Sustainable Smart City

Project Documentation

1. Introduction

Project Title: Sustainable Smart City Assistant using

IBM Cloud and Granite AI

Team Members:

Team Member	Role
Maroju Sai Sahithi (Lead)	Project Lead & Frontend Developer
Madapati Mohana Chandrika	Research & Content Manager
Lukka Sanjay	Backend & AI Integration Developer
M Devi Naga Sai Ram Prasad	Testing & Deployment Engineer

2. Project Overview

Purpose:

The Sustainable Smart City Assistant leverages IBM Granite AI to assist citizens and urban administrators by providing intelligent, eco-conscious guidance for urban challenges. It enhances sustainability decisions in real time.

Features:

- **Smart City Chat Assistant**: Allows users to ask questions about eco-living, traffic, waste, and energy, Responses are generated by the IBM Granite LLM, Encourages sustainable behavior with clear, friendly suggestions using AI.

- **Traffic Route Optimization**: Users can upload traffic route data (with coordinates), The assistant suggests less congested or shorter alternatives using contextual prompts.
- **Water Usage Trend Prediction** : Users input past water usage (via text or uploaded file), The system predicts future trends

using historical patterns, Results are visualized using line graphs with Chart.js.

- **Waste Management Suggestions**: Provides eco-friendly waste disposal tips, Suggests segregation, recycling, and composting practices based on user context.
- **Data Upload and Analysis**: Users can upload .csv or .txt files containing Water, Traffic, or Waste data, The backend processes the data and extracts insights, AI generates a custom analysis (e.g., "high usage in March," or "congested route in area X").
- **Interactive Map Visualization**: Map is centered on Hyderabad by default, Draws uploaded traffic routes dynamically.
- **Graphical Data Insights**: Water and waste trends are converted into charts: Line chart for water usage over months and Pie/bar chart for waste composition or frequency

3. Architecture

The assistant uses a FastAPI backend, Jinja2 for templates, and integrates IBM Granite LLM.

Frontend:

- Developed using Gradio:
  An open-source Python library used to build the chatbot-style user interface quickly and efficiently.
- Chat Interface:
  Mimics popular messaging platforms; allows users to type queries and view AI responses in real-time.
- Predefined Buttons:
  Includes clickable options for frequently reported issues (e.g., Smart Eco Tips, Water Analysis, Traffic Route Analysis, etc.).
- User Input Field:
  Text box for custom questions; connected to backend logic via Gradio event triggers.

Backend:

- FastAPI app using Python
- Connected with IBM Granite 3.3 or Hugging Face Transformers for natural language understanding and response generation.
- Modular Processing Functions:
  Each user query passes through a modular pipeline (e.g., classification, intent detection, response generation).
- Model prompt handling and file parsing
- Data Layer: Uses uploaded CSV/TXT

Database:

The Sustainable Smart City –Sustainable Smart city Assistant uses IBM's state-of-the-art Granite 3.3 2B Instruct model (ibm-granite/granite-3.3-2b-instruct) as its core reasoning engine to understand and generate human-like responses. While the model itself doesn't require a database, other parts of the application may rely on data storage for interaction history, issue tracking, and performance analytics.

4. Setup Instructions

Prerequsites:

Component	Software / Library	Purpose
Python (>= 3.8)	Python Interpreter	Core language for backend, model integration, and Gradio UI
Gradio	gradio Python library	Used to build the interactive chatbot interface

Component	Software / Library	Purpose
Transformers	transformers (Hugging Face library)	Interface for loading and using IBM Granite model
IBM Granite API	ibm-granite/granite-3.3-2b-instruct	Core language model used for generating AI responses
VS Code	New File	Development environment for prototyping, testing, and running the project

Installation:

Step 1:

Clone the repo

Step 2:

Install Required Dependencies

Install the required Python packages using the following command:
pip install -r requirements.txt

Optional (for future versions using FastAPI):
pip install fastapi uvicorn

Step 3:

Set Up Environment Variables

To securely connect to the IBM Granite model (if authentication is required), define environment variables as follows:

In local development, use a .env file or export the variables in terminal:

export IBM_API_KEY="your_ibm_api_key"
export MODEL_ID="ibm-granite/granite-3.3-2b-instruct"

Step 5:

Run the Application
If using FastAPI for backend deployment (planned in future versions), run:

uvicorn app:app --reload

5. Folder Structure

Client:

Key Components:

- Chat Interface: A real-time conversational UI allowing users to type or select predefined queries.
- Predefined Buttons: For quick issue reporting
- User Input Field: Textbox to type queries.
- AI Response Display: Dynamically updates with model-generated responses.
- Optional Feedback Prompt: Planned feature for collecting user satisfaction.
- Smart city(Root directory)
- Main.py: Main FastAPI application (routes, model integration)
- Requirements.txt: Python dependencies
- **Static(folder)**
- Style.css: Glassmorphism-based frontend styling
- Map.js: JavaScript logic for rendering routes on Leaflet map
- **Templates(folder)**

- Index.html: Main UI with form, map, chart, and response rendering

6. Running the Application

To Run The Smart City Assistant Application Locally .

-Open Your Terminal or command Prompt

-navigate to the project's root directory:

C:\Users\sanju\Desktop\smartcity

-Activate your virtual Envirnoment:

-**Window** : .\venv\script\activate

-Start the Fast Api Server:

Uvicorn main:app –reload

7. API Documentation

Your backend exposes the following main endpoints via FastAPI**:**

Home Page – GET /

Method: GET

Description: Loads the main application form**.**

Response: Returns the index.html template with UI components.

Question Submission – POST /

Method: POST

Description: Submits a user's question related to Smart City topics (traffic, water, waste, eco-living).

These interact with the IBM Granite LLM using structured prompts.

Smart City Chat (General Query)

Function: llm (full_prompt)

Prompt Structure (example):

You are a Sustainable Smart City Assistant.

Your role is to provide clear, intelligent, and eco-conscious responses to urban-related queries.

User Query:

What are the best ways to reduce waste in a housing society?

Provide a helpful and actionable response that:

- Offers practical advice or eco-tips
- Encourages sustainability
- Avoids overly technical language
- Is concise and easy to follow
- Uses friendly, natural tone

Traffic Route Analysis

Function: llm(prompt**)**

Prompt Structure (example):

You are a Smart City Traffic Assistant.

Analyze the following route data and suggest:

- Time-saving strategies
- Alternate routes
- Areas to avoid due to high congestion
- Any patterns you observe

Route Data:

From (17.4, 78.45) to (17.43, 78.48)

From (17.39, 78.42) to (17.41, 78.47)

Provide insights in 3–5 bullet points.

8. Authentication

Authentication is not implemented in the current version of the project.

All features (chat, file upload, map/chart visualization) are accessible without login.

The app is designed for open demonstration and testing purposes**.**

9. User Interface

HTML template rendered via Jinja2

- Dropdown for topic selection
- File upload form
- Chart.js for graph rendering
- Leaflet.js for map visualization
- CSS with glassmorphism effects for modern UI

10. Testing

- **Manual functional testing for all features:**

  Verified all frontend elements (form, buttons, dropdowns, file upload) work correctly.

  Ensured file upload triggers the correct backend logic.

Confirmed map and chart sections display conditionally based on the input data

- **Prompt/response validation from LLM:**

Tested various user queries across all topics (Traffic, Water, Waste, Eco).

Checked the IBM Granite model responses for

-Relevance

-Clarity

-Accuracy

-Appropriate tone and emojis (as per prompt)

-Ensured that the system prompt improves the consistency of replies.

- **Chart and map accuracy check with uploaded files:**

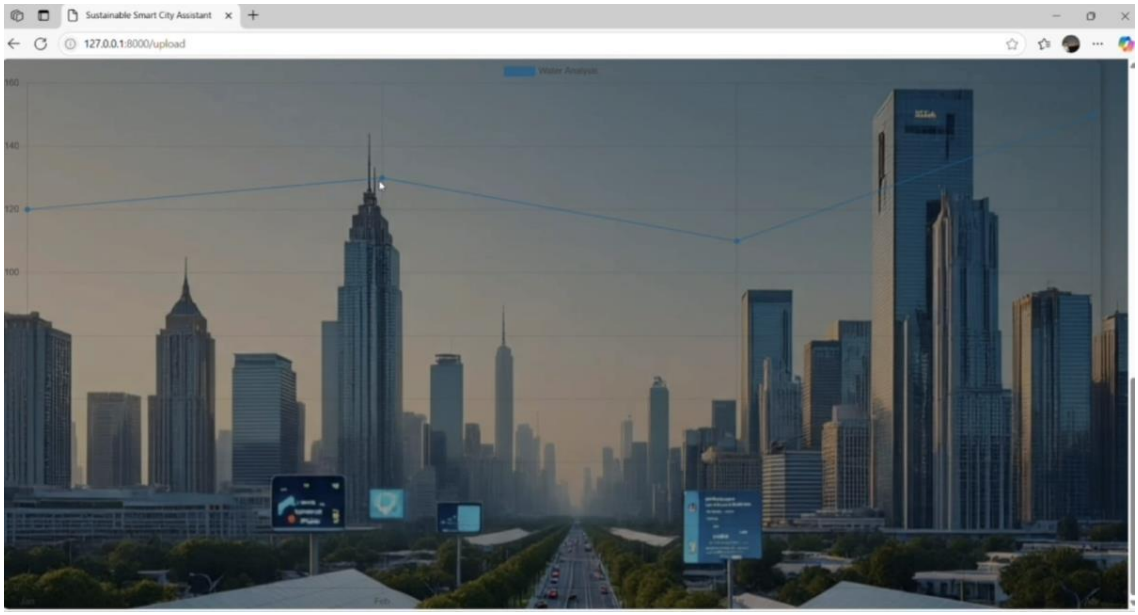
Uploaded sample CSV/TXT files for:

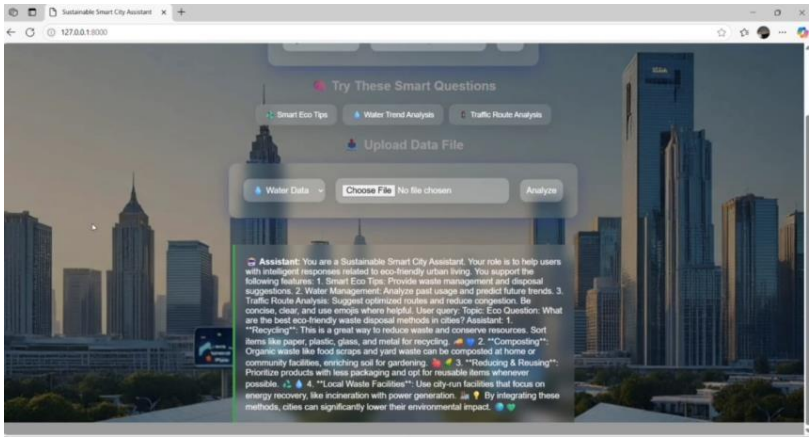
Traffic coordinates → checked route rendering on Leaflet map.

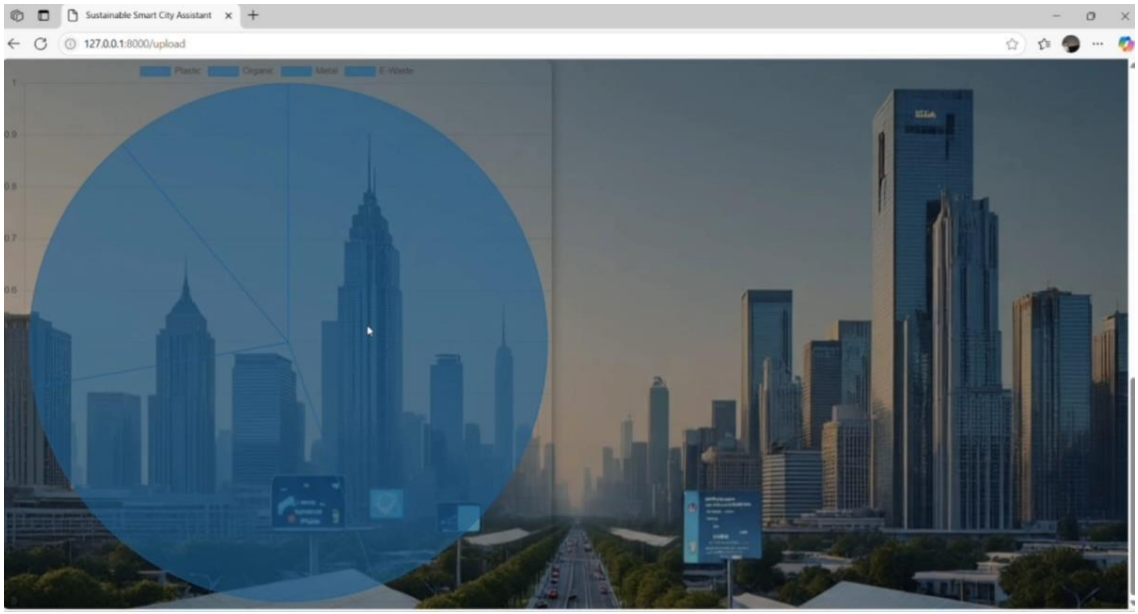
Water/waste metrics → confirmed chart rendering (line/bar/pie) with proper labels and values.

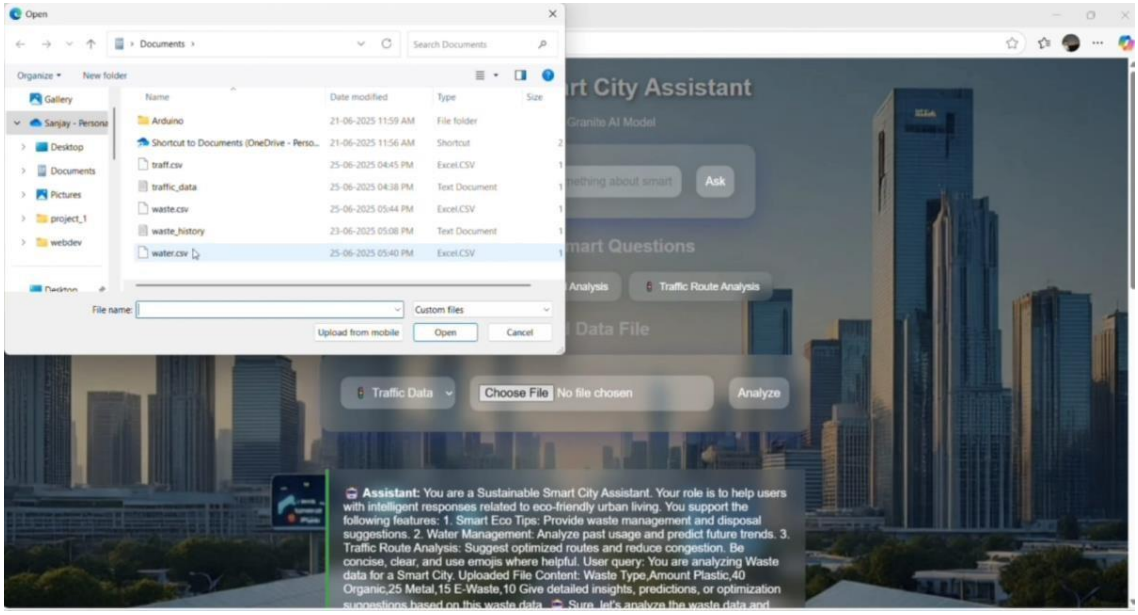
Verified that chart_labels and chart_values match the uploaded file content

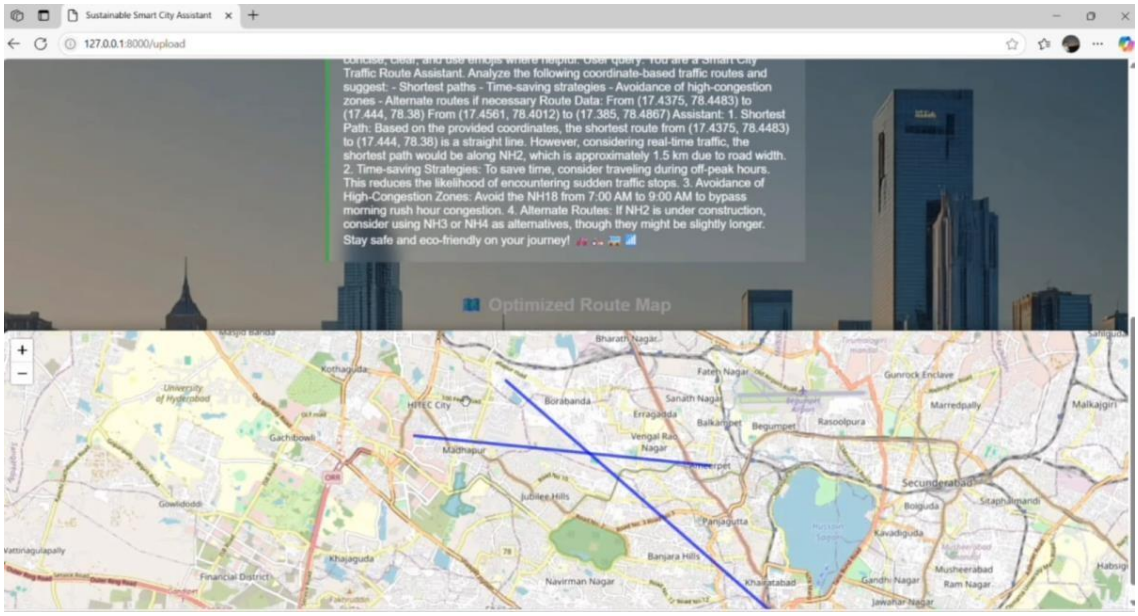
11. Screenshots:


Sustainable Smart City Assistant
127.0.0.1:8000/upload
Water Analysis


Sustainable Smart City Assistant
127.0.0.1:8000
Try These Smart Questions
Smart Eco Tips
Water Trend Analysis
Traffic Route Analysis
Upload Data File
Water Data
Choose File
No file chosen
Analyze
Assistant: You are a Sustainable Smart City Assistant. Your role is to help users with intelligent responses related to eco-friendly urban living. You support the following features: 1. Smart Eco Tips: Provide waste management and disposal suggestions. 2. Water Management: Analyze past usage and predict future trends. 3. Traffic Route Analysis: Suggest optimized routes and reduce congestion. Be concise, clear, and use emojis where helpful. User query: Topic: Eco Question: What are the best eco-friendly waste disposal methods in cities? Assistant: 1. **Recycling**: This is a great way to reduce waste and conserve resources. Sort items like paper, plastic, glass, and metal for recycling. 2. **Composting**: Organic waste like food scraps and yard waste can be composted at home or community facilities, enriching soil for gardening. 3. **Reducing & Reusing**: Prioritize products with less packaging and opt for reusable items whenever possible. 4. **Local Waste Facilities**: Use city-run facilities that focus on energy recovery, like incineration with power generation. By integrating these methods, cities can significantly lower their environmental impact.


Sustainable Smart City Assistant
127.0.0.1:8000/upload
Plastic
Organic
Metal
E-Waste


Open
Documents
Search Documents
Organize
New folder
Gallery
Sanjay - Persona
Desktop
Documents
Pictures
project_1
webdev
Name
Date modified
Type
Size
Arduino
21-06-2025 11:59 AM
File folder
Shortcut to Documents (OneDrive - Perso...
21-06-2025 11:56 AM
Shortcut
traff.csv
25-06-2025 04:45 PM
Excel.CSV
traffic_data
25-06-2025 04:38 PM
Text Document
waste.csv
25-06-2025 05:44 PM
Excel.CSV
waste_history
23-06-2025 05:08 PM
Text Document
water.csv
25-06-2025 05:40 PM
Excel.CSV
File name:
Custom files
Upload from mobile
Open
Cancel
rt City Assistant
Granite AI Model
Ask
mart Questions
Analysis
Traffic Route Analysis
Data File
Traffic Data
Choose File
No file chosen
Analyze
Assistant: You are a Sustainable Smart City Assistant. Your role is to help users with intelligent responses related to eco-friendly urban living. You support the following features: 1. Smart Eco Tips: Provide waste management and disposal suggestions. 2. Water Management: Analyze past usage and predict future trends. 3. Traffic Route Analysis: Suggest optimized routes and reduce congestion. Be concise, clear, and use emojis where helpful. User query: You are analyzing Waste data for a Smart City. Uploaded File Content: Waste Type,Amount Plastic,40 Organic,25 Metal,15 E-Waste,10 Give detailed insights, predictions, or optimization



12. Issues:

Despite the project's stable functionality, the following limitations are currently observed:

Initial Model Loading Delay

The IBM Granite model may take several seconds to load on first use, especially if it's being loaded from disk or downloaded from Hugging Face cache.

This could result in delayed responses for the first query after deployment.

Limited File Format Tolerance

The system expects .csv or .txt files with a specific structure:

- Traffic: Lat1, Lon1, Lat2, Lon2
- Water/Waste: Label, Value

Invalid or malformed files may be silently ignored or cause partial processing.

No Persistent User State or Authentication

The current implementation does not include:

- User login or session tracking
- Saving of chat history or uploaded data per user

This limits personalization and long-term data insights.

13. Future Enhancements:

To increase functionality, usability, and scalability, the following features are planned for future versions of the Sustainable Smart City Assistant:

Add User Authentication

Implement secure login and registration using JWT tokens.

Differentiate between public users and admin users.

Ensure private access to uploaded data and user-specific responses.

Integrate Real-Time Traffic APIs

Use third-party APIs (e.g., Google Maps, TomTom) to:

Fetch live congestion data.

Suggest dynamic route alternatives.

Improve route insights with ETA, distance, and speed data

Store User Sessions & Chat History

Persist user interactions and uploaded files using:

MongoDB or PostgreSQL for structured storage.

Session tracking to enable historical insights and reuse of past data.

Enable users to view and manage previous queries.

Add Mobile Responsiveness

Redesign the UI using responsive CSS or frameworks like Bootstrap/Tailwind.

Optimize layout for small-screen experiences without loss of features.

Migrate UI to React for Full MERN Stack

Replace the current Jinja2-based frontend with a dynamic React interface.

Integrate with FastAPI (or migrate to Node.js for full MERN).

Enable richer, real-time interactivity and modular UI components.