Project Title:

Sustainable Smart City Assistant Using IBM Granite LLM

Team members:

Atchaya.S

Atchaya Priyanka.S

Gobika.M

Harini.V

1. Introduction

Purpose:

To empower cities and residents to thrive in an eco-conscious and connected environment.

Optimizes resources (energy, water, waste)

Provides sustainable lifestyle guidance

Helps officials with insights, forecasting, and policy summarization

Features:

Conversational Interface (chat in natural language)

Policy Summarization (simplified understanding)

Resource Forecasting (predictive analytics)

Eco-Tip Generator (personalized advice)

Citizen Feedback Loop (community engagement)KPI Forecasting (strategic planning)

Anomaly Detection (early warnings)

Multimodal Input (text, PDF, CSVs)

Streamlit/Gradio UI (user-friendly dashboard

2. Architecture:

Frontend: Streamlit (dashboards, chat, file uploads, reports, feedback forms)

Backend: FastAPI (document processing, chat, tips, reports, embeddings)

LLM: IBM Watsonx Granite (summarization, insights, sustainability tips)

Vector Search: Pinecone (semantic search on policy docs)

ML Modules: Forecasting & Anomaly Detection (Scikit-learn, pandas, matplotlib)

3. Setup Instructions:

Requirements: Python 3.9+, pip, API keys (IBM Watsonx, Pinecone)

Steps: Clone repo → install requirements → configure .env → run FastAPI → launch Streamlit UI

4. Folder Structure:

app/ – backend logic (FastAPI routers, models)

app/api/ – modular APIs (chat, feedback, vectorization)

ui/ – Streamlit UI components

smart_dashboard.py – main UI entry point

granite_llm.py – Watsonx Granite integration

document_embedder.py – document embeddings + Pinecone

kpi_file_forecaster.py – forecasting models

anomaly_file_checker.py – anomaly detection

report_generator.py – sustainability reports

5. Running the Application:

1. Start FastAPI backend

2. Launch Streamlit dashboard

3. Upload docs/CSVs & interact with chat, forecasting, tips, reports

4. Real-time interaction through APIs

6. APIs:

POST /chat/ask → AI-generated response

POST /upload-doc → embed docs

GET /search-docs → semantic search results

GET /get-eco-tips → sustainability tips

POST /submit-feedback → store citizen feedback

7. Authentication:

Demo mode: open environment

Secure options: JWT tokens, OAuth2, Role-based access, IBM Cloud integration

8. User Interface:

Sidebar navigation

KPI visualizations & summary cards

Tabs for chat, eco tips, forecasting

Real-time forms & PDF downloads

Minimalist, accessible, fast

9. Testing

Unit Testing → prompt engineering, utilities

API Testing → Swagger, Postman, scripts

Manual Testing → uploads, responses, outputs

Edge Cases → large files, malformed inputs, invalid keys

10. Future Enhancements:

Role-based dashboards (citizen, official, researcher)

User session history tracking

Integration with IoT data streams

Advanced ML for sustainability insight

Screenshot: