

1. Introduction

- **Project Title: Sustainable Smart City Assistant Using IBM Granite LLM**
 - **Team Members:**
 - Sanjay R
 - Rasool khan R
 - Wilson J
 - Shanmuganathan N
-

2. Project Overview

- **Purpose**

The **Sustainable Smart City Assistant** is an AI-powered system built to help cities become more eco-friendly, efficient, and citizen-centric. By leveraging **IBM Watsonx Granite LLM**, the assistant enables both city officials and residents to interact with urban data and policies using natural language. It supports strategic planning, policy summarization, anomaly detection, and personalized eco-advice.

The goal is to bridge **technology, governance, and citizen engagement** to create smarter and more sustainable cities.

- **Features**

Feature	Key Point	Functionality
Conversational Assistant	Natural Language Interaction	Citizens and officials can chat with the system and get real-time guidance
Policy Summarization	Easy Policy Understanding	Converts lengthy government documents into short, actionable insights
Resource Forecasting	Predictive Analytics	Forecasts energy, water, and waste usage using historical data
Eco-Tip Generator	Sustainability Advice	Offers personalized tips to reduce environmental impact
Citizen Feedback Loop	Public Engagement	Gathers and analyzes citizen feedback to support decision-making
KPI Forecasting	Strategic Support	Projects key performance indicators (KPIs) for city departments
Anomaly Detection	Alert System	Detects abnormal patterns in sensor or usage data
Multimodal Input	Data Flexibility	Supports PDF, text, and CSV files for analysis

Feature	Key Point	Functionality
Streamlit Interface	User-Friendly UI	Provides interactive web UI for citizens and city officials

3. System Architecture

- **Frontend – Streamlit**

- Built with **Streamlit** for quick, responsive UIs
- Sidebar navigation using streamlit-option-menu
- Multiple pages: Dashboard, Chat, Uploads, Feedback, Reports

- **Backend – FastAPI**

- RESTful APIs built with **FastAPI**
- Endpoints for chat, summarization, eco tips, anomaly detection, etc.
- Supports asynchronous execution and Swagger UI testing

- **Language Model – IBM Watsonx Granite LLM**

- Handles natural language processing tasks:
 - Summarization
 - Chat responses
 - Report generation
- Custom prompts for sustainability and policy contexts

- **Vector Search – Pinecone + Sentence Transformers**

- Stores and indexes embedded documents
- Enables semantic search using cosine similarity

- **Machine Learning Modules**

- Forecasting and anomaly detection powered by:
 - **Scikit-learn**
 - **Pandas**
 - **Matplotlib**
 - Works with time-series KPI data
-

4. Setup Instructions

- **Prerequisites**

- Python 3.9+
- pip and virtual environment
- IBM Watsonx & Pinecone API keys
- Internet access

• Installation Steps

1. Clone the repository
2. Install packages:
pip install -r requirements.txt
3. Create .env file and add API keys
4. Run backend:
uvicorn app.main:app --reload
5. Start frontend:
streamlit run smart_dashboard.py
6. Use UI to upload files, chat, and access features

5. Folder Structure

sustainable-smart-city/

```
|
| └─ app/                # FastAPI backend
| | └─ api/              # Modular API routes
| |   └─ granite_llm.py   # IBM Granite model logic
| |   └─ document_embedder.py # Vector embeddings and Pinecone
| |   └─ kpi_file_forecaster.py # KPI forecasting
| |   └─ anomaly_file_checker.py # Detects anomalies
| |   └─ report_generator.py # Auto-generates reports
| |
| └─ ui/                 # Streamlit frontend components
|   └─ smart_dashboard.py # Streamlit main file
|   └─ requirements.txt   # Python dependencies
└─ .env                  # API key file (user-created)
```

6. Running the Application

1. **Backend:**
uvicorn app.main:app --reload
 2. **Frontend:**
streamlit run smart_dashboard.py
 3. **Features to Explore:**
 - Chat with the assistant
 - Upload documents and forecast KPIs
 - Download sustainability reports
 - Submit feedback and view eco tips
-

7. API Documentation

Endpoint	Method	Description
/chat/ask	POST	AI response to user query
/upload-doc	POST	Upload policy documents
/search-docs	GET	Semantic search on policies
/get-eco-tips	GET	Return eco-tips by category
/submit-feedback	POST	Save user feedback

All APIs are documented using **Swagger UI** for easy testing.

8. Authentication

Currently, the project runs in **demo mode** with no restrictions.

Planned enhancements for production:

- JWT/API key-based access
 - OAuth2 integration with IBM Cloud
 - Role-based permissions (admin, user, researcher)
 - Session tracking and history logging
-

9. User Interface (UI)

- Built for **non-technical users**
- Pages:
 - Dashboard with summary cards

- Chat assistant
 - Upload files and view results
 - Generate/download reports
 - Intuitive layout with tooltips and tabs
-

10. Testing

• Types of Testing

- **Unit Testing:** Prompt functions, utilities
- **API Testing:** Swagger UI and Postman
- **Manual Testing:** Chat, file uploads, UI responsiveness
- **Edge Cases:**
 - Corrupted files
 - Incorrect data formats
 - Missing API keys

Each module passed verification in both online and offline use cases.

11. Screenshots








(Attach screenshots for:)

- Main dashboard
 - Chat interface
 - KPI forecasts
 - Report generator
 - Feedback form
-

12. Known Issues

- No login/authentication in current version
 - Uploading very large files may cause delays
 - Limited multilingual support
 - Incomplete anomaly tagging for edge datasets
-

13. Future Enhancements

-  Authentication and user roles
-  Multilingual interface for wider adoption
-  Live IoT sensor data integration
-  Geospatial KPI visualization (map-based)
-  Email or SMS alerts for anomalies
-  Admin analytics panel
-  Scheduled reporting and forecasting