**Sustainable Smart City**

**Assistant Using IBM Granite LLM**

# 1.Introduction

**Project title : Sustainable Smart City Assistant Using IBM Granite LLM**

**Team member : N.Nishu Kumari**

* **Team member : S.Nivetha**
* **Team member : A.Nivetha**
* **Team member : Nasree Begum**

**The Sustainable Smart City Assistant** is like a friendly guide that helps cities and people live in a smarter and greener way. It connects technology, people, and government to make urban life better, cleaner, and easier to manage**.**

**Purpose of the Assistant:**

⭐ **For Citizens (People Living in the City):**

* Gives **simple eco-friendly tips** for daily life.
* Helps people **save energy, water, and reduce waste**.
* Shares **personalized reminders** like turning off unused lights, using public transport, or reducing plastic use.
* Connects people with **local green services** (e.g., recycling centers, water-saving tools, or solar programs).
* Encourages **community participation** by promoting local events, awareness drives, and green activities.

⭐ **For City Officials (Leaders and Planners):**

* Acts as a **decision-making partner**.
* Provides **clear insights** from real-time city data (traffic, pollution, energy use).
* Uses AI to **forecast future needs**—like water shortage, traffic congestion, or waste growth.
* Summarizes **complex reports and policies** into easy-to-understand points.
* Helps in **strategic planning** for a sustainable future.

**KEY BENEFITS**:

⭐ **For the Environment:**

* Reduces unnecessary energy use.
* Promotes **clean water management**.
* Improves **waste handling and recycling**.
* Supports eco-friendly **transportation solutions**.

⭐ **For People:**

* Makes city life more **comfortable and safe**.
* Saves money through **energy-efficient living**.
* Increases awareness of **green practices**.
* Builds a sense of **community belonging**.

⭐ **For the City:**

* Creates a **modern, eco-conscious image**.
* Attracts investments and smart development.
* Ensures the city is **inclusive, resilient, and future-ready**.

**VISION OF THE AI ASSITANT**

The Sustainable Smart City Assistant is **not just a tool**.  
It is like a **bridge** between:

* 🌐 **Technology** (AI, sensors, data)
* 🏛️ **Governance** (city officials, policies)
* 👨‍👩‍👧 **Community** (citizens and their lifestyle)

👉 Together, these create a **smart, green, and connected city** where people can grow while caring for nature.

## **CONVERSATIONAL INTERFACE**

✨ **Key Point:**

* Talk to the system in **natural language** (just like chatting with a friend).

✨ **Why it matters:**

* No need for technical skills.
* Easy for **citizens** and **city officials** to use.
* Makes information **clear, quick, and friendly**.

**What It Can Do (Functionality)**

⭐ Citizens can:

* Ask simple questions like “Where is the nearest recycling center?”
* Get daily **eco-friendly tips**.
* Receive **updates on city services**.

⭐ City officials can:

* Ask for **data insights** in plain words.
* Get **real-time updates** on traffic, water, or energy use.
* Receive **guidance for planning** in an easy-to-read way.

# SMARTER POLICIES FOR EVERYONE

✨ **Policy Summarization**

* Government documents are often **long and difficult**.
* The assistant makes them **short and clear**.
* Citizens and officials can quickly **understand rules and take action**.

✨ **Resource Forecasting**

* Uses **past and present data** to predict the future.
* Estimates **energy, water, and waste needs**.
* Helps cities prepare before problems happen.

**Green Living Made Easy**

✨ **Eco-Tip Generator**

* Shares **personalized advice** for daily life.
* Example: “Switch off unused appliances to save energy.”
* Encourages small actions that create **big positive impact**.

✨ **Citizen Feedback Loop**

* Collects **opinions from people**.
* Shares them with officials for better planning.
* Builds **trust and teamwork** between city and community.

**Smarter City Decisions**

✨ **KPI Forecasting**

* Cities track **important goals** like energy savings, clean air, or waste reduction.
* Assistant predicts progress in advance.
* Officials can **adjust plans early** to stay on track.

✨ **Anomaly Detection**

* Finds **unusual patterns** in city data.
* Example: sudden spike in water usage = possible leak.
* Works like an **early warning system** for quick fixes.

**Easy Data Handling**

✨ **Multimodal Input Support**

* Not just text! The assistant also works with **PDFs, CSVs, and reports**.
* Makes it simple to analyze **different kinds of documents**.
* Saves time for both citizens and officials.

**User-Friendly Design**

✨ **Streamlit or Gradio UI**

* Provides a **clean and easy-to-use dashboard**.
* Citizens can see tips, updates, and alerts.
* Officials can view reports, forecasts, and trends.
* No complex setup—just **simple interaction**.

**Why It Matters**

* Citizens get **helpful tips, clear policies, and easy access** to services.
* Officials get **data insights, predictions, and planning tools**.
* The city becomes more **eco-friendly, efficient, and people-centered**.
* Together, it creates a **smarter, greener, and happier city**.

# ****Architecture – Frontend (Streamlit)****

✨ **What it is:**

* The **frontend** is the face of the system, where **citizens and officials interact**.
* Built using **Streamlit**, which makes it simple to create web apps with **interactive elements**.

✨ **Key Features in the UI:**

* 📊 **Dashboards** → Show live updates like energy usage, water levels, or waste management.
* 📂 **File Uploads** → Citizens or officials can upload PDFs, CSVs, or reports for quick analysis.
* 💬 **Chat Interface** → Ask questions in **plain language** and get real-time answers.
* 📝 **Feedback Forms** → Citizens can share opinions, problems, or suggestions easily.
* 📑 **Report Viewer** → View summarized policies or detailed reports in a simple format.

✨ **Navigation:**

* A **sidebar menu** helps users move smoothly between sections.
* Uses the **streamlit-option-menu** library to keep navigation **organized and user-friendly**.

✨ **Scalability & Flexibility:**

* Each feature (dashboard, chat, reports, etc.) is built as a **separate module**.
* This makes the system easy to **update, expand, or customize** for different cities.
* More pages and tools can be added in the future without breaking the system.

✨ **Why Streamlit?**

* Fast development with **Python-friendly tools**.
* Supports **real-time updates**.
* Gives both **citizens and officials** a smooth, modern experience.

# ****Architecture – Backend (FastAPI)****

✨ **What it is:**

* **FastAPI** is the **backend engine** of the assistant.
* It creates **REST API endpoints** that connect the frontend with AI models and databases.
* Works like a **bridge** between what the user asks and how the system responds.

✨ **Key Functions of FastAPI:**

* 📑 **Document Processing** → Reads and simplifies PDFs, CSVs, and reports.
* 💬 **Chat Interactions** → Handles questions from users and gives AI-powered replies.
* 🌱 **Eco-Tip Generator** → Suggests personalized sustainability tips.
* 📊 **Report Creation** → Generates summaries, dashboards, and easy-to-read outputs.
* 🧩 **Vector Embedding** → Converts text into smart data for fast search and retrieval.
* ⚡ **Async Performance** → Runs tasks quickly without slowing down the system.
* 📖 **Swagger Integration** → Provides auto-generated API docs for developers.

✨ **Why FastAPI?**

* Super **fast and lightweight**.
* Easy to **scale** as city data grows.
* Built-in support for **real-time responses**.
* Developer-friendly → Easy testing with Swagger UI.

**Simple Architecture Diagram**

Here’s a **basic diagram** you can use in your project (described so you can paste/replicate in PPT/Docs):

👩‍💻 **Citizens / Officials**

**│**

**▼**

**🌐 Frontend (Streamlit UI)**

**────────────────────────────────**

**📊 Dashboards | 📂 Uploads**

**💬 Chat | 📝 Feedback**

**────────────────────────────────**

**│ API Calls**

**▼**

**⚙️ Backend (FastAPI)**

**────────────────────────────────**

**📑 Document Processing**

**💬 Chat Engine (AI/NLP)**

**🌱 Eco-Tip Generator**

**📊 Reports & Forecasting**

**🧩 Vector Embedding**

**────────────────────────────────**

**│**

**▼**

**🗄️ Databases + AI Models**

**(Real-time Data, Policies, Sensors)**

👉 This shows a **clear flow**:

* Users interact with the **Frontend (Streamlit)**
* Requests go to the **Backend (FastAPI)**
* FastAPI uses **AI + Databases** to process and return result
* **LLM Integration (IBM Watsonx Granite)**

✨ The **Granite LLM models from IBM Watsonx** are the **intelligent engine** behind the Sustainable Smart City Assistant.  
They enable the system to **understand natural language** and **generate meaningful outputs** for both citizens and city officials.

### 🌟 Key Functions:

* 📝 **Policy Summarization** – Breaks down long, technical government policies into **easy-to-read summaries** that anyone can understand.
* 🌱 **Eco-Tip Generator** – Provides **personalized green tips** based on user behavior, such as using public transport or reducing water wastage.
* 📊 **Report Writing** – Automatically creates **structured sustainability reports** that highlight energy savings, waste reduction, and water management.
* 💬 **Conversational Interface** – Lets people **chat with the system naturally** without needing technical knowledge.
* 🔮 **Forecast Guidance** – Helps city officials by **explaining predictions** about future needs in simple words.

**Why IBM Watsonx Granite?**

* ✅ **Advanced Language Skills** – Understands **context, tone, and intent** to give accurate and human-like responses.
* ✅ **Scalability** – Can handle everything from **short queries** to **large reports** with the same efficiency.
* ✅ **Customizable Prompts** – Designed to **adapt responses** to different use cases (citizens, officials, or policy documents).
* ✅ **Trusted Technology** – Built by IBM, ensuring **reliability, transparency, and security**.
* ✅ **Seamless Integration** – Works smoothly with **FastAPI (backend)** and **Streamlit (frontend)** for a complete system.

**Additional Benefits:**

* 🌍 **Supports Sustainability Goals** → Helps promote eco-friendly behavior among citizens.
* 📖 **Bridges the Knowledge Gap** → Makes complex information simple for non-technical users.
* ⏱️ **Saves Time** → Cuts hours of reading policies or analyzing reports into a few seconds.
* 🏛️ **Policy Decision Support** → Gives city officials **clear insights** for planning better strategies.

**Vector Search (Pinecone) – In Our Project**

In a Sustainable Smart City, policies and reports are often long, technical, and difficult to understand. Citizens and even officials may struggle to find the exact information they need.

This is where Vector Search with Pinecone plays an important role. It works like a smart librarian inside our assistant, ensuring that users can quickly find the right answers using natural language instead of complex keywords.

**How It Works in Our Assistant**

* **All uploaded policy documents** on energy, water, waste, and other areas are converted into vectors using Sentence Transformers.
* These vectors are stored in Pinecone, a specialized database designed for fast vector search.
* When a user asks a question in plain English or their local language, the system looks for the meaning of the question, not just the words.
* Using cosine similarity, Pinecone finds the closest and most relevant passages from the stored policies and reports.
* The results are then passed to IBM Granite LLM, which summarizes the content and presents it in short, clear, and human-friendly language.

**Why It’s Important for a Smart City**

* Bridges the gap between citizens and complex policies by making them easier to access.
* Saves valuable time for both citizens and officials who no longer need to search manually through large documents.
* Supports sustainability by giving quick access to rules, subsidies, and eco-policies that encourage greener lifestyles.
* Scalable and reliable, capable of handling thousands of documents without losing speed.

**Example in Our Project**

* A citizen asks: “What are the rules about rainwater harvesting in housing societies?”  
  → Pinecone locates the exact section of the water policy.  
  → IBM Granite converts it into a short and simple answer.
* An official asks: “Show me all policies related to plastic waste reduction.”  
  → Pinecone searches across multiple documents and provides the most relevant results instantly.

In our Sustainable Smart City Assistant, Pinecone ensures the system is not just a chatbot, but a policy-aware guide. It helps citizens live greener and enables city leaders to plan smarter with quick and reliable information.

**ML Modules (Forecasting and Anomaly Detection)**

In a growing city, challenges like **energy shortages, water leaks, and unexpected waste spikes** can cause serious problems if not managed in time. That’s where the **ML modules** in our Sustainable Smart City Assistant come in. They act like the city’s **smart guardians**, keeping an eye on data and predicting what might happen next.

**Forecasting – Looking Ahead**

* Just like how we plan groceries for the week by checking past usage, our system looks at **old data** (like energy, water, and waste records).
* Using this, it can **predict future needs**—for example:
  + How much water will be required during summer?
  + Will energy demand rise during festivals?
* These predictions are shown in **easy-to-read graphs and charts**, so officials don’t have to dig through numbers.
* With this, the city can **prepare in advance**, avoiding shortages or overuse of resources.

**Anomaly Detection – Spotting the Unusual**

* Imagine if suddenly a neighborhood uses **10x more water overnight**. That could mean a **leak or pipe burst**.
* Or if waste levels in one area drop suddenly, it could mean **collection trucks missed the route**.
* The anomaly detection system catches these **unusual patterns** in real time.
* This works like an **early warning system**, alerting officials before small issues become big problems.

Why This Matters for a Smart City

* **Forecasting** = helps the city **plan ahead smartly**.
* **Anomaly detection** = ensures **quick action on problems**.
* Together, they make the assistant **proactive, reliable, and citizen-friendly**.
* The result: a city that is **efficient, sustainable, and resilient**—serving both its people and the environment.

Bottom of Form

Before bringing the Sustainable Smart City Assistant to life, we need to set up a strong foundation. These prerequisites act like the **tools, passes, and connections** that make sure everything runs without errors.

**Prerequisites**

**Python 3.9 or later**  
Python is the **core engine** of this project. All modules, from data handling to AI communication, are built in Python. Using version 3.9 or above ensures that modern libraries (like Scikit-learn, Streamlit, and Transformers) run smoothly.

* **pip and virtual environment tools**  
  pip is our **installer tool**, helping us bring in external libraries quickly. Along with this, a virtual environment makes sure our project has its **own clean workspace**. This avoids conflicts with other Python projects and keeps everything organized.
* **API keys for IBM Watsonx and Pinecone**
  + **IBM Watsonx (Granite LLMs)** → These keys unlock access to the AI brain of our assistant. With this, the system can summarize policies, generate eco-tips, and chat naturally with citizens.
  + **Pinecone** → This key connects us to the vector database, allowing smart semantic searches across uploaded documents. Instead of scanning blindly, users can ask questions in plain language and get precise answers.  
    Together, these keys are like **secure ID cards** that grant entry into powerful AI services.
* **Internet Access**  
  Since our project relies heavily on **cloud services and real-time data**, a stable internet connection is non-negotiable. Without it, the assistant cannot communicate with Watsonx or Pinecone, and features like forecasting, anomaly detection, or chat would not function.

### Why These Prerequisites Matter

* Python provides the **backbone**.
* pip and virtual environments give us a **safe workspace**.
* API keys open the doors to **AI and smart search**.
* Internet access ties everything together, keeping the assistant connected to live data.

With these four pillars ready, the Sustainable Smart City Assistant becomes capable of **learning, predicting, guiding, and helping both citizens and officials in real time**.

**Installation Process**

To get the Sustainable Smart City Assistant up and running, follow these simple steps:

**1. Clone the Repository**

* First, we bring the project files from the online repository (e.g., GitHub) to your local computer.
* This creates a copy of the assistant’s code so you can run and customize it.

**2. Install Dependencies from requirements.txt**

* Every project depends on external libraries (like **Streamlit, FastAPI, Scikit-learn, Pinecone, Watsonx SDK**).
* The requirements.txt file is like a shopping list of all these libraries.
* Run the command to install everything at once, ensuring your system has all the tools the project needs.

**3. Create a .env File and Configure Credentials**

* The .env file securely stores your **API keys** (for IBM Watsonx Granite and Pinecone).
* Think of it as a private **locker of passwords** that your project can use without exposing them publicly.
* Add your keys in this file so the assistant can connect with AI models and vector databases.

**4. Run the Backend Server using FastAPI**

* Start the **FastAPI backend**, which is the **engine room** of the assistant.
* It powers API endpoints for:
  + Document processing
  + Chat interactions
  + Forecasting
  + Eco-tip generation
* Once the backend is running, it becomes the “brain” that responds to requests.

**5. Launch the Frontend via Streamlit**

* Next, open the **Streamlit frontend**, which is the **face of the assistant**.
* This is where users interact with dashboards, upload files, ask questions, and provide feedback.
* Streamlit ensures the interface is simple and user-friendly.

**6. Upload Data and Interact with Modules**

* Finally, upload policy documents, CSVs, or sensor data.
* The assistant will:
  + Summarize long documents into easy points.
  + Predict future resource usage.
  + Detect unusual activity in the data.
  + Suggest eco-friendly actions.
* At this stage, the system is **fully functional**, ready to help citizens and city officials.

**Folder Structure**

The project is organized into a clean folder structure so that every component has its own place. This makes the assistant **easier to maintain, scale, and understand**. Here’s how it is structured:

### app/

* This is the **core backend folder**, powered by **FastAPI**.
* It contains all the main logic such as routers, models, and integration modules.
* Think of it as the **engine room** where most of the backend operations happen.

### app/api/

* A **subdirectory** inside app/ dedicated to **API routes**.
* Each feature (like chat, feedback, reports, or document vectorization) has its own modular route.
* This makes the code **clean and modular**, so new routes can be added without disturbing existing ones.

### ui/

* This is the **frontend folder** where all the **Streamlit pages** live.
* Contains layouts such as dashboards, cards, and forms that citizens and officials interact with.
* Acts as the **face of the assistant**, ensuring the experience is smooth and user-friendly.

### smart\_dashboard.py

* The **entry script** for launching the Streamlit dashboard.
* Acts as the **starting point** of the frontend, bringing together different pages (like data uploads, eco-tips, and reports) into one interactive interface.

### granite\_llm.py

* Handles communication with the **IBM Watsonx Granite LLM**.
* Manages tasks such as **summarization, natural language chat, and eco-tip generation**.
* This is where the **AI brain** of the assistant connects with the system.

### document\_embedder.py

* Converts uploaded documents into **vector embeddings** using sentence transformers.
* Stores these embeddings in **Pinecone**, making semantic search possible.
* Enables users to ask **natural language questions** and get accurate answers from large documents.

### kpi\_file\_forecaster.py

* Responsible for **forecasting trends** like energy usage, water demand, and waste levels.
* Uses regression models to make predictions from historical and real-time data.
* Supports officials in **strategic planning** by showing what the future might look like.

### anomaly\_file\_checker.py

* Scans uploaded KPI data for **unusual or abnormal patterns**.
* Helps detect problems like water leaks, energy spikes, or waste mismanagement before they grow bigger.
* Works like an **early warning system** for city resources.

### report\_generator.py

* Creates **AI-generated sustainability reports** using insights from Watsonx Granite.
* These reports summarize resource use, forecasts, and eco-tips into **easy-to-read documents**.
* Makes complex data understandable for both officials and citizens.

# Running the Application

Getting the Sustainable Smart City Assistant up and running is simple and intuitive. Here’s how it works:

* **Start the Backend (FastAPI)** – This is the engine of the system. It handles all the heavy work, like processing documents, generating eco-tips, managing chats, and creating reports. Once it’s running, the system is ready to respond to requests.
* **Launch the Frontend (Streamlit)** – This is the interface where users interact with the assistant. Dashboards, forms, file uploads, chat windows, and report viewers are all available in a clean, easy-to-navigate layout.
* **Navigate the Interface** – The sidebar menu allows you to move between pages seamlessly. Users can explore dashboards, upload data, interact with the AI assistant, or provide feedback without any confusion.
* **Upload and Interact** – You can upload policy documents or CSV data. The system automatically processes this information to provide summaries, predictions, or detect unusual patterns.
* **Real-Time Feedback** – All actions are instant. The frontend communicates with the backend so that summaries, reports, and tips are updated dynamically as soon as data is processed.

Frontend (Streamlit)

The frontend is designed to be **friendly and intuitive**:

* Streamlit powers the web interface, making it simple to display dashboards, upload forms, chat interfaces, and reports.
* The sidebar menu guides users across different sections effortlessly.
* Each page is **modular**, which means new features can easily be added later.
* Citizens and officials can use the system without technical knowledge, making it **accessible to everyone**.

# Backend (FastAPI)

The backend acts as the **brain of the assistant**:

* FastAPI handles all API requests for chat, document processing, eco-tip generation, report creation, and vector-based semantic search.
* Optimized for **speed and efficiency**, ensuring that the system can process multiple requests at once.
* Comes with **Swagger UI**, which makes it easy to inspect and test all endpoints during development.

# API Documentation

The system includes several ready-to-use APIs:

* **POST /chat/ask** – Ask a question, and the AI responds naturally.
* **POST /upload-doc** – Upload policy documents and store them in Pinecone for smart searches.
* **GET /search-docs** – Search through policies using natural language, not just keywords.
* **GET /get-eco-tips** – Get sustainability tips for energy, water, or waste.
* **POST /submit-feedback** – Record citizen feedback for analysis and future improvements.

All APIs are **documented in Swagger UI**, so testing and understanding them is straightforward.

# Authentication

Currently, the project runs in an **open demonstration environment**. For a real deployment:

* Token-based authentication (JWT or API keys) can secure endpoints.
* OAuth2 with IBM Cloud credentials provides **safe cloud access**.
* Role-based access ensures different users (admin, citizen, researcher) have the right permissions.
* Future improvements may include **user sessions** and **interaction history tracking**.

# 9. User Interface

The interface of the Sustainable Smart City Assistant is designed to be **minimalist, functional, and friendly** for non-technical users. Its goal is to make interaction **intuitive and efficient**:

* **Sidebar Navigation** – Quickly switch between dashboards, chat, eco tips, forecasting, and reports.
* **KPI Visualizations with Summary Cards** – Key performance indicators like energy, water, and waste trends are displayed in clear cards for easy understanding.
* **Tabbed Layouts** – Different tabs for chat, eco tips, and forecasting make information **well-organized and accessible**.
* **Real-Time Form Handling** – Users can submit data or feedback instantly, with results reflected immediately.
* **PDF Report Download** – Generated sustainability reports can be downloaded for offline review.

The design emphasizes **clarity, speed, and guidance**, with help texts and intuitive flows to assist first-time users and ensure everyone can navigate easily.

# 10. Testing

Testing was performed in **multiple stages** to ensure reliability, accuracy, and robustness:

* **Unit Testing** – Each function, especially prompt-engineering scripts and utilities, was tested independently.
* **API Testing** – All backend endpoints were tested via Swagger UI, Postman, and automated test scripts.
* **Manual Testing** – File uploads, chat interactions, and outputs like reports, summaries, and forecasts were checked manually.
* **Edge Case Handling** – Special cases like malformed inputs, very large files, or invalid API keys were tested to ensure the system handles them gracefully.

Every function was **validated for both offline and online modes**, guaranteeing smooth operation whether connected to APIs or running locally.

# 11. Screenshots

The project includes **screenshots** demonstrating:

* Dashboards with KPI visualizations
* Chat assistant interactions
* Eco-tip suggestions
* Forecasting graphs and anomaly detection results
* PDF report downloads

These visuals help **illustrate the functionality** and guide users on how to use the system effectively.

# 12. Known Issues

While the project is stable, some limitations exist:

* Large document uploads may slow down processing slightly.
* Real-time updates may experience minor delays with very large datasets.
* Some edge-case queries may return less accurate summaries until prompts are fine-tuned further.
* Role-based access and advanced authentication are not fully implemented yet.

These issues are being monitored and **planned for resolution** in future versions.

# 13. Future Enhancements

Planned improvements for upcoming releases include:

* **Advanced User Authentication** – JWT, OAuth2, and role-based access for secure deployment.
* **User Sessions and History Tracking** – Keep track of previous queries, uploads, and report downloads.
* **Interactive Dashboards** – More dynamic visualizations for forecasting and KPIs.
* **Expanded AI Capabilities** – Improved prompt engineering and multilingual support for global cities.
* **Mobile Interface** – Optimized UI for smartphones and tablets to increase accessibility.
* **Integration with IoT Sensors** – Real-time city data collection for smarter predictions and anomaly detection.

**CONCLUSION:**

A **Sustainable Smart City** is more than just buildings and roads.  
It is about **living in harmony with the environment** while using technology wisely.  
The **Assistant** ensures that both citizens and city leaders:

* **Think green** 🌿
* **Act smart** ⚡
* **Live better** 🏙️