

# **SUSTAINABLE SMART CITY ASSISTANT USING IBM GRANITE LLM**

## **1. Introduction**

- Project title : Sustainable Smart City Assistant Using IBM Granite LLM
- Team member : Devathursini.S (Team Leader)
- Team member : Kiruthiga.R
- Team member : Logeshwari.G
- Team member : Maria Flora Jones.F

## **2. Project Overview**

### **Purpose:**

The **Sustainable Smart City Assistant project using IBM Granite LLM** aims to leverage AI to enhance urban sustainability, efficiency, and livability by optimizing city services like traffic, energy, waste, and water management through real-time data analysis. The assistant would promote green technologies, suggest sustainable practices, and engage citizens by providing personalized recommendations on energy conservation, transportation, and waste reduction. It would also support city officials in making data-driven decisions, offer insights on climate resilience and disaster management, and automate tasks related to sustainability. Additionally, the assistant would facilitate real-time interactions, integrate with IoT devices, and assist in long-term urban planning to create eco-friendly, adaptable cities. Ultimately, the project seeks to foster a smarter, more sustainable urban ecosystem that balances development with environmental preservation.

### **Features:**

#### **Real-Time City Optimization**

Key Points: Manages traffic, energy, waste, and water.

Functionality: Uses IoT data to optimize city services in real-time.

#### **Sustainability Recommendations**

Key Points: Recommends green technologies and eco-friendly urban planning.

Functionality: Analyzes environmental data and suggests improvements.

### **Citizen Engagement**

Key Points: Allows citizens to interact, report issues, and get personalized eco-tips.

Functionality: Provides a platform for real-time citizen queries and feedback

### **Environmental Monitoring**

Key Points: Tracks pollution, air quality, and energy usage.

Functionality: Sends alerts and suggestions for reducing environmental impact.

### **Data-Driven Decision Support**

Key Points: Supports city planning and sustainability efforts.

Functionality: Analyzes data for informed urban development decisions.

### **Automated Sustainability Reporting**

Key Points: Automates tracking of sustainability progress.

Functionality: Generates reports on energy use, waste reduction, and other metrics.

### **Climate Risk Prediction**

Key Points: Identifies climate-related threats (e.g., floods, heatwaves).

Functionality: Offers strategies for disaster preparedness and risk mitigation.

### **Personalized Recommendations**

Key Points: Provides tailored advice for individuals on sustainability.

Functionality: Recommends energy-saving, transport, and recycling options.

### **IoT Integration**

Key Points: Integrates with smart devices across the city.

Functionality: Automates and optimizes city infrastructure in real-time.

### **Long-Term Urban Planning**

Key Points: Simulates future urban growth and sustainability scenarios.

Functionality: Offers solutions for eco-friendly city expansion and development.

## **3. Architecture**

### **Data Collection:**

IoT Devices (sensors, cameras) gather real-time data on traffic, energy, waste, and environment.

**Data Processing:**

Edge Computing handles immediate decisions (e.g., traffic).

Data Lake stores and processes data for analysis.

**AI & Machine Learning:**

IBM Granite LLM powers NLP and AI-driven insights.

Predictive Analytics and Optimization improve city services.

**Smart Infrastructure:**

Integrates with city systems (traffic, energy, waste) for automation and optimization.

**Frontend (User Interface):**

Citizen Apps & Web Portals for interaction, queries, and receiving personalized tips.

Real-Time Dashboards show sustainability metrics.

**Backend:**

API Layer connects frontend with backend and external systems.

Cloud Services (IBM Cloud) host AI models and store data securely.

**Feedback & Monitoring:**

Real-Time Analytics track performance and user feedback for system improvements.

**Reporting & Analytics:**

Dashboards provide actionable insights and reports for city officials.

## 4. Setup Instructions

**Prerequisites**

IBM Cloud Account (for AI, IoT, and cloud services).

IoT Devices (sensors, smart meters).

Development Tools: Node.js, React (or Angular), Python, Docker.

**Backend Setup**

IBM Cloud Setup:

Create Watson Studio (AI/ML), Cloud Functions (serverless functions), and IoT Platform

API Development:

Set up a Node.js backend with Express to handle requests and integrate IBM Granite LLM for AI responses.

IoT Integration:

Use IBM IoT Platform to connect devices and send data (traffic, energy) to cloud.

## Frontend Setup

React App:

Create a React app for citizen interaction and dashboards.

```
npm create-react-app smart-city-dashboard
```

API Integration:

Use axios to call backend APIs for real-time data and recommendations.

Deploy Frontend:

Deploy the React app to IBM Cloud (via Cloud Foundry or Static Hosting).

## Cloud Integration & Security

Cloud Functions:

Deploy backend as serverless functions.

Security:

Use API Gateway and OAuth/JWT for secure access.

## Testing & Deployment

Test IoT Data:

Ensure sensors send data correctly.

Test API & Frontend:

Test backend responses and frontend displays.

Deploy:

Deploy everything to IBM Cloud and monitor with Cloud Monitoring.

## 5. Folder Structure

**/backend:**

**/controllers:** API logic and handlers.

**/routes:** API route definitions.

**server.js:** Main backend server file.

### **/frontend:**

**/components:** React or Angular UI components.

**App.js:** Main entry point for the frontend.

### **/config:**

**config.js:** Configuration files for settings, API keys, etc.

**README.md:** Project documentation and instructions.

## **6. Running the Application**

### **Clone the Repository**

- Clone the repository to your local machine:
- `git clone https://github.com/your-repository/smart-city-assistant.git`
- `cd smart-city-assistant`

### **Backend Setup**

#### **Step 1: Install Dependencies**

- Navigate to the /backend directory:
- `cd backend`
- Install required packages:
- `npm install`

#### **Step 2: Configure Environment**

- Create a .env file in the /backend directory and add necessary configuration variables (e.g., database URL, API keys):
- `PORT=3000`
- `DB_URL=your-database-url`
- `API_KEY=your-api-key`

#### **Step 3: Start the Backend Server**

- Run the backend server:
- `npm start`
- The backend should be live at <http://localhost:3000>.

## Frontend Setup

### Step 1: Install Dependencies

- Navigate to the /frontend directory:
- `cd frontend`
- Install frontend dependencies:
- `npm install`

### Step 2: Configure Environment

- Create a .env file in the /frontend directory and set the API URL:
- `REACT_APP_API_URL=http://localhost:3000`

### Step 3: Start the Frontend Server

- Run the frontend server:
- `npm start`
- The frontend should now be running at `http://localhost:3001`.

## Accessing the Application

- Backend: Open `http://localhost:3000` for API testing.
- Frontend: Open `http://localhost:3001` to interact with the UI.

## Stopping the Servers

- Press `Ctrl + C` in both backend and frontend terminal windows to stop the servers.

## 7. 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.

## 8. 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.

## 9. 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.

## 10. Testing

Testing was done in multiple phases:

**Unit Testing:** For prompt engineering functions and utility scripts

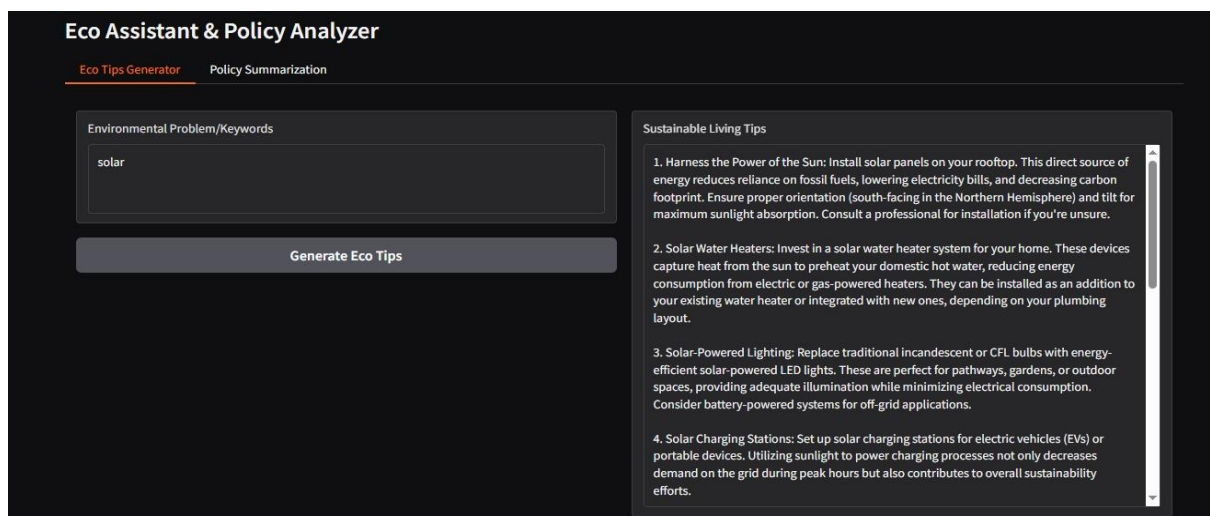
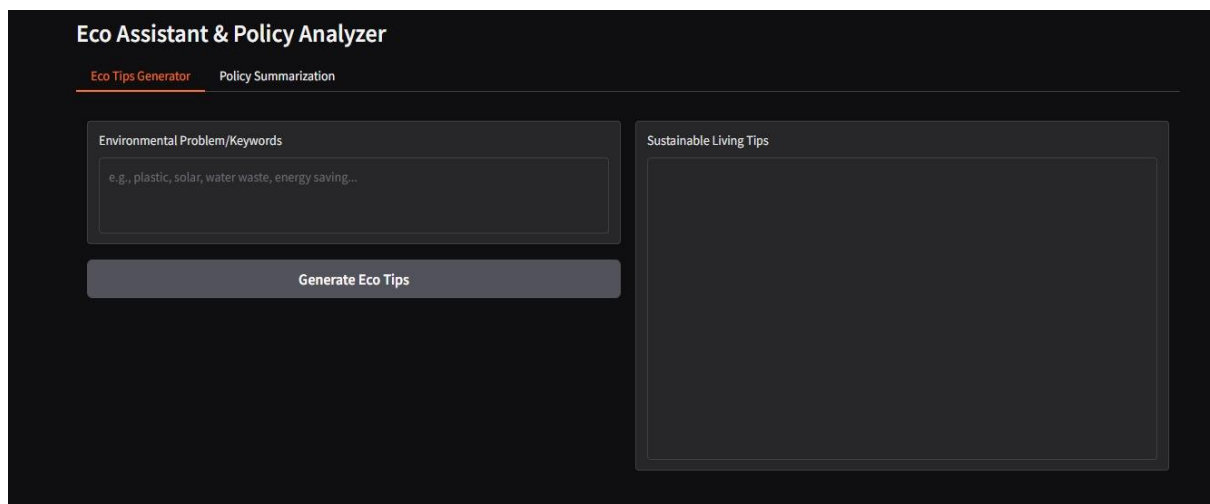
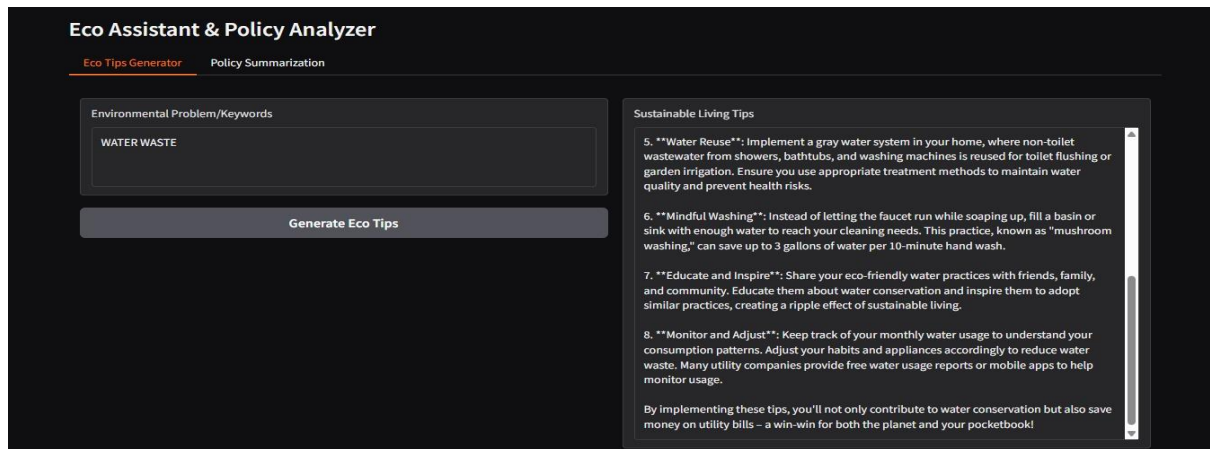
**API Testing:** Via Swagger UI, Postman, and test scripts

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

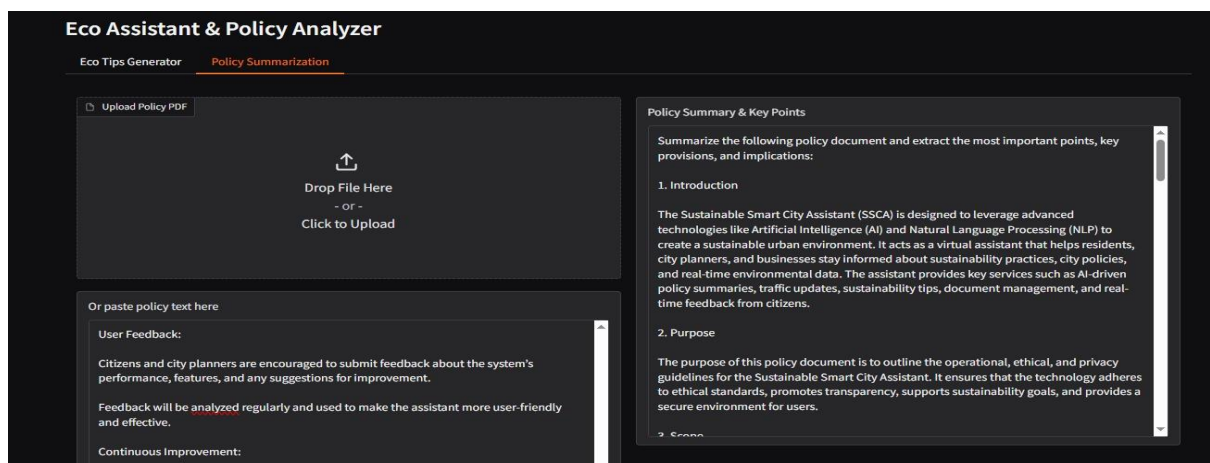
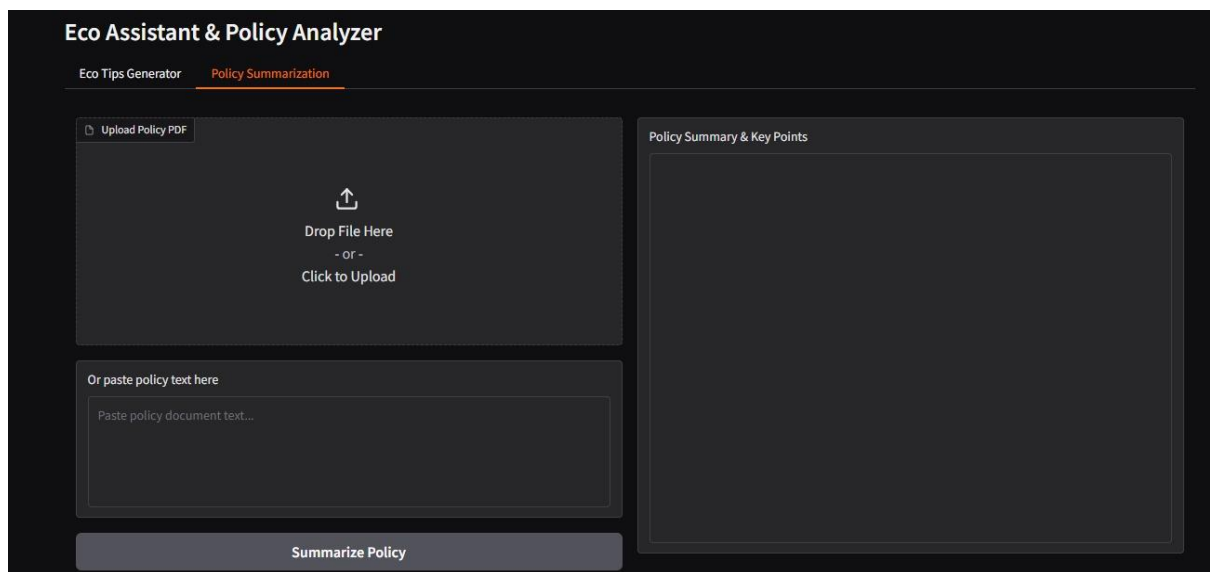
**Edge Case Handling:** Malformed inputs, large files, invalid API keys

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

## 11.Screenshots







## 12. Known Issues

### Technical Issues

- **Hallucinations:** May give incorrect or made-up answers.
- **Prompt Sensitivity:** Output quality changes with small prompt tweaks

**Limited Domain Accuracy:** May struggle with smart city-specific topics (e.g., energy grids, urban planning).

**Latency:** Large models can be slow, not ideal for real-time needs.

**Multimodal Limitations:** Speech/image inputs may have higher error rates.

## **Safety & Ethics**

**Bias & Unsafe Outputs:** Risk of offensive, biased, or misleading content.

**Explainability:** Hard to trace why the assistant made a decision.

## **Data & Privacy**

**Sensitive Data Handling:** Risk if private city data is used without proper control.

**Compliance:** Needs to meet local privacy laws (e.g. GDPR).

## **Sustainability Concerns**

**High Energy Use:** Large models require lots of compute, raising environmental concerns.

## **Maintenance Needs**

Requires regular updates for models, data, and safety check

# **13.Future Enhancement**

## **Environmental Monitoring**

- Real-time analysis of air, water, and noise using IoT + Granite.
- Predictive alerts for pollution and climate risks.
- Personal & neighborhood carbon footprint tracking.

## **Citizen Interaction**

- Multilingual and voice-based support for inclusivity.
- Personalized sustainability tips and alerts.

## **Urban Planning & Mobility**

- AI-generated smart urban development models.
- Sustainable transport suggestions and traffic optimization.

## **Energy & Waste Management**

- Smart energy-saving recommendations for homes.
- AI-assisted waste sorting and optimized collection routes.

## **AI Ethics & Engagement**

- Privacy-first data handling and explainable AI.
- Eco-score gamification to engage citizens.

## **IBM Ecosystem Integration**

- Connect with IBM Maximo, Watsonx for infrastructure and decision support
- Use Granite LLM for intelligent, scalable city solutions.