Sustainable Smart City Assistant Using IBM Granite LLM Document

Introduction

- Project title : Sustainable Smart City Assistant Using IBM Granite LLM
- Team member : Jothika.K
- Team member :Abivarshini .G
- Team member :Aysha bygum .U.A
- Team member :Bhavani. V

- **project overview**
- **Architecture**
- **Setup Instructions**
- **Folder Structure**
- **Running the Application**
- **API Documentation**
- **User Interface**
- **Testing**

Project Overview

The **Sustainable Smart City Assistant** is an AI-powered conversational platform designed to help citizens, city administrators, and policymakers make better decisions for building and managing eco-friendly urban environments. Leveraging **IBM Granite LLM** along with real-time city data, the assistant provides actionable insights, guidance, and support on sustainability initiatives.

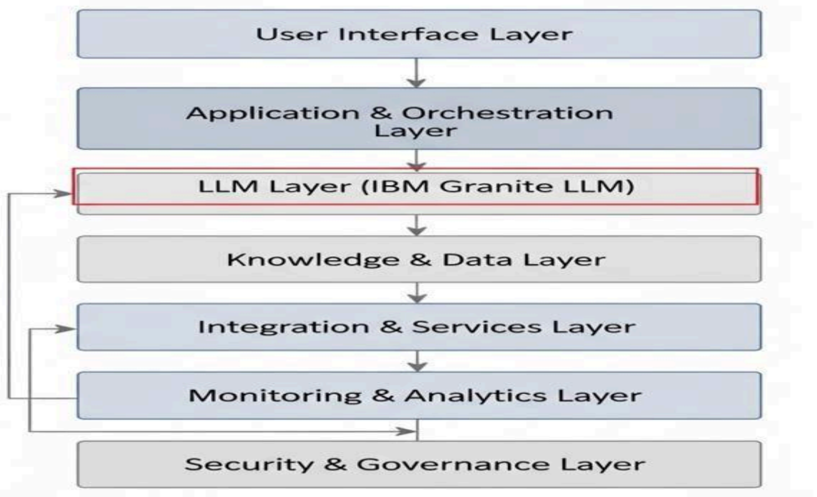
This project integrates **large language models (LLMs)** with **smart city infrastructure**, enabling citizens to interact with urban systems through natural language. The assistant simplifies complex data — such as energy usage, traffic congestion, waste management, air quality, and water conservation — into clear, actionable recommendations.

Key Objectives

1. **Citizen Support** – Provide instant answers to queries about public services, recycling, renewable energy programs, or eco-friendly transport options.
2. **Policy Assistance** – Help city planners and officials analyze sustainability data, draft reports, and track progress against carbon reduction targets.
3. **Real-time Insights** – Deliver up-to-date information on air quality, energy demand, or water consumption from IoT and sensor networks.
4. **Awareness & Engagement** – Educate citizens about sustainable practices, incentives, and community programs to encourage greener living.
5. **Scalability & Safety** – Ensure the system is secure, transparent, and adaptable to different cities and policy frameworks.

Architecture

The **Sustainable Smart City Assistant** follows a modular and scalable architecture that integrates **IBM Granite LLM** with city data sources, IoT infrastructure, and user-facing applications. The system ensures secure, real-time, and context-aware interactions while supporting both citizens and city administrators.


User Interface Layer
Application & Orchestration Layer
LLM Layer (IBM Granite LLM)
Knowledge & Data Layer
Integration & Services Layer
Monitoring & Analytics Layer
Security & Governance Layer

1. User Interface Layer

- **Channels**: Web app, mobile app, chatbots (WhatsApp, Messenger), voice assistants.
- **Features**: Conversational interface, quick-action buttons (e.g., "Report waste issue"), multilingual support, accessibility compliance.

2. Application C Orchestration Layer

- **Request Handler**: Manages user queries, applies authentication & authorization.
- **Prompt Orchestrator**: Prepares structured prompts for IBM Granite LLM using user context + retrieved documents.
- **Workflow Engine**: Triggers backend workflows (e.g., scheduling waste pickup, fetching energy reports).
- **Safety & Compliance Filters**: Content moderation, bias checks, escalation to human agents if needed.

3. LLM Layer (IBM Granite LLM)

- **Core Reasoning**: Generates responses in natural language.
- **RAG Integration**: Enhanced with **retrieval-augmented generation (RAG)** from city datasets and policy documents.
- **Domain Adaptation**: Fine-tuned or customized with sustainability-specific data.

4. Knowledge C Data Layer

- **Vector Database**: Stores municipal documents, sustainability guidelines, FAQs (for RAG).
- **City Data APIs**: Energy usage, transport schedules, waste collection, water supply, etc.

- **IoT & Sensor Feeds**: Real-time data from air quality monitors, traffic systems, weather stations.
- **Citizen Feedback DB**: Stores support tickets, surveys, and complaints for analytics.

5. Integration C Services Layer

- **GIS / Mapping Systems**: Location-aware insights (nearest recycling center, bike station).
- **External APIs**: Weather services, national energy dashboards, health advisories.
- **Municipal Services**: Waste pickup booking, public transport updates, utility billing.

6. Monitoring C Analytics Layer

- **System Monitoring**: Performance, uptime, response latency.
- **Usage Analytics**: Query types, service requests, citizen engagement stats.
- **Sustainability KPIs**: Track emissions reduction, recycling rates, energy savings.

7. Security C Governance Layer

- **Data Privacy Controls**: Compliance with GDPR, local data laws.
- **Audit Logs**: For traceability of responses and actions.
- **Role-based Access**: Different permissions for citizens, officials

Setup Instructions

Follow these steps to set up the **Sustainable Smart City Assistant** locally or on a cloud server.

1. Prerequisites

- **Python 3.9+** installed
- **pip** package manager
- Access to **IBM Granite LLM API** (via IBM watsonx.ai or Granite SDK)
- Git & Virtual Environment tools
- Optional: Docker (for containerized deployment)

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.py`** → main entry point (runs Gradio interface).
- **`modules/`** → all core functionality (eco tips, summarization, PDF handling).
- **`models/llm_handler.py`** → wrapper for calling IBM Granite (or Hugging Face) so you can swap models easily.
- **`scripts/`** → one-off scripts (like loading docs for RAG).
- **`data/`** → local test PDFs + vector DB storage.
- **`logs/`** → debugging + monitoring.
- **`tests/`** → ensures each module works

Running the Application

```
/source venv/bin/activate
```

```
pip install -r requirements.txt
fo: Isoaly rextiromens
fo: Iscorton ston
fo: Iscalys eertiom
```

```
export API_KEY=youy_here
/your_KEY=your_key.here
```

```
python app.py
/python_app.py {
/sxecurts_cortectiqpe
```

1. Activate Virtual Environment

If you created a virtual environment earlier, activate it:

```
source venv/bin/activate
venv\Scripts\activate
```

2. Install Dependencies

Make sure all requirements are installed:

```
pip install -r requirements.txt
```

3. Set Environment Variables

Create a `.env` file in the project root with your API keys:

```
IBM_API_KEY=your_ibm_granite_api_key
IBM_PROJECT_ID=your_project_id
IBM_REGION=us-southq
```

4. Run the Application

(a) Gradio Web App

```
python
```

(b) FastAPI Backend (Optional)

If you added an API server (`api.py`):

```
uvicorn api:app --reload
```

→ Visit Swagger docs at: `http://127.0.0.1:8000/docs`

Stop the Application

Press:

```
CTRL + C
```

to stop the running server.

API Documents – Sustainable Smart City Assistant

📝 API Documents – Sustainable Smart City Assistant

Endpoint	Method	Description	Request Format	Response Format
`/eco-tips`	`POST`	Generate eco-friendly tips based on keywords.	JSON: `{ "keywords": "plastic waste, energy saving" }`	JSON: `{ "tips": "Switch to reusable bags, use LED bulbs, unplug appliances when not in use." }`
`/policy-summary`	`POST`	Summarize sustainability policy (from PDF upload or raw text).	**Option A (PDF):** Multipart Form with `file` field. **Option B (Text):** JSON: `{ "policy_text": "Policy description here..." }`	JSON: `{ "summary": "Policy promotes renewable energy adoption and emission reduction by 2030." }`
`/health`	`GET`	Check if API is running.	None	JSON: `{ "status": "ok", "service": "Sustainable Smart City Assistant", "version": "1.0" }`

User interface

The **User Interface** of the Sustainable Smart City Assistant is designed to be **intuitive, interactive, and accessible**, enabling citizens, policymakers, and city administrators to engage with the system easily. The UI is built using **Gradio** for web-based interaction, with support for text, file uploads, and actionable buttons.

1. Design Principles

- **Simplicity:** Minimalist layout to focus on tasks like generating eco tips or summarizing policies.
- **Responsiveness:** Works on desktop, tablet, and mobile devices.
- **Accessibility:** Clear labels, high contrast, and readable fonts to accommodate all users.
- **Guided Interaction:** Step-by-step prompts, tooltips,

Testing

- Testing was done in multiple phases:
- Unit Testing: For prompt engineering functions and utility scripts
- API Testing: Via Swagger UI, Postman, and test scripts
- Manual Testing: For file uploads, chat responses, and output consistency
- Edge Case Handling: Malformed inputs, large files, invalid API keys
- Each function was valida