

# Sustainable Smart City Assistant Using IBM Granite LLM

## Project Documentation

- Introduction

- Project title :Sustainable Smart City Assistant Using IBM

- Granite LLB

- Team member : S. ARTHI
- Team member : R.PRIYADHARSHINI
- Team member : K.NANDHINI
- Team member : K.PRADEEPA
- Team member :F.SHINY SANDHANA MARY

## Project overview

Purpose :

A Sustainable Smart City aims to create a livable, efficient, and environmentally friendly urban environment by leveraging technology, innovation, and sustainable practices. The purpose is to:

1. Improve Quality of Life: Enhance the well-being and quality of life for citizens through better services, infrastructure, and amenities.
2. Reduce Environmental Impact: Minimize waste, pollution, and carbon footprint through sustainable practices, renewable energy, and efficient resource management.
3. Promote Economic Growth: Foster entrepreneurship, and economic growth through smart infrastructure, digitalization, and data-driven decision-making.
4. Enhance Sustainability, Governance, and Citizen Engagement: Promote transparency, accountability, and citizen engagement through digital governance, participatory planning, and data-driven policy-making.

## Key Features

### 1. Renewable Energy:

Utilizes solar, wind, and other renewable sources to power homes and businesses

### 2. Efficient Transportation:

Implements smart traffic management and promotes public transport, cycling, and walking

### 3. Smart Infrastructure:

Uses IoT sensors and data analytics to optimize resource usage and improve services

### 4. Green Spaces:

Incorporates parks and green roofs to improve air quality and biodiversity

### 5. Waste Management:

Implements efficient waste collection and recycling systems

### 6. Citizen Engagement:

Encourages participation through digital platforms and community initiatives

## Benefits

### 1. Reduced Carbon Footprint:

Lower greenhouse gas emissions and energy consumption

### 2. Improved Quality of Life:

Enhanced public services, safety, and livability

### 3. Economic Growth:

Attracts businesses and investments through innovation and sustainability

## Examples

### 1. Barcelona's Smart City Initiatives:

Focus on smart lighting, waste management, and public transportation

### 2. Singapore's Smart Nation Program:

Integrates technology to improve healthcare, transportation, and public services

## Architecture

The architecture consists of three main layers:

### 1. Frontend (Streamlit)

- Provides a user-friendly interface with various features, including:

- Viewing KPIs (Key Performance Indicators)
- Submitting feedback
- Interacting with a chat assistant
- Searching policy vectors
- Generating reports
- Fetching eco-tips

## 2.Backend (FastAPI)

- Manages API requests and handles file uploads
- Integrates with:
  - Machine Learning (ML) models
  - Pinecone (vector indexing and semantic search)
  - Watsonx Granite LLM (Large Language Model)

## 3.External Services

- IBM Watsonx: Provides LLM prompts and capabilities
- Pinecone: Offers vector indexing and semantic search functionality
- Local/Hosted Data Stores: Stores data for analytics and reporting

## Potential Applications

1.Sustainability Analytics: Analyzing data to provide insights on sustainability and environmental impact.

2.Policy Analysis: Using ML models and semantic search to analyze and understand policy documents.

Eco-Friendly Recommendations: Providing eco-tips and recommendations to users based on data analysis and ML models.

- SetupInstructions

### Prerequisites:

- Python 3.9 or later
- pip and virtual environment tools
- API keys for IBM Watsonx and Pinecone
- Internet access to access cloud services

### Installation Process:

- Clone the repository
- Install dependencies from requirements.txt
- Create a .env file and configure credentials
- Run the backend server using Fast API
- Launch the frontend via Stream lit
- Upload data and interact with the modules

### • Folder Structure

app/ – Contains all Fast API backend logic including routers, models, and integration modules.

app/api/ – Subdirectory for modular API routes like chat, feedback, report, and document vectorization.

ui/ – Contains frontend components for Stream lit pages, card layouts, and form UIs.

smart\_dashboard.py – Entry script for launching the main Stream lit dashboard.

granite\_llm.py – Handles all communication with IBM Watsonx Granite model including summarization and chat.

document\_embedder.py – Converts documents to embeddings and stores in Pinecone.

kpi\_file\_forecaster.py – Forecasts future energy/water trends using regression.

anomaly\_file\_checker.py – Flags unusual values in uploaded KPI data.

report\_generator.py – Constructs AI-generated sustainability reports.

Required libraries:

streamlit: For building interactive dashboard interfaces

fastapi: Backend API framework for rapid development

uvicorn: ASGI server to run FastAPI

requests: For API communication from frontend

python-dotenv: Manage environment variables

sentence-transformers: Text embedding model

pydantic-settings: Handle configuration management

pinecone-client: For semantic document search

scikit-learn, pandas: For anomaly detection and forecasting

matplotlib: For report visualizations

## Project Milestones & Development Flow

### Phase 1 – Project Initialization

Modular Folder Structure Defined: Created separate folders for app/api, services, vectorstore, core, frontend/components, and utils for organized and scalable development.

### Environment Setup:

.env file created with keys for Pinecone and Watsonx. config.py loads environment variables securely using pydantic.

### .env file

```
1 WATSONX_API_KEY=pAjcxi3Df0g687V0mCe3_Q8TmRKSDlp9wulXo52qwNn5
2 WATSONX_PROJECT_ID=f371addd-61dd-4ff0-882d-571db5a32aea
3 WATSONX_URL=https://eu-de.ml.cloud.ibm.com
4 WATSONX_MODEL_ID=ibm/granite-13b-instruct-v2
5 PINECONE_API_KEY=pcsk_22YGb3_9RY8BMqaUZN55nkxUA7nR7ZyhBnKA1LjW44XtRpkeo43rCmj2yW4HrPhQfQbafu
6 PINECONE_ENV=us-east-1
7 INDEX_NAME=smartcity-policies
```

## Config.py file

```
app > core > config.py > Settings > Config
8  class Settings(BaseSettings):
12
13
14      # Watsonx configs
15      WATSONX_API_KEY: str = "pAjcxi3Df0g687V0mCe3_Q8TmRKSDlp9wulXo52qwNn5"
16      WATSONX_PROJECT_ID: str = "f371addd-61dd-4ff0-882d-571db5a32aea"
17      WATSONX_URL: str = "https://eu-de.ml.cloud.ibm.com"
18      WATSONX_MODEL_ID: str = "ibm/granite-13b-instruct-v2"
19
20
21      class Config:
22          env_file = ".env"
23          extra = "allow"
24
25  settings = Settings()
26
```

## Pinecone Initialization:

pinecone\_client.py written to initialize the Pinecone vector index (smartcity-policies). Ensured creation with correct dimension=384 matching embedding model.

## Phase 2 – IBM Watsonx Integration

Watsonx Key & Model Configuration: Set up .env with:



WATSONX\_API\_KEY, PROJECT\_ID, MODEL\_ID

Endpoint Testing:

Validated /chat, /policy/summarize, and /get-eco-tips FastAPI routes using Swagger UI.



## Citizen Feedback

POST	/submit-feedback	Submit Feedback
------	------------------	-----------------

## Citizen Tips

GET	/get-eco-tips	Get Tips
-----	---------------	----------

## Admin Tools

POST	/generate-report	Generate Report
------	------------------	-----------------

GET	/anomaly-alerts	Get Alerts
-----	-----------------	------------

Phase 3 – Backend API Routers API Routes Implemented:

Developed modular routers:

chat\_router.py

feedback\_router.py

eco\_tips\_router.py

kpi\_upload\_router.py

anomaly\_checker.py

vector\_router.py, etc.

Testing & Validation:

Each route tested for:

JSON payload correctness

File upload parsing

Error handling & logging

Swagger auto-documentation generation

## Chat Assistant

**POST** `/chat/ask-assistant` Ask Question

## Policy Summarizer

**POST** `/policy/summarize-policy` Summarize

**GET** `/policy/test-llm` Test Llm

**GET** `/policy/summarize-from-file` Summarize From File

**POST** `/policy/summarize-uploaded-file` Summarize Uploaded File

**POST** `/policy/generate-markdown-report` Generate Md From Text

**POST** `/policy/generate-pdf-report` Generate Pdf From Text

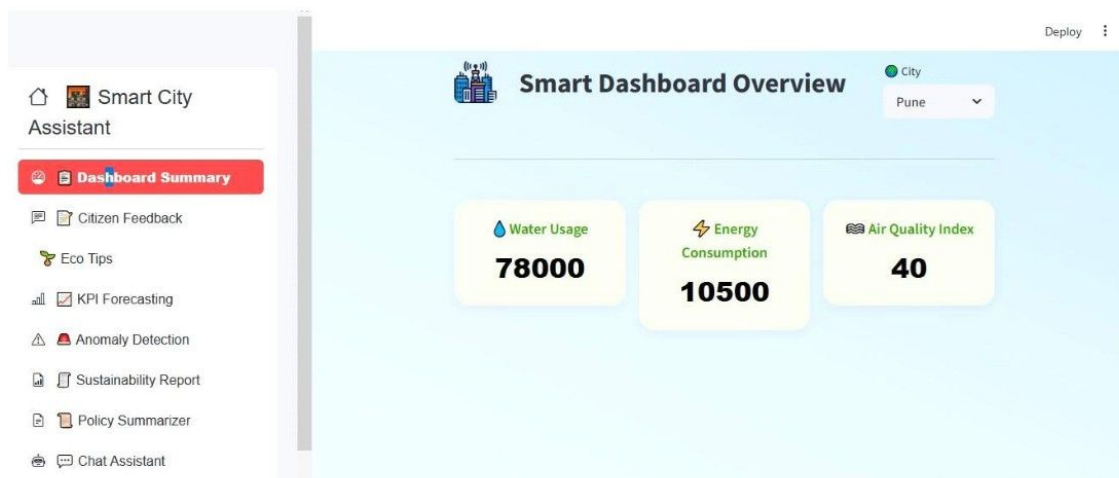
**POST** `/policy/upload-txt-generate-markdown` Generate Md From Uploaded Txt

**POST** `/policy/upload-txt-generate-pdf` Generate Pdf From Uploaded Txt

## Phase 4 – Frontend UI Design

Streamlit UI Structure Implemented:

Created central file `smart_dashboard.py` with conditional rendering for each module using sidebar navigation.



Component Development:

Developed reusable Streamlit components: `summary_card.py` – Beautiful KPI cards `chat_assistant.py` – Text prompt and AI reply `feedback_form.py`, `eco_tips.py`, `report_generator.py`, etc.

The screenshot shows the 'Smart City Assistant' sidebar on the left with the 'Citizen Feedback' option highlighted in red. The main panel is titled 'Submit Your Feedback or Report an Issue' and includes a 'Deploy' button in the top right. The form contains the following fields:

- A text input for 'Your Name'.
- A dropdown menu for 'Type of Issue' with 'Garbage' selected.
- A text area for 'Describe the issue or suggestion'.
- A 'Submit' button at the bottom.

The screenshot shows the 'Smart City Assistant' sidebar on the left with the 'Eco Tips' option highlighted in red. The main panel is titled 'Eco-Friendly Tips Assistant' and includes a 'Deploy' button in the top right. The form contains the following fields:

- A text input for 'Enter a topic (e.g., recycling, energy, water, plastic):'.
- A 'Get Eco Tip' button at the bottom.

UI Enhancements Done:

Gradient backgrounds

Icon-rich sidebar using streamlit-option-menu Rounded buttons, font styles, padding fixes

Phase 5 – Pinecone & Document Embedding

Embedding Logic Built:

Created document\_embedder.py and document\_retriever.py using sentence-transformers.

Phase 6 – Report Generation & Deployment

Granite LLM Report Generator:

report\_generator.py takes city name and KPI data, generates detailed city sustainability report using Granite LLM prompts.

Markdown & PDF Support:

Output formatted to text block for copy/paste or PDF download (optional).

End-to-End Integration Testing: Final dashboard tested on all 8 features: KPI dashboard, feedback form, policy summarization, eco tips, chat, anomaly check, vector search, report generation.