

Sustainable Smart City Assistant Using Granite LLM

Project Documentation

1. Introduction

- Project Title: Sustainable Smart City Assistant
- Team Members: Rahman Khan .A
- Team Members: Sanjai.S
- Team Members: Prajin
- Team Members: Sanjairam

2. Project Overview

Purpose

The Sustainable Smart City Assistant empowers cities and citizens to thrive in an eco-conscious and connected urban environment. By leveraging IBM Watsonx Granite LLMs, Pinecone vector database, and real-time data analytics, it optimizes energy, water, and waste usage while providing actionable sustainability insights.

For citizens, it delivers policy summaries, eco-tips, and chatbot support.

For officials, it provides decision-making insights, anomaly detection, KPI forecasting, and sustainability reports.

Features

- **Conversational Chat Assistant** – Natural language Q&A powered by IBM Watsonx Granite.
- **Policy Summarization** – Converts lengthy government documents into concise, actionable summaries.
- **Eco Tip Generator** – Provides personalized, practical sustainability advice.
- **Citizen Feedback Loop** – Collects and analyzes public feedback to guide planning.
- **KPI Forecasting** – Predicts trends in air quality, energy, and green cover.

- **Anomaly Detection** – Flags unusual values in KPI/sensor data.
- **Document & Data Search** – Uses Pinecone vector embeddings for semantic retrieval.
- **City Report Generator** – Produces structured sustainability reports.
- **Multimodal Input** – Accepts PDFs, CSVs, and text files.
- **Interactive Dashboard** – Built with Streamlit for a user-friendly interface.

3. Architecture

Frontend (Streamlit)

- Provides navigation sidebar with modules: **Chat, Policy Summarizer, Eco Tips, KPI & Report Dashboard**.
- Visualizes KPIs with metrics, anomaly detection flags, and bar charts.

Backend (FastAPI)

- Exposes REST API endpoints for summarization, chat, eco tips, feedback, reports, and vector search.
- Fully documented via **Swagger UI** (/docs).

LLM Integration (IBM Watsonx Granite)

- Uses **Granite family of LLMs** via IBM Cloud API key authentication.
- Handles summarization, report generation, eco-tips, and chat.

Vector Search (Pinecone)

- Sentence-transformer embeddings stored in Pinecone.
- Semantic search enables users to query policies and city data in natural language.

ML Modules

- **Forecasting:** Uses scikit-learn regression to project KPIs.
- **Anomaly Detection:** Identifies unusual KPI values with threshold and statistical methods.

4. Setup Instructions

Prerequisites

- Python 3.9+
- IBM Watsonx API key, Project ID, Model ID
- Pinecone API key
- Virtual environment & pip

Installation

```
git clone <your-repo-url>
cd smartcity-assistant-starter
python -m venv .venv
.\.venv\Scripts\activate # Windows
pip install -r requirements.txt
```

Environment Variables (.env)

```
WATSONX_API_KEY=pAjcxj3Df0g687V0mCe3_Q8TmRKSDlp9wulxo52qwNn5
WATSONX_PROJECT_ID=f371addd-61dd-4ff0-882d-571db5a32aea
WATSONX_URL=https://eu-de.ml.cloud.ibm.com
WATSONX_MODEL_ID=ibm/granite-13b-instruct-v2
PINECONE_API_KEY=pcsk_22YGB3_9RY8BMqaUZN55nkxUA7nR7ZyhBnKA1LjW
44XtRpkeo43rcmj2yw4HrPhQfQbafu
PINECONE_ENV=us-east-1
INDEX_NAME=smartcity-policies
```

Running

Backend:

```
uvicorn app.main:app --reload
```

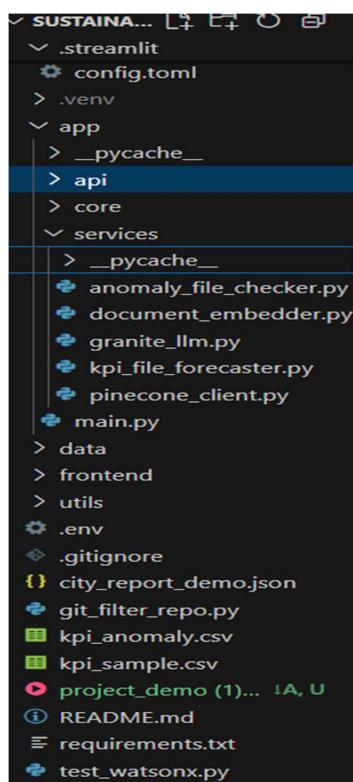
Frontend:

```
streamlit run frontend/smart_dashboard.py
```

5. Folder Structure

app/

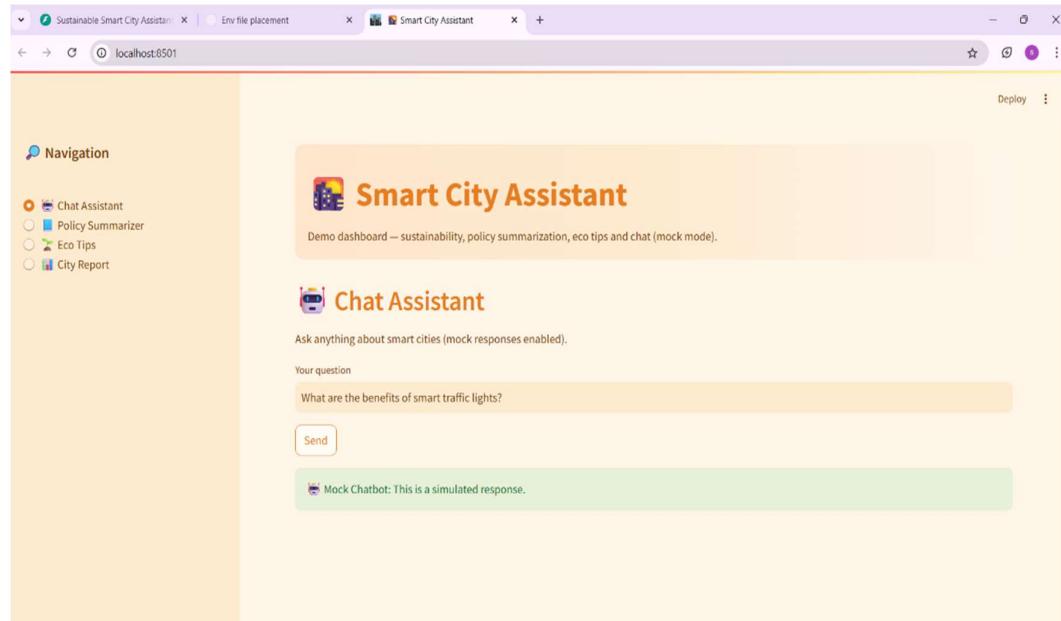
```
|—— api/      # FastAPI routers  
|—— core/     # Config & Pinecone client  
|—— services/ # LLM, embeddings, forecasting, anomaly detection  
  
frontend/  
    └── smart_dashboard.py # Streamlit frontend
```



6. Running the Application

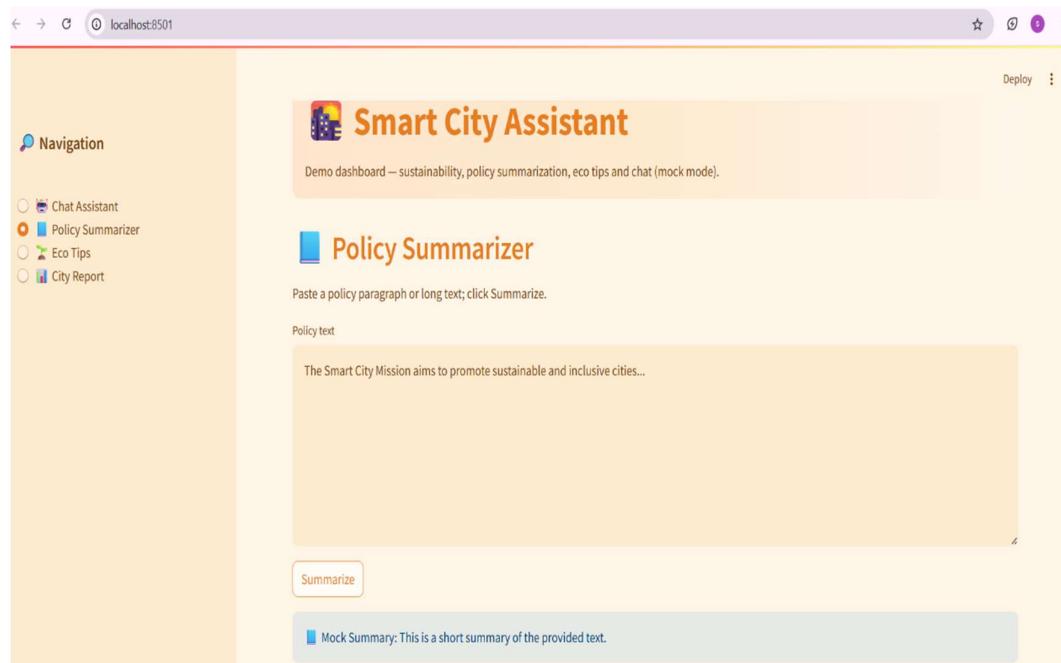
1. Start FastAPI backend → <http://127.0.0.1:8000/docs>
2. Start Streamlit frontend → <http://localhost:8501>
3. Navigate through:

-  Chat Assistant



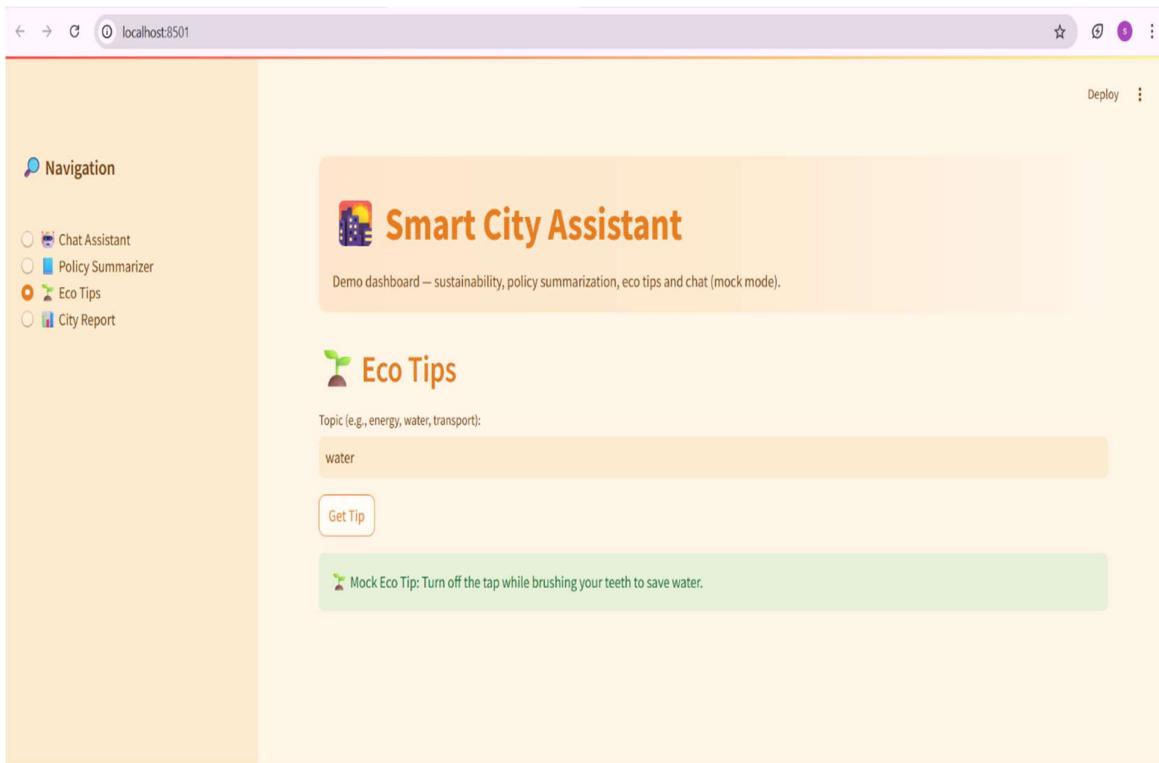
A screenshot of a web browser window titled "Smart City Assistant" at "localhost:8501". The page features a navigation sidebar on the left with icons for Chat Assistant, Policy Summarizer, Eco Tips, and City Report. The main content area has a yellow header bar with the title "Smart City Assistant" and a subtitle "Demo dashboard — sustainability, policy summarization, eco tips and chat (mock mode)". Below this is a section titled "Chat Assistant" with the sub-instruction "Ask anything about smart cities (mock responses enabled)". A text input field contains the question "What are the benefits of smart traffic lights?", and a "Send" button is below it. A green message box displays the response "Mock Chatbot: This is a simulated response.".

-  Policy Summarizer



A screenshot of a web browser window titled "Smart City Assistant" at "localhost:8501". The page features a navigation sidebar on the left with icons for Chat Assistant, Policy Summarizer, Eco Tips, and City Report. The main content area has a yellow header bar with the title "Smart City Assistant" and a subtitle "Demo dashboard — sustainability, policy summarization, eco tips and chat (mock mode)". Below this is a section titled "Policy Summarizer" with the sub-instruction "Paste a policy paragraph or long text; click Summarize". A text input field contains the text "The Smart City Mission aims to promote sustainable and inclusive cities...", and a "Summarize" button is below it. A blue message box displays the response "Mock Summary: This is a short summary of the provided text.".

○ Eco Tips



The screenshot shows a web browser window for 'localhost:8501'. The title bar says 'localhost:8501'. The main content area has a light orange background. On the left, there's a vertical navigation sidebar with a yellow background, titled 'Navigation' and containing links: Chat Assistant, Policy Summarizer, Eco Tips (which is selected and highlighted in orange), and City Report.

The main content area features a large orange header with the text 'Smart City Assistant' and a small icon of a building with a gear. Below it is a sub-header: 'Demo dashboard — sustainability, policy summarization, eco tips and chat (mock mode.)'. The main content is divided into sections:

- Eco Tips**: A section with a green leaf icon. It asks for a 'Topic (e.g., energy, water, transport):' and has a text input field with 'water'. A button labeled 'Get Tip' is below it. A green box contains a 'Mock Eco Tip: Turn off the tap while brushing your teeth to save water.'

○ KPI Dashboard & City Report



○ Anomaly Checker

7. API Documentation

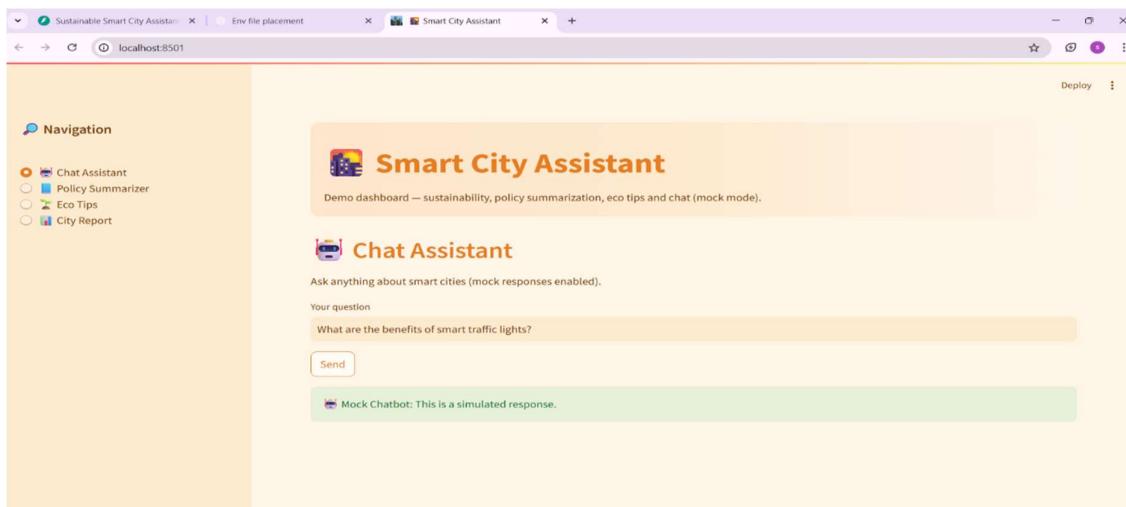
- POST /chat/ask → Smart city Q&A
- POST /policy/summarize → Summarize policy text
- GET /eco/tips?topic= → Eco tips
- POST /report/ → Generate city sustainability report
- POST /vector/upsert → Upload & embed documents
- GET /vector/search?query= → Search uploaded documents
- POST /feedback/ → Store citizen feedback

8. Authentication

- All IBM Watsonx endpoints require **IAM token (fetched via API key)**.
- Pinecone requires API key & environment ID.
- Can extend with JWT/OAuth2 for secure production deployment.

9. User Interface

- Clean **Streamlit dashboard** with sidebar navigation.
- KPI visualizations with cards & bar charts.
- Tabbed layouts for summaries, eco tips, anomaly detection, and forecasting.



10. Testing

- **Unit Tests** for LLM prompt functions.
- **API Tests** via Swagger & Postman.
- **Manual Testing** for frontend flows.
- **Edge Cases** → invalid inputs, large files, missing credentials.

The screenshot shows the API documentation for the Sustainable Smart City Assistant version 0.1.0, generated by OpenAPI 3.1. The interface includes a sidebar with sections like chat, eco, feedback, policy, vector, and error types. The main area displays various API endpoints with their methods, paths, and descriptions.

chat

- POST /chat/task** Ask

eco

- GET /eco/tips** Eco Tips

feedback

- POST /feedback/submit** Submit

policy

- POST /policy/summarize** Summarize

vector

- POST /vector/upsert** Upsert
- POST /vector/query** Query

Error Types

- ChatReq > Expand all object
- Chunk > Expand all object
- Feedback > Expand all object
- HTTPValidationError > Expand all object
- PolicyDoc > Expand all object
- QueryReq > Expand all object
- ReportReq > Expand all object
- ValidationError > Expand all object

The screenshot shows the report API endpoint in the Postman interface. It's a POST request to /report/generate with the "Generate" operation. The parameters section shows "No parameters". The request body is required and contains a JSON schema for a city report. The schema defines a "city" field with a "name" and a "pollution" field containing "PM2.5", "CO2", "Energy Use", and "Water Usage".

report

POST /report/generate Generate

Parameters

No parameters

Request body **required**

Edit Value | Schema

```
{  
  "city": "Demo City",  
  "pollution": {  
    "PM2.5": 75,  
    "CO2": 1200,  
    "Energy Use": 500  
  }  
}
```

Cancel Reset application/json

Execute Clear

11. Future Enhancements

- Role-based access control (Admin, Citizen, Researcher).
- Real-time IoT sensor integration for KPIs.
- Geo-mapping dashboard with sustainability layers.
- Multilingual support with IBM Watson Language Translator.