Sustainable Smart City Assistant

Project Documentation

Introduction

- Project title : Sustainable Smart City Assistant
- Team Leader : PAWAN KHATRI
- Team member : KHALIQUR RAHMAN.A
- Team member : SADA KIRAN
- Team member : ADITYA KUMAR SINGH.S

Project Overview

• Purpose :

The purpose of a Sustainable Smart City Assistant is to empower cities and their residents to thrive in a more eco-conscious and connected urban environment. By leveraging AI and real-time data, the assistant helps optimize essential resources like energy, water, and waste, while also guiding sustainable behaviors among citizens through personalized tips and services. For city officials, it serves as a decision making partner—offering clear insights, forecasting tools, and summarizations of complex policies to support strategic planning. Ultimately, this assistant bridges technology, governance, and community engagement to foster greener cities that are more efficient, inclusive, and resilient.

• Features:

Conversational Interface

Key Point: Natural language interaction

Functionality: Allows citizens and officials to ask questions, get updates, and receive guidance in plain language

Policy Summarization

Key Point: Simplified policy understanding

Functionality: Converts lengthy government documents into concise, actionable summaries.

Resource Forecasting

Key Point: Predictive analytics

Functionality: Estimates future energy, water, and waste usage using historical and real-time data.

KPI Forecasting

Key Point: Strategic planning support

Functionality: Projects key performance indicators to help officials track progress and plan ahead.

Anomaly Detection

Key Point: Early warning system

Functionality: Identifies unusual patterns in sensor or usage data to flag potential issues.

Multimodal Input Support

Key Point: Flexible data handling

Functionality: Accepts text, PDFs, and CSVs for document analysis and forecasting.

Streamlit or Gradio UI

Key Point: User-friendly interface

Functionality: Provides an intuitive dashboard for both citizens and city officials to interact with the assistant.

Architecture

•Frontend (Stream lit):

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for document processing, chat interactions, eco tip generation, report creation, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

LLM Integration (IBM Watsonx Granite):

Granite LLM models from IBM Watsonx are used for natural language understanding and generation. Prompts are carefully designed to generate summaries, sustainability tips, and reports.

Vector Search (Pinecone):

Uploaded policy documents are embedded using Sentence Transformers and stored in Pinecone. Semantic search is implemented using cosine similarity to allow users to search documents using natural language queries.

ML Modules (Forecasting and Anomaly Detection):

Lightweight ML models are used for forecasting and anomaly detection using Scikit-learn. Time-series data is parsed, modeled, and visualized using pandas and matplotlib.

Setup Instructions

Prerequisites:

• Python 3.9 or later

• pip and virtual environment tools

• API keys for IBM Watsonx and Pinecone

• Internet access to access cloud services

Installation Process:

• Clone the repository

• Install dependencies from requirements.txt

• Create a .env file and configure credentials

• Run the backend server using Fast API

• Launch the frontend via Stream lit

•Upload data and interact with the modules

Folder Structure

app/ – Contains all Fast API backend logic including routers, models, and integration modules.

app/api/ – Subdirectory for modular API routes like chat, feedback, report, and document vectorization.

ui/ – Contains frontend components for Stream lit pages, card layouts, and form UIs.

smart_dashboard.py – Entry script for launching the main Stream lit dashboard.

granite_llm.py – Handles all communication with IBM Watsonx Granite model including summarization and chat.

document_embedder.py – Converts documents to embeddings and stores in Pinecone.

kpi_file_forecaster.py – Forecasts future energy/water trends using regression. anomaly_file_checker.py – Flags unusual values in uploaded KPI data. report_generator.py – Constructs AI-generated sustainability reports.

Running the Application

To start the project:

➢ Launch the FastAPI server to expose backend endpoints.

➢ Run the Streamlit dashboard to access the web interface.

➢ Navigate through pages via the sidebar.

➢ Upload documents or CSVs, interact with the chat assistant, and view outputs like reports, summaries, and predictions.

➢ All interactions are real-time and use backend APIs to dynamically update the frontend.

Frontend (Stream lit):

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for document processing, chat interactions, eco tip generation, report creation, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

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.

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)

• Auth2 with IBM Cloud credentials

• Role-based access (admin, citizen, researcher)

• Planned enhancements include user sessions and history tracking.

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.

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

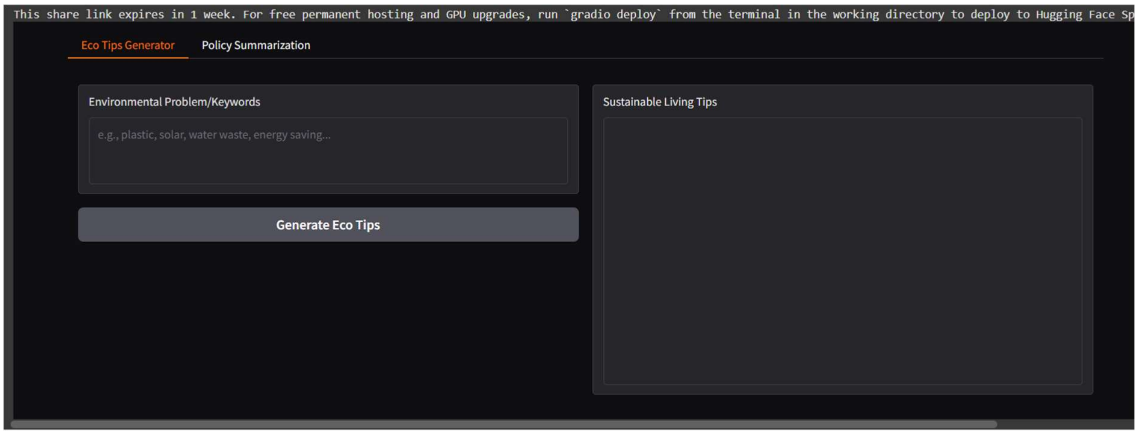
Manual Testing: For file uploads, chat responses, and output consistency

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

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

Screenshot

Output:



Known Issues

Technical Issues:

- Hallucinations and inaccurate responses.
- Difficult integration with IoT, GIS, and real-time data.
- High compute needs and latency

Privacy & Security:

- Risk of surveillance and data misuse
- Vulnerable to cyberattacks.
- Must comply with privacy laws (e.g., GDPR)

Sustainability Concerns:

- LLMs have a high **carbon footprint**.
- Can contradict sustainability goals if not optimized

Ethical & Social Issues:

- Bias in AI decisions.
- Lack of transparency and explainability.
- Risk of digital exclusion.

Data Challenges:

- Real-time data may be missing, outdated, or siloed.
- Lack of data standardization across systems

User Experience:

- Over-reliance on AI.
- Accessibility issues for diverse user groups.

Future enhancement

Future Enhancements:

- **Real-time data integration** (IoT, sensors, traffic, pollution).
- **Green AI deployment** on low-power/edge devices.
- **Explainable AI** for transparent decisions.
- **User personalization** and accessibility support.
- **Multilingual and inclusive interfaces.**
- **Predictive analytics** for traffic, energy, etc.
- **Human-in-the-loop** for oversight and control.
- **Adaptive policy alignment** with city goals.
- **Privacy-focused AI** (e.g., federated learning).
- **Feedback loops** for continuous improvement.

Conclusion

The Sustainable Smart City Assistant demonstrates how advanced technologies like AI, real-time data analytics, and user-friendly interfaces can bridge the gap between governance, citizens, and sustainability goals. By integrating forecasting, anomaly detection, policy summarization, and conversational interfaces, the system empowers cities to manage resources efficiently while engaging citizens in eco-conscious practices.

Although challenges such as data quality, security risks, ethical concerns, and technical limitations remain, the project provides a strong foundation for future innovations. With planned enhancements like real-time IoT integration, explainable AI, multilingual support, and privacy-preserving methods, the assistant has the potential to become a vital tool in creating greener, smarter, and more resilient urban environments.

In essence, this project is not just a technological solution but a step toward a sustainable future where technology, policy, and community come together to shape inclusive and eco-friendly cities.

Sustainable Smart City Assistant Using IBM Granite LLM

Generative AI with IBM

Thank You