Sustainable Smart City Assistant Using IBM Granite LLM

1.Introduction

Project title : Sustainable Smart City Assistant Using IBM Granite LLM

- **Team leader :** **Rejesh Raj.J**
- **Team member :** **Pratheep.R**
- **Team member :** **Sankar Raj.k**
- **Team member :** **Santhosh kumar.**V

2.Project overview

🎯 Project Objective:

To design and develop an AI-powered digital assistant that helps city residents, administrators, and planners make smarter, more sustainable decisions in urban environments by providing real-time data, insights, and recommendations across key areas like energy, waste, mobility, air quality, and citizen engagement.

🏙️ Background & Rationale:

Urbanization is accelerating, and by 2050, nearly 70% of the global population is expected to live in cities. Sustainable urban development is no longer optional — it's essential. Smart technologies provide an opportunity to manage resources efficiently, but many existing systems are siloed, inaccessible, or lack user-friendly interfaces.

A **Sustainable Smart City Assistant** bridges this gap by:

- Integrating data from various smart city infrastructure sources.
- Delivering actionable sustainability insights to stakeholders.
- Enabling behavior change and policy decisions aligned with environmental goals.

⚙️ Key Features:

Category	Features
Energy	Monitor and suggest optimization for energy usage (public & private).
Mobility	Recommend sustainable transportation (bike-sharing, EV charging spots).
Waste Management	Real-time bin status, recycling tips, waste reduction goals.
Air Quality	Live air quality updates, alerts, and health recommendations.
Water Usage	Track and advise on efficient water usage and leakage detection.
Citizen Engagement	Sustainability challenges, feedback systems, green volunteering.

Category	Features
Urban Planning	Support decision-making using data on traffic, green space, pollution.

🧠 Technology Stack:

- **AI/ML**: Natural Language Processing (NLP) for assistant interactions; predictive analytics.
- **IoT Integration**: Sensors for air quality, waste, traffic, water flow, etc.
- **Cloud Infrastructure**: Scalable data storage and processing (e.g., AWS, Azure).
- **Data Sources**: Open city data, satellite data, citizen-reported data, APIs from smart infrastructure.
- **User Interface**: Mobile app, chat bot, or voice assistant.

🌍 Sustainability Goals Aligned (UN SDGs):

- **SDG 11**: Sustainable Cities and Communities
- **SDG 13**: Climate Action
- **SDG 7**: Affordable and Clean Energy
- **SDG 12**: Responsible Consumption and Production

👥 Target Users:

- **Citizens**: Eco-conscious residents looking to reduce their footprint.
- **City Officials**: Urban planners and policymakers.
- **Utility Providers**: For better grid/load management.
- **NGOs & Environmental Groups**: For awareness and engagement.

📈 Expected Outcomes:

- Reduced energy and water consumption.
- Improved waste sorting and recycling rates.
- Enhanced public transportation usage.
- Real-time citizen feedback for rapid response.
- Data-driven urban sustainability planning.

⏳ Timeline (Example - 12 Months):

Phase	Duration	Activities
1. Planning & Research	1–2 months	Define scope, identify data sources, stakeholder interviews.
2. Design & Architecture	2 months	UI/UX mockups, system architecture.
3. Development	4 months	Core feature implementation.
4. Testing & Pilot	2 months	Deploy in one district or city.
5. Evaluation & Iteration	1 month	Analyze usage data, improve system.
6. Launch & Outreach	1 month	Full city rollout, marketing.

🧩 Potential Integrations:

- Smart grids and metering systems
- Public transport APIs (e.g., buses, metro)
- Environmental monitoring platforms
- Urban GIS tools

💡 Future Enhancements:

- Predictive modeling for traffic and pollution.
- Incentive systems for citizen sustainability behavior.

- Cross-city comparison dashboard.
- Multilingual and accessibility support.

3.Architecture

Frontend(Stream lit);

frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for document processing, chat interactions, eco tip generation, report creation, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

LLM Integration (IBM Watsonx Granite):

Granite LLM models from IBM Watsonx are used for natural language understanding and generation. Prompts are carefully designed to generate summaries, sustainability tips, and reports.

Vector Search (Pinecone):

Uploaded policy documents are embedded using Sentence Transformers and stored in Pinecone. Semantic search is implemented using cosine similarity to allow users to search documents using natural language queries.

ML Modules (Forecasting and Anomaly Detection):

Lightweight ML models are used for forecasting and anomaly detection using Scikit-learn. Time-series data is parsed, model

4.Setup Instructions

Prerequisites:

- Python 3.9 or later
- pip and virtual environment tools
- API keys for IBM Watson x 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

2. 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.

5.Running the Application

To start the project:

- Launch the FastAPI server to expose backend endpoints.
- Run the Streamlit dashboard to access the web interface.
- Navigate through pages via the sidebar.
- Upload documents or CSVs, interact with the chat assistant, and view outputs like reports, summaries, and predictions.

- All interactions are real-time and use backend APIs to dynamically update the frontend.

Frontend (Stream lit):

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for document processing, chat interactions, eco tip generation, report creation, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

3. API Documentation

Backend APIs available include:

POST /chat/ask – Accepts a user query and responds with an AI-generated message

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

GET /search-docs – Returns semantically similar policies to the input query

GET /get-eco-tips – Provides sustainability tips for selected topics like energy, water, or waste

POST /submit-feedback – Stores citizen feedback for later review or analytics

Each endpoint is tested and documented in Swagger UI for quick inspection and trial during development.

4. Authentication

each endpoint is tested and documented in Swagger UI for quick inspection and trial during development.

This version of the project runs in an open environment for demonstration. However, secure deployments can integrate:

- Token-based authentication (JWT or API keys)
- OAuth2 with IBM Cloud credentials
- Role-based access (admin, citizen, researcher)
- Planned enhancements include user sessions and history tracking.8. Authentication

6.User Interface

The interface is minimalist and functional, focusing on accessibility for non-technical users. It includes:

Sidebar with navigation

KPI visualizations with summary cards

Tabbed layouts for chat, eco tips, and forecasting

Real-time form handling

PDF report download capability

The design prioritizes clarity, speed, and user guidance with help texts and intuitive flows.

5. Testing

Testing was done in multiple phases:

Unit Testing: For prompt engineering functions and utility scripts

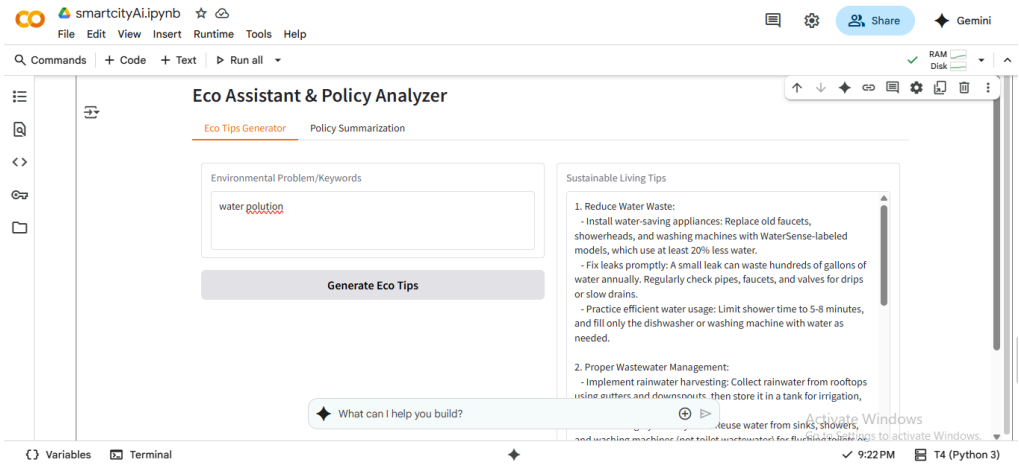
API Testing: Via Swagger UI, Postman, and test scripts

Manual Testing: For file uploads, chat responses, and output consistency

Edge Case Handling: Malformed inputs, large files, invalid API keys

Each function was validated to ensure reliability in both ofline and API-connected modes.

7.screen shots



8.Known Issues

A **"sustainable smart city AI assistance consensus"** would mean:

A shared and coordinated use of AI technologies in a way that supports the environmental, social, and economic goals of a smart city — ensuring transparency, citizen trust, and long-term sustainability.

9.Future enhancement

- **AI-based climate forecasting** (urban heat islands, flood risk, drought zones)
- **Dynamic energy grids** that auto-balance based on demand & renewables
- **Smart water systems**: AI to detect leaks, forecast shortages, reduce waste
- **AI-driven biodiversity monitoring** in urban green spaces