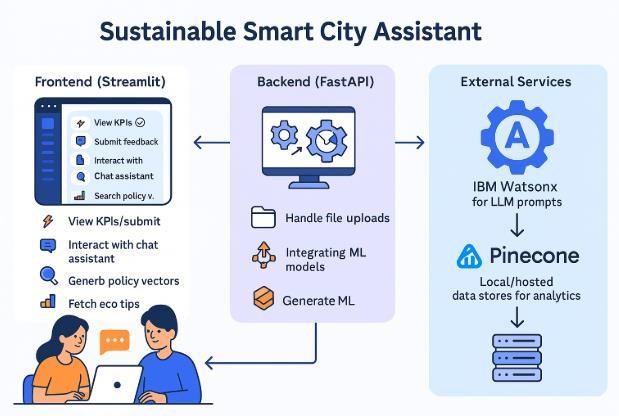
The purpose of the Sustainable Smart City Assistant is to minimize manual effort in city operations, improve decision-making, and promote sustainable living through AI-powered support for citizens and city administrators.

**Key Features:**

* Policy Document Upload & Summarization
* Eco Tips Generator
* KPI Forecasting Module
* Anomaly Detection for Resource Usage
* AI-Powered Chat Assistant
* Citizen Feedback Reporting

Each feature integrates IBM's Granite LLM to interpret user input and provide AI-generated insights.

### 3. Architecture



The assistant follows a clean, modular architecture:

* **User Interface (Streamlit):** Collects inputs, displays results, manages interactions.
* **Application Logic (Python):** Processes user requests, forms AI prompts, handles session state.
* **AI Service Layer (IBM Watsonx):** Granite LLM generates contextual outputs for summaries, eco-tips, reports, and chat.
* **Document Processor:** (If implemented) for extracting content from uploaded policy documents.
* **Data Storage:** Streamlit session state for temporary user interactions; optional local file storage for static data.
* **Security Layer:** Environment variables for secure API key handling for Watsonx services.

### 4. Setup Instructions

**Prerequisites:**

* Python 3.8 or above
* IBM Cloud account with Watsonx access
* Streamlit, ibm-watsonx-ai, langchain, python-dotenv

**Installation Steps:**

1. Clone the project and navigate to the folder.
2. Create a virtual environment: python -m venv venv
3. Activate the virtual environment:
   * Windows: .\venv\Scripts\activate
   * macOS/Linux: source venv/bin/activate
4. Install dependencies: pip install -r requirements.txt
5. Create a .env file in the root directory with the following variables (replace placeholders with your actual credentials):
6. WATSONX\_API\_KEY="your\_key"
7. PROJECT\_ID="your\_project\_id"
8. WATSONX\_URL="https://your-region.ml.cloud.ibm.com"
9. WATSONX\_MODEL\_ID="ibm/granite-13b-instruct-v2"
10. Run the app: streamlit run app.py

### 5. API Documentation

The assistant uses IBM Watsonx's generate\_text() via the ibm-watsonx-ai Python SDK.

**Example Prompt Usage:**

* **Policy Summarization:** "Summarize this policy for citizens."
* **Eco Tips:** "Generate eco-friendly tips for reducing plastic waste."
* **Chat Queries:** "How can my city improve air quality?"

**AI Parameters:**

* max\_new\_tokens = 500
* temperature = 0.7
* top\_p = 1.0
* decoding\_method = "sample" (or "greedy" as seen in image)

### 6. Authentication

IBM Watsonx APIs are accessed securely using API keys stored in a .env file, loaded via python-dotenv.

**Security Practices:**

* Never hardcode API keys.
* Use .env and .gitignore to prevent accidental exposure.
* Load variables at runtime only.

### 7. User Interface

The app features a sidebar for navigation between modules. Inputs and results are displayed using st.file\_uploader, st.text\_area, st.chat\_input, and dynamic cards.

**Modules:**

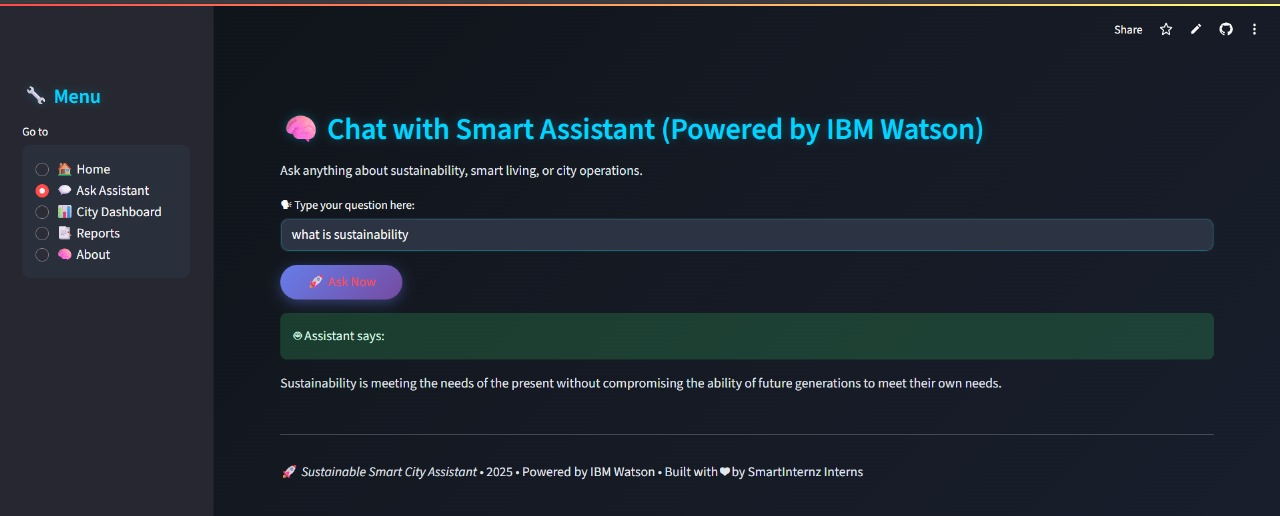
* **Policy Document Upload** → file uploader
* **Eco Tips Generator** → topic input
* **KPI Forecasting** → file uploader
* **Anomaly Detection** → dataset input
* **Chat Assistant** → natural language chat
* **Citizen Feedback** → form for issue reporting

### 8. Testing

The application includes multiple testing mechanisms:

* **Unit Testing:** AI prompt creation, output cleaning functions.
* **Integration Testing:** Streamlit interaction with Watsonx.
* **Manual Testing:** Input validation, output quality, anomaly detection, and chatbot responses.

### 9. Screenshots



Screenshots of the dashboard, modules, and AI-generated outputs to be added after deployment.

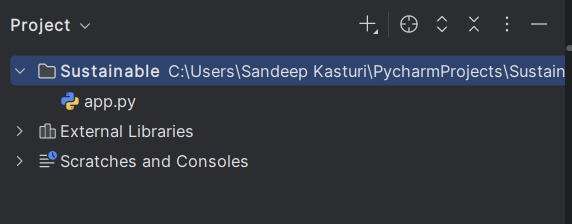
### 10. Known Issues

* No persistent user login system.
* No database for long-term storage (session-based only).
* Limited role-based access control.
* IBM Watsonx API rate limits based on subscription.

### 11. Future Enhancements

* Add persistent database (MongoDB, PostgreSQL).
* Dockerize for scalable deployment.
* Role-based login and access control.
* City-wise sustainability report generator.
* Support for audio-based prompts and mobile responsiveness.

### Folder Structure



Sustainable/

├── app.py → Main Streamlit application file

├── .env → Environment variables for API keys

├── requirements.txt → Project dependencies

├── venv/ → Virtual environment (created by `python -m venv venv`)

### 13. Modules Breakdown

Each module calls a function (e.g., ask\_granite(prompt)) to interact with the AI model.

* **Policy Summarizer** → Document-to-summary
* **Eco Tips Generator** → Topic-to-actionable tips
* **KPI Forecasting** → File-to-resource prediction
* **Anomaly Detection** → Dataset-to-alerts
* **Chat Assistant** → Natural language Q&A
* **Citizen Feedback** → Issue reporting

### 14. Technology Stack

* **Frontend:** Streamlit
* **Backend:** Python
* **AI Model:** IBM Watsonx Granite LLM
* **Authentication:** python-dotenv + .env
* **Deployment Target:** IBM Cloud Foundry / Localhost

### 15. Conclusion

The Sustainable Smart City Assistant demonstrates how AI can enhance urban sustainability, citizen engagement, and governance by providing automated insights, resource management tools, and citizen-friendly interaction. Future versions will extend its capabilities with persistent storage, mobile support, and deeper smart city integrations.