Project Documentation

1. Introduction

Project Title: Sustainable Smart City Assistant using IBM Granite LLM

Team Members:

S. Vinsi Priyanga

T. A. Vinisha

M. Vinodhana

S. Vaijayanthi

2. Project Overview

Purpose:
The Sustainable Smart City Assistant empowers cities and residents to thrive in an eco-conscious and connected environment. By leveraging IBM Granite LLM, AI, and real-time data, it optimizes essential resources like energy, water, and waste while encouraging sustainable citizen behaviors. For city officials, it serves as a decision-making partner by providing insights, forecasts, and simplified summaries of complex policies.

Features:

Conversational Interface: Natural language Q&A for citizens and officials.

Policy Summarization: Converts lengthy policy documents into actionable summaries.

Resource Forecasting: Predicts future energy, water, and waste usage.

Eco-Tip Generator: Provides personalized sustainability advice.

Citizen Feedback Loop: Collects/analyzes feedback for city planning.

KPI Forecasting: Projects key performance indicators for strategic planning.

Anomaly Detection: Detects unusual patterns in sensor/usage data.

Multimodal Input: Handles text, PDFs, and CSVs for analysis.

User-Friendly Dashboard: Built with Streamlit/Gradio for accessibility.

3. Architecture

Frontend (Streamlit): Interactive UI with dashboards, chat, file upload, and reports.

Backend (FastAPI): REST APIs for chat, document processing, and forecasting.

LLM Integration (IBM Watsonx Granite): Generates summaries, tips, and reports.

Vector Search (Pinecone): Enables semantic search in uploaded policy docs.

ML Modules: Forecasting & anomaly detection with Scikit-learn, pandas, and matplotlib.

4. Setup Instructions

Prerequisites:

Python 3.9+

pip and virtual environment tools

API keys (IBM Watsonx & Pinecone)

Internet access

Installation Steps:

1. Clone the repository

2. Install dependencies (requirements.txt)

3. Create .env and configure credentials

4. Run backend with FastAPI

5. Launch frontend with Streamlit

6. Upload data and interact with the modules

5. Folder Structure

app/ – Backend logic (FastAPI routers, models, integrations)
app/api/ – Modular API routes (chat, feedback, reports, docs)
ui/ – Streamlit frontend pages and layouts
smart_dashboard.py – Entry script for dashboard
granite_llm.py – IBM Granite LLM integration
document_embedder.py – Handles embeddings with Pinecone
kpi_file_forecaster.py – Forecasting KPIs
anomaly_file_checker.py – Anomaly detection
report_generator.py – AI-generated reports

6. Running the Application

Start FastAPI backend

Run Streamlit dashboard

Use sidebar navigation

Upload documents/CSVs, interact via chat, and view outputs

7. API Documentation

POST /chat/ask – Responds to user queries

POST /upload-doc – Upload and embed documents

GET /search-docs – Retrieve semantically similar policies

GET /get-eco-tips – Provides sustainability tips

POST /submit-feedback – Stores citizen feedback

8. Authentication

Supports:

JWT or API key tokens

OAuth2 with IBM Cloud credentials

Role-based access (citizens, officials, admins)

9. User Interface

Sidebar navigation

KPI visualization dashboards

Tabs for chat, eco tips, forecasting

Real-time forms

Downloadable reports

10. Testing

Unit Testing: Prompt functions, utilities

API Testing: Swagger UI, Postman

Manual Testing: File uploads, chats, outputs

Edge Cases: Malformed inputs, large files, invalid keys

11. Screenshots

12. Known Issues

Dependency on internet/cloud services

Limited accuracy with very small datasets

13. Future Enhancements

Multi-language support

Mobile app integration

IoT sensor real-time streaming

Advanced predictive analytics with deep learning