Sustainable Smart City Assistant Using Ibm Granite Llm

*Team ID: LTVIP2025TMID32104*

**1. Introduction**

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

**Team Members:**

* Team Leader: Kandula Bhargavi
* Team Member:K Hemanth
* Team Member: K Santhosh Mani
* Team Member: Jaya Prakash

**2. Project Overview**

**Purpose:**  
To support urban sustainability and citizen engagement by providing an AI-powered assistant that can summarize policies, analyze KPIs, detect anomalies, and respond to citizen feedback in real-time.

**Features:**

* AI-powered sustainability chat assistant
* Real-time dashboard for city KPIs (Air, Water, Energy, Waste)
* AI-generated insights from KPI trends
* Professional AI-enhanced sustainability reports
* Sidebar-based navigation for intuitive UX
* Visually themed dark UI with glowing highlights and metrics

**3. Architecture**

**Frontend:**

* Built in **Streamlit** with enhanced CSS for dark mode, gradients, glowing text, and UI polish
* Sidebar navigation using streamlit-option-menu
* Pages: Home, Ask Assistant, City Dashboard, Reports, About

**Backend Integration:**

* **LangChain** to structure LLM prompts and run chains
* **IBM Watsonx Granite 13B Instruct V2** model via langchain\_ibm

**LLM Configuration:**

* decoding\_method: greedy
* max\_new\_tokens: 500
* temperature: 0.7

**4. Setup Instructions**

**Prerequisites:**

* Python 3.8+
* IBM Watsonx Cloud credentials
* Streamlit
* LangChain
* langchain-ibm, python-dotenv

**Installation Steps:**

1. Clone the repo
2. Create virtual environment: python -m venv venv
3. Activate: source venv/bin/activate (Linux/Mac) or venv\Scripts\activate (Windows)
4. Install dependencies: pip install -r requirements.txt
5. Create .env:
6. WATSONX\_URL=your\_url
7. WATSONX\_APIKEY=your\_key
8. WATSONX\_PROJECT\_ID=your\_project\_id
9. Run app: streamlit run smart\_dashboard.py

**5. Folder Structure**

**Frontend:**

* smart\_dashboard.py – Entry point and router for all modules
* Inline markdown style blocks for full dark-themed UI

**Backend (within Streamlit app):**

* WatsonxLLM object from langchain\_ibm
* LLMChain for structured responses
* PromptTemplates for each use case: chat, dashboard insights, reports

**6. Running the Application**

streamlit run smart\_dashboard.py

**7. API Documentation (Internal Only)**

**Integrated Functions:**

* Ask Assistant: AI Q&A on sustainability (e.g., “How can citizens contribute?”)
* City Dashboard: Metric cards + AI insights (air quality, water use, energy, waste)
* Generate Report: Date-range-based summary report + AI-enhanced version

All calls routed via LangChain and processed through IBM Watsonx.

**8. Authentication**

* .env stores credentials securely
* Variables used with python-dotenv:
  + WATSONX\_URL
  + WATSONX\_APIKEY
  + WATSONX\_PROJECT\_ID

**9. User Interface**

**Home Page:**

* Welcome screen introducing project goals and assistant features

**Ask Assistant Page:**

* Text input + response box for sustainability-related queries
* Styled "Ask Now" button

**City Dashboard:**

* KPI cards for Air Quality, Electricity, Water, Waste
* Weekly trend chart
* AI-generated insights on city performance metrics

**Reports Page:**

* Date pickers for custom range
* Static summary + AI-enhanced professional report

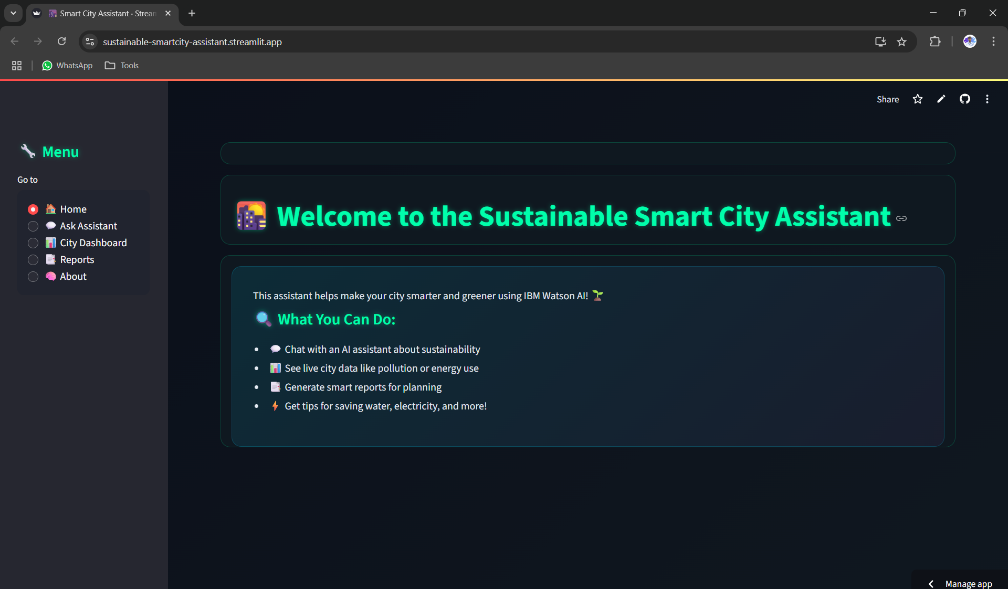
**Design Highlights:**

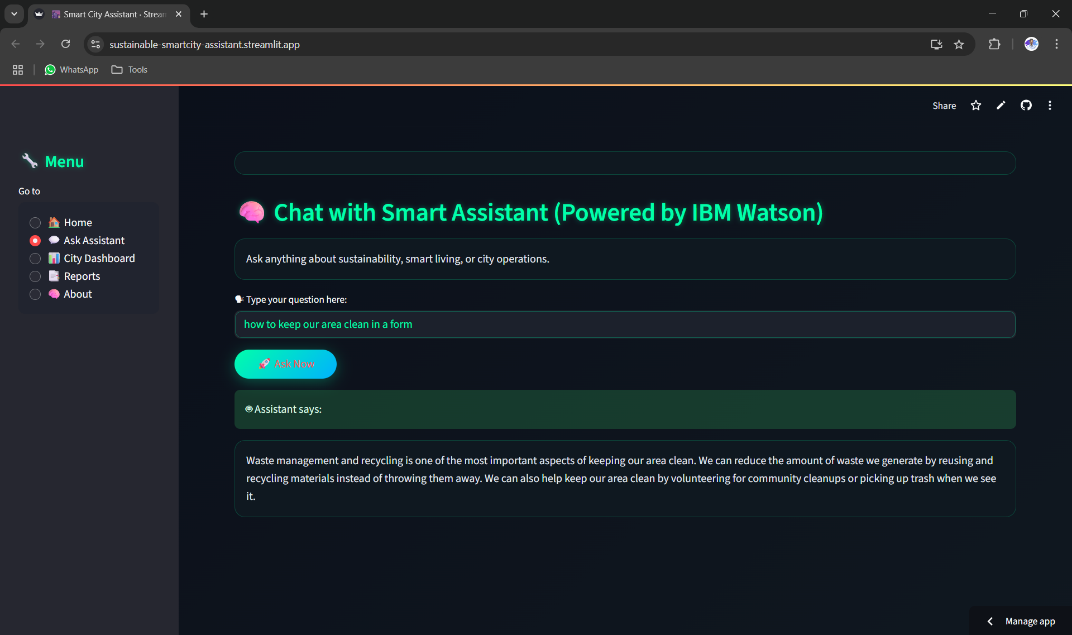
* Neon glow UI (green & cyan)
* Gradient cards & metrics
* Responsive layout & real-time feedback

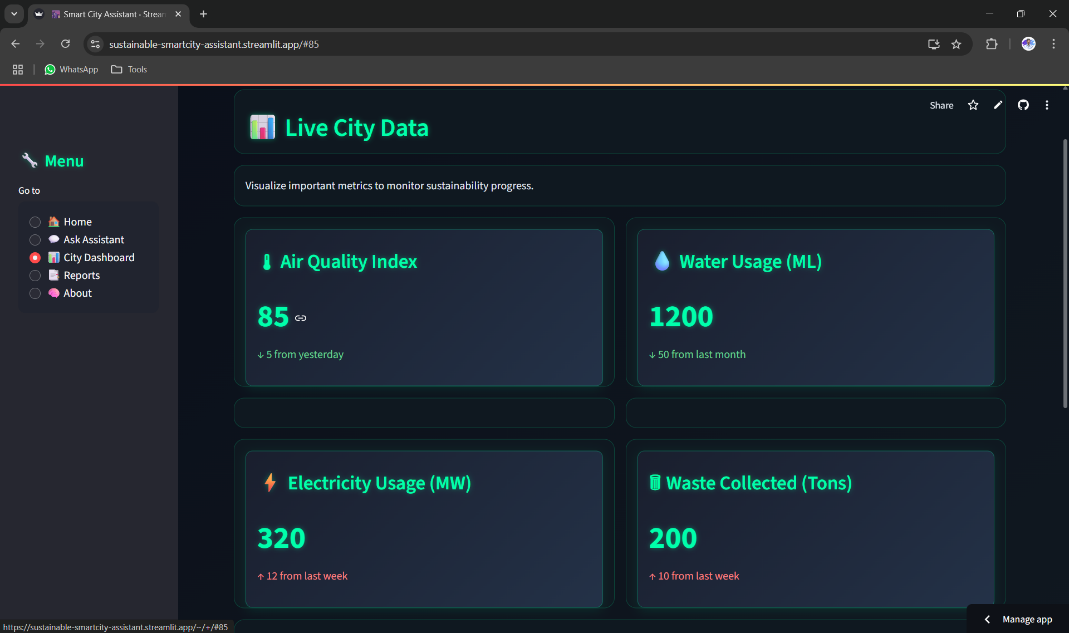
**10. Testing**

* End-to-end testing of all interaction flows
* Validated model responses for accuracy and relevance
* Checked for empty inputs, invalid dates, and loading failures

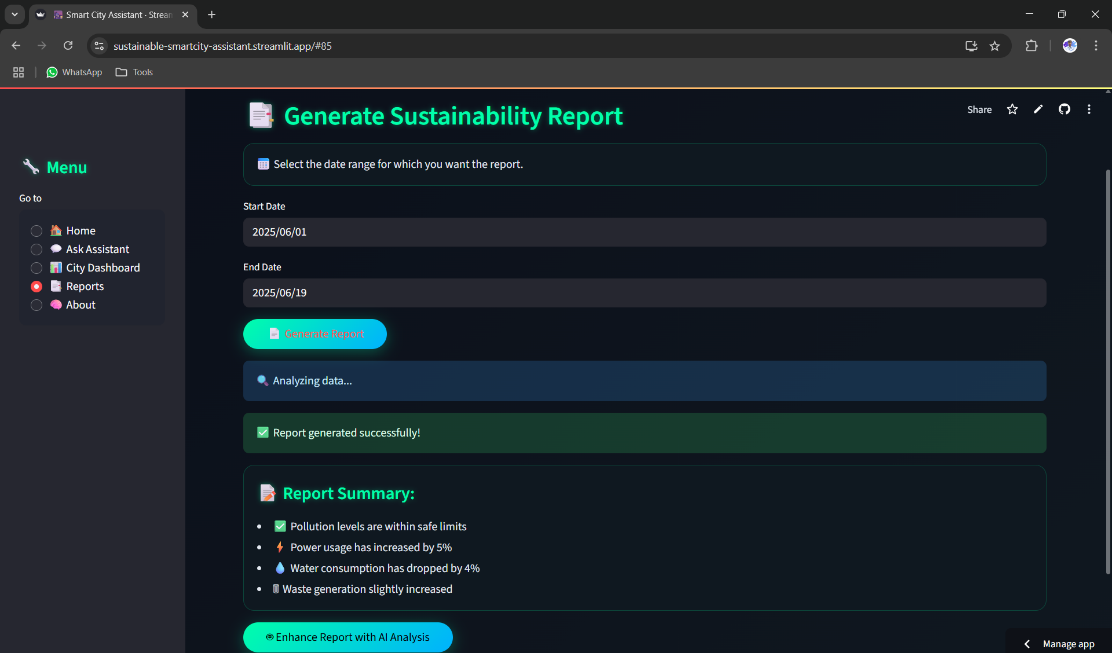
**11. Screenshots or Demo**

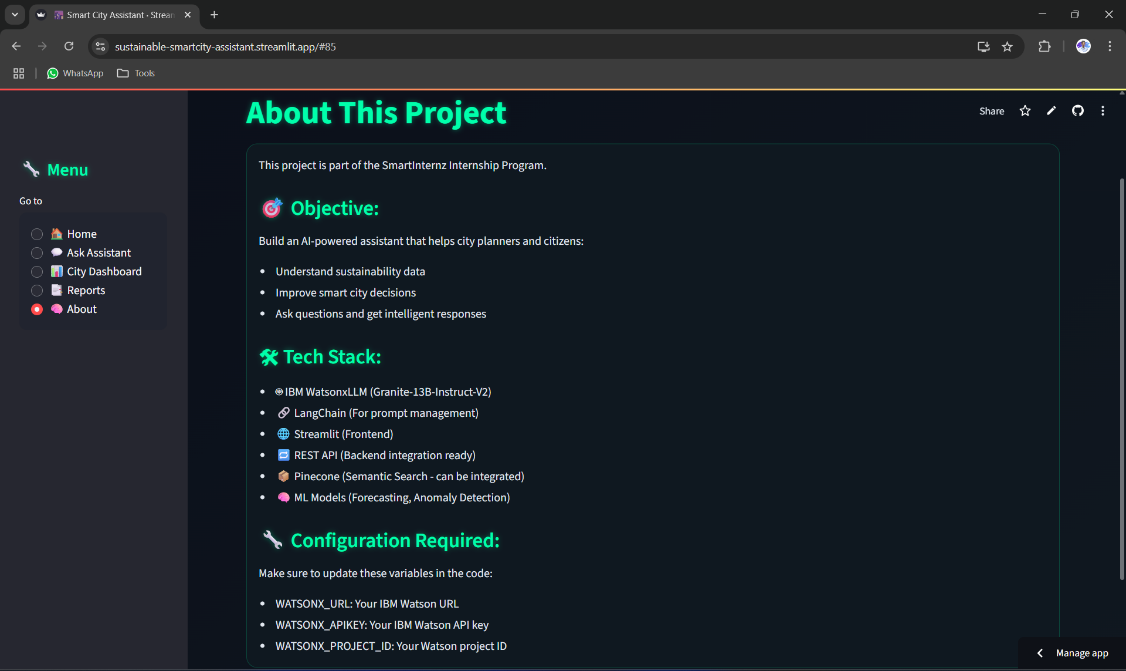












**12. Known Issues**

* Lacks database for persistent history/logs
* Dependent on Watsonx API limits (rate/quota)
* Current insights use static metrics (future version to support live API/KPI data)

**13. Future Enhancements**

* Integrate real-time sensor data APIs for city KPIs
* Add Pinecone-based semantic search for document Q&A
* Implement user accounts and login sessions
* Extend charts with historical KPI uploads via CSV
* Deploy via Docker, add CI/CD pipelines for DevOps
* Use dynamic LLM context for better response relevance