📌 Project Title

Sustainable Smart City Assistants

Team leader : Srinidhi. M

Team mates : Pavithra K

Team mates : Ramya V

Team mates : Sridevi G

Team mates: Kaviyasri M

📝 1. Introduction

Purpose:

To help cities and residents become more eco-conscious and connected using AI + real-time data.

Optimizes energy, water, and waste.

Provides personalized eco tips.

Assists officials with insights, forecasts, and policy summaries.

Encourages community engagement.

Features:

Conversational interface (citizen/official Q&A).

Policy summarization (simplifies government docs).

Resource forecasting (energy, water, waste).

Eco-tip generator (personalized advice).

Citizen feedback loop (public input → city planning).

KPI forecasting (strategic planning).

Anomaly detection (sensor data issues).

Multimodal input (text, PDFs, CSVs).

Streamlit/Gradio UI.

2. Architecture

Frontend (Streamlit) → Interactive dashboards, chat, feedback forms, modular pages.

Backend (FastAPI) → APIs for docs, chat, eco-tips, reports, embeddings.

LLM (IBM Watsonx Granite) → Summaries, sustainability tips, natural language interaction.

Vector DB (Pinecone) → Semantic search on documents.

ML Modules → Forecasting & anomaly detection with Scikit-learn.

3. Setup Instructions

Prerequisites:

Python 3.9+

pip + venv

IBM Watsonx & Pinecone API keys

Internet access

Steps:

1. Clone repo

2. Install dependencies (requirements.txt)

3. Configure .env with credentials

4. Run FastAPI backend

5. Launch Streamlit frontend

6. Upload data & interact

📂 4. Folder Structure

app/ → Backend logic

app/api/ → API routes

ui/ → Streamlit UI

smart\_dashboard.py → Main dashboard

granite\_llm.py → Granite LLM integration

document\_embedder.py → Embeddings + Pinecone

kpi\_file\_forecaster.py → Forecasting

anomaly\_file\_checker.py → Anomaly detection

report\_generator.py → Sustainability reports

🚀 5. Running the Application

1. Launch FastAPI server.

2. Run Streamlit dashboard.

3. Navigate via sidebar → chat, eco-tips, forecasts, uploads.

4. Interact in real-time.

📡 6. API Endpoints

POST /chat/ask → AI-generated responses

POST /upload-doc → Upload & embed docs

GET /search-docs → Semantic search

GET /get-eco-tips → Sustainability advice

POST /submit-feedback → Collect citizen feedback

🔐 7. Authentication

Current demo: open environment.

Future secure deployment:

Token-based auth (JWT/API keys)

OAuth2 with IBM Cloud

Role-based access (admin/citizen/researcher)

User sessions/history

🎨 8. User Interface

Sidebar navigation

KPI visualizations + summary cards

Tabs for chat, eco tips, forecasting

Real-time forms

9. Testing

Unit testing → prompt functions

API testing → Swagger, Postman

Manual testing → file uploads, chats

Edge case handling → malformed inputs, large files, invalid keys