

# Sustainable Smart City Assistant using IBM Granite LLM

## 1. Introduction

Project Title: Sustainable Smart City Assistant using IBM Granite LLM

Team Member 1: Sahana R

Team Member 2: \_\_\_\_\_

Team Member 3: \_\_\_\_\_

Team Member 4: \_\_\_\_\_

Team Member 5: \_\_\_\_\_

## 2. Project Overview

**Purpose:** The purpose of this project is to design an AI-powered assistant that helps cities transition into sustainable, intelligent, and citizen-friendly ecosystems. Powered by IBM Granite LLM within the Watsonx platform, the assistant enables cities to manage resources efficiently, engage with communities, and support officials in strategic decision-making.

It integrates AI-driven forecasting, anomaly detection, and policy summarization to reduce environmental impact and enhance governance transparency.

**Features:**

- Conversational Interface: Natural language interaction for citizens and officials.
- Policy Summarization: Simplifies lengthy government policies into clear, actionable insights.
- Resource Forecasting: Predicts future trends in water, energy, and waste usage.
- Eco-Tip Generator: Provides personalized sustainability recommendations.
- Citizen Feedback Loop: Collects, analyzes, and integrates public input.
- KPI Forecasting: Assists city officials with strategic planning and performance monitoring.
- Anomaly Detection: Identifies unusual patterns in urban data for early issue resolution.
- Multimodal Input: Supports text, PDFs, and CSVs for analysis.
- Streamlit/Gradio UI: Intuitive interface for both citizens and officials.

## 3. Architecture

**Frontend (Streamlit/Gradio):** Provides an interactive dashboard with modules for chat, file uploads, reports, and forecasting.

**Backend (FastAPI):** REST API framework handling requests for summarization, forecasting, eco-tips, and anomaly detection.

**LLM Integration (IBM Watsonx Granite):** Granite models enable natural language understanding, summarization, and advice generation.

**Vector Search (Pinecone):** Stores embedded documents and enables semantic search of policies and citizen queries.

**ML Modules (Scikit-learn, Pandas, Matplotlib):** Implements forecasting and anomaly detection using time-series data.

## 4. Setup Instructions

**Prerequisites:**

- Python 3.9+
- pip and virtual environment tools
- IBM Watsonx API key
- Pinecone API key
- Internet connection

Installation Process:

1. Clone repository
2. Install dependencies from requirements.txt
3. Configure .env with credentials
4. Start FastAPI backend
5. Run Streamlit dashboard
6. Upload data and interact with modules

## 5. Folder Structure

app/ – FastAPI backend (routes, models, integration)

app/api/ – Modular API routes (chat, feedback, reports, embeddings)

ui/ – Frontend Streamlit components

smart\_dashboard.py – Main entry for Streamlit dashboard

granite\_llm.py – Handles IBM Granite LLM communication

document\_embedder.py – Document embedding & Pinecone storage

kpi\_file\_forecaster.py – Forecasting trends in urban data

anomaly\_file\_checker.py – Detects anomalies in datasets

report\_generator.py – AI-generated sustainability reports

## 6. Running the Application

- Launch FastAPI server
- Start Streamlit UI
- Navigate via sidebar to access dashboards, chat, eco-tips, forecasting, and reports
- Upload files (PDF/CSV) and interact with the AI assistant
- Get real-time summaries, insights, and predictions

## 7. API Documentation

POST /chat/ask – User queries, AI-generated responses

POST /upload-doc – Uploads and embeds documents in Pinecone

GET /search-docs – Returns similar policies using semantic search

GET /get-eco-tips – Provides sustainability suggestions

POST /submit-feedback – Stores citizen feedback

## 8. Authentication

- Token-based authentication (JWT/API Keys)
- IBM Cloud OAuth2 credentials
- Role-based access (citizen, official, admin)
- Planned: User sessions & history tracking

## **9. User Interface**

- Sidebar navigation
- KPI dashboards with visual summaries
- Tabbed layouts (Chat, Eco Tips, Forecasting, Reports)
- Real-time feedback forms
- Exportable PDF reports

## **10. Testing**

- Unit Testing – For prompt functions & utilities
- API Testing – Swagger UI/Postman
- Manual Testing – File uploads, chat outputs, report accuracy
- Edge Case Handling – Large files, malformed input, invalid keys

## **11. Screenshots**

(To be added after implementation – dashboards, chat, forecasting graphs, eco-tips, reports)

## **12. Known Issues**

- Requires stable internet for IBM Watsonx API calls
- Large file uploads may increase processing time
- Limited offline capability

## **13. Future Enhancements**

- Multilingual support
- Mobile app version
- Integration with IoT smart city sensors
- Real-time traffic & pollution monitoring
- AI-driven citizen engagement dashboards