**Sustainable Smart City**

**Assistant Using IBM Granite LLM**

**Project Documentation**

**1)Introduction**

**Project Title** :Sustainable Smart City Assistant using IBM Granite LLM

**Team Members**: SARANYADEVI P

**Team Members**: DHANUSHKA S

**Team Members**: KEERTHIKA R

**Team Members**: MATHUSRI R

**2)Project Overview**

The **Sustainable Smart City Assistant** leverages **IBM Granite LLMs** (Large Language Models) to provide AI-driven insights, decision support, and citizen engagement tools for managing urban sustainability. The assistant integrates with IoT systems, open city data, and municipal applications to improve **energy efficiency, mobility, waste management, and citizen services**.

**Key Goals:**

* **Sustainability:** Optimize energy use, reduce emissions, promote renewable resources.
* **Smart Mobility:** Support traffic management, EV charging, and public transport optimization.
* **Citizen Engagement:** Provide multilingual, natural language interactions for residents and city officials.
* **Policy & Governance:** Assist decision-makers with data-driven recommendations and scenario simulations.
* **Interoperability:** Integrate with IoT devices, sensors, and cloud systems.

**3)Architecture**

**1. User Interaction Layer**

* **Channels:** Web app, mobile app, voice assistant, chatbot kiosks.
* **Multimodal Support:** Text, speech, image (e.g., citizens reporting potholes or waste issues).
* **Personalized Interfaces:** Tailored dashboards for citizens, city officials, and utility providers.

**2. AI & LLM Layer (IBM Granite)**

* **Granite LLM Core:** Provides natural language understanding, reasoning, and response generation.
* **Domain Adaptation:** Fine-tuned on sustainability, urban planning, mobility, and policy data.
* **Responsible AI Controls:** Built-in bias mitigation, explainability, and guardrails for safe responses.
* **Conversational Orchestration:** Manages complex multi-turn conversations with context retention.

**3. Knowledge & Data Layer**

* **City Knowledge Graph:** Aggregates structured/unstructured data (policies, regulations, city plans).
* **IoT Data Streams:** Traffic sensors, energy grids, air quality monitors, waste bins, water systems.
* **External APIs & Open Data:** Weather data, transportation APIs, renewable energy forecasts.
* **Sustainability Metrics Engine:** Tracks KPIs like carbon footprint, energy usage, and water savings.

**4. Application & Service Layer**

* **Smart Mobility Services:** Route optimization, EV charging guidance, public transit updates.
* **Energy Management Services:** Demand-response optimization, smart grid insights, renewable integration.
* **Waste & Water Services:** Intelligent collection routing, water leakage detection.
* **Citizen Services:** Report issues, request permits, access sustainable living recommendations.

**5. Integration & Middleware**

* **APIs & Connectors:** Secure integration with IoT platforms, GIS systems, and enterprise apps.
* **Event Bus / Kafka Layer:** Real-time data streaming and processing.
* **Identity & Access Management (IAM):** Role-based access for citizens, officials, and enterprises.

**6. Infrastructure Layer**

* **Deployment Environment:** Hybrid Cloud (IBM Cloud + Edge Nodes for IoT).
* **Data Storage:** Time-series databases (for sensor data), document DB (for policies), graph DB (for city relationships).
* **Edge AI Processing:** Low-latency analysis near IoT sources.
* **Security & Compliance:** Data encryption, GDPR/CCPA compliance, ethical AI safeguards.

**4)Setup Instructions**

**1. Prerequisites**

* IBM Cloud account with **AI and Granite LLM access**
* Access to **Watsonx.ai** (IBM’s AI/ML platform)
* Python 3.10+ (for client SDKs and backend services)
* Node.js (if you’re building a web or mobile UI)
* Docker / Kubernetes (for deployment at scale)
* IoT data sources or city open data APIs

**2. Provision Granite LLM on IBM Cloud**

1. Log into IBM Cloud.
2. Go to **Watsonx.ai** → Create a **Granite LLM instance**.
3. Select a **Granite model** (e.g., Granite-13B-chat for conversational tasks).
4. Generate an **API Key** and note the **endpoint URL**.

**3. Integrate IoT / Open Data APIs**

* Connect real-time **traffic sensors**, **air quality monitors**, **energy usage data**.
* Store data in a time-series DB (e.g., Influx DB, MongoDB).
* Provide API endpoints
* **. Front** to the assistant so Granite can retrieve contextual data.

**4.** **Frontend (Optional)**

* **Web app:** React / Vue with REST calls to /ask endpoint.
* **Mobile app:** Flutter/React Native with API integration.
* **Voice assistant:** Connect to Speech-to-Text and Text-to-Speech (IBM Watson Speech APIs).

**5)Folder structure**

smart-city-assistant/

│

├── backend/ # Python backend (Flask/FastAPI)

│ ├── app.py # Main entry point (Flask API server)

│ ├── config.py # Configs (API keys, endpoints, env vars)

│ ├── requirements.txt # Python dependencies

│ ├── services/

│ │ ├── granite\_client.py # Granite LLM client wrapper

│ │ ├── iot\_service.py # IoT data fetch + preprocessing

│ │ ├── knowledge\_graph.py # City policies + sustainability data

│ │ └── analytics.py # Metrics, KPIs (carbon, water, energy)

│ ├── routes/

│ │ ├── ask.py # /ask endpoint → Granite LLM

│ │ ├── mobility.py # Smart mobility APIs

│ │ ├── energy.py # Energy management APIs

│ │ ├── waste.py # Waste + water APIs

│ │ └── citizens.py # Citizen engagement services

│ └── utils/

│ ├── auth.py # IAM + token handling

│ ├── formatters.py # Response formatting (JSON → UI)

│ └── logger.py # Central logging

│

├── frontend/ # React (or Vue/Angular) UI

│ ├── public/ # Static assets

│ ├── src/

│ │ ├── components/ # UI components (chat, dashboards, maps)

│ │ ├── pages/ # App pages (Home, Dashboard, Reports)

│ │ ├── services/ # API calls to backend

│ │ ├── App.js # Main React app

│ │ └── index.js # React entry point

│ ├── package.json # Frontend dependencies

│ └── tailwind.config.js # Styling (if using Tailwind)

│

├── data/ # Data pipelines + sample datasets

│ ├── sensors/ # IoT sensor data (traffic, energy, air quality)

│ ├── open\_data/ # Open government datasets

│ ├── policies/ # City sustainability policies/regulations

│ └── knowledge\_base/ # Text docs for fine-tuning/context

│

├── infra/ # Deployment + DevOps

│ ├── Dockerfile # Build backend service

│ ├── docker-compose.yml # Local multi-container setup

│ ├── k8s/ # Kubernetes manifests

│ │ ├── deployment.yaml

│ │ ├── service.yaml

│ │ └── ingress.yaml

│ └── code-engine.yaml # IBM Cloud Code Engine deployment config

│

├── tests/ # Unit & integration tests

│ ├── test\_backend.py

│ ├── test\_granite.py

│ └── test\_routes.py

│

├── docs/ # Documentation

│ ├── architecture.md # System architecture diagrams

│ ├── setup\_instructions.md # Setup & running guide

│ └── api\_reference.md # API documentation

│

├── .env.example # Example environment variables

├── README.md # Project overview

└── LICENSE # License file

**6)Running the Application**

**Test API**

curl -X POST http://localhost:5000/ask \

-H "Content-Type: application/json" \

-d '{"query":"What is today’s air quality and best route to work?"}'

 **Citizen asks a question** via app/voice →

 Query sent to **Flask backend** →

 Backend calls **Granite LLM** with context (IoT data, city policies) →

 Granite LLM generates a **response** →

 Answer returned to **frontend dashboard or chatbot**.

**7)API Documentation**

 **IBM Granite LLM**: Backend uses an **IBM IAM access token** to call Granite APIs.

 **Users (citizens, officials)**: Authenticate via **JWT tokens**.

**Smart Mobility – Get Routes**

**GET** /mobility/routes

Suggests sustainable transport routes.

**Query Parameters:**

* origin (string, required)
* destination (string, required)
* mode (optional: car, bus, bike, walk, ev)

**Example:**

GET /mobility/routes?origin=StationA&destination=StationB&mode=bike

**Response:**

{

"best\_route": "StationA → Greenway → StationB",

"duration": "15 min",

"co2\_saving": "0.42 kg CO₂ compared to car"

### **Sustainability Reports (Officials Only)**

**GET** /reports/sustainability

Generates a Granite-powered summary of KPIs.

**Response:**

{

"summary": "This week, renewable energy share increased to 50%, water consumption anomalies were detected in 2 districts, and air quality improved by 12%.",

"recommendations": [

"Expand EV charging in District A",

"Schedule water pipeline checks in District C"

]

}

**8)Authentication**

The system needs **two layers of authentication**:

1. **Backend ↔ IBM Granite LLM** (service-to-service)
2. **Citizen/Official ↔ Assistant API** (user authentication)

**Backend ↔ IBM Granite LLM Authentication**

* Uses **IBM Cloud IAM (Identity and Access Management)**
* The backend exchanges an **IBM Cloud API Key** for a **Bearer token**
* This token is then used in all Granite LLM requests

**Step 1: Get IAM Access Token**

curl -X POST "https://iam.cloud.ibm.com/identity/token" \

-H "Content-Type: application/x-www-form-urlencoded" \

-d "grant\_type=urn:ibm:params:oauth:grant-type:apikey&apikey=YOUR\_IBM\_API\_KEY"

**Response:**

{

"access\_token": "eyJhbGciOiJSUzI1NiIs...",

"expiration": 1737021210,

"token\_type": "Bearer"

}

**Step 2: Use IAM Token in Granite Requests**

Example (Python):

headers = {

"Authorization": "Bearer " + iam\_token,

"Content-Type": "application/json"

}

payload = {

"model\_id": "granite-13b-chat",

"input": "Suggest ways to reduce traffic emissions"

}

response = requests.post(

"https://us-south.ml.cloud.ibm.com/ml/v1/text/generation?version=2023-05-29",

headers=headers,

json=payload

)

**9)User Interface**

**The UI should be:**

* **Citizen-friendly**: Simple, conversational, mobile-first.
* **Official-friendly**: Data-rich dashboards with analytics.
* **Multimodal**: Support text, voice, and maps.
* **Accessible**: Multilingual, ADA-compliant, and responsive.

**1. Home Screen**

* Welcome message
* Quick actions:
  + “Ask Assistant” (chat)
  + “Report Issue” (pothole, waste overflow, etc.)
  + “Plan a Trip” (mobility & eco-friendly routes)
* Notifications (waste pickup schedule, air quality alerts)

**2. Chat Assistant (Granite LLM)**

* Conversational interface (chat bubbles)
* Voice input (speech-to-text) + output (text-to-speech)
* Map integration (best bike/EV routes, cleanest areas)

**3. Citizen Reports**

* Form for reporting issues (photo upload, location tagging via GPS/map)
* View status of submitted tickets (tracking progress)

**4. Eco-Dashboard (Personal)**

* Daily carbon footprint estimate
* Water/electricity usage tips
* Local air quality index (AQI)

**Official (City Planner / Admin) UI**

**1. Dashboard Overview**

* City KPIs:
  + **Energy** (usage, renewable share)
  + **Mobility** (traffic, EV charging load)
  + **Environment** (air quality, emissions)
  + **Waste/Water** (collection efficiency, anomalies)
* Alerts & anomalies (e.g., “Water leak detected in District A”).

**2. Reports Section**

* Generate **Granite-powered summaries** (weekly/monthly sustainability reports).
* Export to PDF/Excel.

**3. Mobility Insights**

* Heatmaps of traffic + pollution
* EV charging station utilization

**4. Issue Management**

* Incoming citizen reports (with photos + map locations)
* Ticketing system integration (assign, update, resolve)

**10)Testing**

**Unit Testing (Backend + Granite Integration)**

* Test **Granite API wrapper** (granite\_client.py)
* Test **data fetchers** (IoT, open data APIs)
* Use **mock Granite responses** so tests don’t burn credits

**API Testing (Flask/FastAPI Endpoints)**

* Use **Postman** or **Pytest with HTTPX**
* Validate /ask, /mobility/routes, /energy/usage, etc.
* Check **auth failures (401/403)**, missing params (400), success (200).

**Integration Testing**

* Simulate **real IoT + Granite calls** end-to-end.
* Example flow:
  1. Citizen asks: “Best cycling route today?”
  2. Backend pulls **air quality + traffic sensor data**
  3. Granite LLM uses that context → returns suggestion
* Validate response contains **data + reasoning**.

**11)Known Issues**

**Authentication & Security**

* **Short IAM token expiry** → IBM Granite IAM tokens expire in ~1 hour → backend must refresh automatically, or requests fail with 401 Unauthorized.
* **JWT management** → If refresh tokens aren’t implemented, users may be logged out too often.
* **API key leaks** → If IBM Cloud API keys are hardcoded or exposed in frontend, attackers could misuse Granite APIs.

**Granite LLM Limitations**

* **Hallucinations** → LLM may invent non-existent city policies or suggest invalid routes if not grounded in real data.
* **Context length limits** → Granite models have input token limits, which restricts how much IoT + policy data you can provide per request.
* **Latency** → Large Granite models (13B+) may introduce delays in real-time queries (e.g., route planning).
* **Bias** → Risk of biased or incomplete sustainability advice if training data lacks city-specific diversity.

**IoT & Data Integration**

* **Data freshness** → If IoT sensors fail or APIs lag, the assistant may provide outdated recommendations.
* **Heterogeneous data** → Different sensors (traffic, air, water) may use incompatible formats or units.
* **Missing fallback** → If APIs fail, the LLM may not know how to gracefully say: *“Real-time data unavailable, here’s last known info.”*

**Scalability & Performance**

* **Load spikes** → During peak hours (e.g., citizens checking air quality), backend may bottleneck.
* **Memory usage** → Storing large knowledge graphs or historical sensor data may exceed limits if not optimized.
* **Offline edge nodes** → Edge devices may lose connection with the central Granite-powered backend.

**User Experience (UI/UX)**

* **Complex dashboards** for officials may overwhelm non-technical staff if not simplified.
* **Voice assistant issues** → Accent/language recognition (via STT) may misinterpret queries.
* **Mobile responsiveness** → Some charts/maps may not render properly on smaller screens.

**Testing & Reliability**

* **Mocking Granite** → Hard to fully simulate LLM responses in tests (may break snapshots).
* **Load testing limits** → Free/standard IBM Cloud plans may throttle Granite API calls.
* **Version drift** → IBM Granite models may update, changing response style/quality unexpectedly.

**Governance & Compliance**

**Data residency** → Some cities may require IoT/citizen data to stay within local jurisdiction (GDPR, CCPA).

**Explainability** → Granite LLM outputs need traceability (why was this route recommended?).

**Accessibility** → Multilingual + ADA compliance may be partially covered but needs custom tuning.

**12)Future Enhancement**

Future enhancements in artificial intelligence will focus on increased integration into daily life, greater automation of complex tasks, and significant advancements in machine learning algorithms and natural language processing. Key areas of development include accelerating scientific discovery through big data analysis, creating more human-like AI systems for better decision-making, and expanding AI applications in fields like personalized healthcare, autonomous transportation, and climate change solutions. However, these advancements also raise ethical concerns around job displacement, privacy, and algorithmic bias, necessitating strong regulations for responsible AI deployment.

**Technological Advancements**

**Sophisticated Machine Learning:**

AI systems will become more adept at learning and decision-making, powered by increasingly complex algorithms and architectures.

**Advanced Natural Language Processing (NLP):**

Future NLP will enable more intuitive and natural interactions with AI, leading to intelligent chatbots and virtual assistants that can comprehend and respond to human language more effectively.

Routine and repetitive tasks will be further automated, freeing humans for more creative and strategic work, leading to significant productivity gains across various industries Quantum Computing Integration:

Quantum AI could overcome computational limitations, making previously unsolvable problems feasible, such as complex material simulations and vast data analysis in scientific research.

**Societal and Industry Impact**

**Enhanced Automation:**

**Personalized Services:**

AI will enable highly personalized experiences, from product recommendations in e-commerce to customized learning plans in education and tailored preventative care in healthcare.

**Revolutionized Healthcare:**

AI will improve diagnostic accuracy, enhance patient care, and increase access to healthcare, particularly for underrepresented groups.

**Solutions to Global Challenges:**

AI is expected to play a critical role in tackling complex global issues, such as climate change, by analyzing vast datasets for better prediction and understanding of climate models.

**Ethical and Regulatory Considerations**

**Job Displacement and Creation:**

While some jobs may be automated, new opportunities are expected to emerge in areas like data analysis and machine learning, though responsible management of the transition is crucial.

**Data Privacy and Bias:**

The pervasive nature of AI in healthcare and other sectors raises significant ethical questions regarding data privacy and the potential for bias in algorithms, requiring robust regulatory frameworks.

**Responsible AI Deployment:**

The development and integration of AI must be guided by strong ethical principles and comprehensive regulations to ensure its safe, fair, and beneficial use.