**Sustainable Smart City Assistant**

**Project Documentation**

# 1. Introduction

Project title: Sustainable Smart City Assistant

* Team member: Shivani Pathak R
* Team member: Bhavana VP
* Team member: Dhanuja Shankar Dakshayani
* Team member: Hasmathunnisa. S

**2. Project Overview** Purpose:

The purpose of a Sustainable Smart City Assistant is to empower cities and residents to thrive in a more eco-conscious and connected environment.

It helps optimize resources (energy, water, waste) and guides sustainable behaviors via tips and insights.

For officials, it provides forecasting, summaries, and decision support to create efficient, inclusive, and resilient cities.

Features:

* Conversational Interface (Natural language Q&A)
* Policy Summarization (Concise summaries)
* 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, CSV)
* Streamlit/Gradio UI (User-friendly interface)

# 3. Architecture

Frontend: Streamlit dashboards, file uploads, chat, feedback, reports

Backend: FastAPI powering APIs for docs, chat, tips, reports

LLM: IBM Watsonx Granite for summaries & eco-tips

Vector Search: Pinecone for embeddings & semantic search

ML: Forecasting & anomaly detection with Scikit-learn

# 4. Setup Instructions

Prerequisites: Python 3.9+, pip, API keys for IBM Watsonx & Pinecone

Steps: Clone repo → Install dependencies → Configure .env → Run FastAPI → Launch

Streamlit → Upload data & interact

**5. Folder Structure** app/ – Backend logic app/api/ – API routes ui/ – Streamlit components smart\_dashboard.py – Dashboard entry granite\_llm.py – LLM handler document\_embedder.py – Embeddings kpi\_file\_forecaster.py – Forecasting anomaly\_file\_checker.py – Anomaly detection report\_generator.py – AI reports

# 6. Running the Application

Start FastAPI → Run Streamlit → Navigate via sidebar → Upload docs/CSVs → Chat & view outputs (reports, summaries, predictions)

# 7. API Documentation

POST /chat/ask – AI response

POST /upload-doc – Upload docs

GET /search-docs – Semantic search

GET /get-eco-tips – Eco-tips

POST /submit-feedback – Store feedback

**8. Authentication**

Demo runs open. Future: JWT, OAuth2, role-based access, sessions & history tracking

# 9. User Interface

Sidebar navigation, KPI visualizations, tabs (chat, eco tips, forecasting), real-time forms, PDF reports

# 10. Testing

Unit tests (functions), API tests (Swagger, Postman), Manual tests (uploads, chat), Edge cases (large/malformed inputs)

# 12. Known Issues

* Limited scalability with large datasets
* Needs internet for APIs
* Forecast accuracy depends on data
* Minimal authentication
* UI slowdown with big files
* Basic error handling

# 13. Future Enhancements

* Stronger security
* Mobile support
* Multilingual features
* IoT integration
* Smarter forecasting
* Personalized tips
* More file formats
* Better visuals & auto-reports