Citizen-AI: Detailed Documentation

## SUSTAINABLE CITIZEN ASSISTANT USING IBM GRANITE LLM

## PROJECT DOCUMENTATION

## INTRODUCTION :

PROJECT TITLE: SUSTAINABLE CITIZEN-AI ASSISTANT USING IBM GRANITE LLM

TEAM MEMBERS : ALISHA ZARIN.J

TEAM MEMBERS : KAVI BARATHI.G

TEAM MEMBERS : FAMIDHA.MR

TEAM MEMBERS : KARISHMA KUMARI.D

# Project Overview:

Citizen-AI is a project designed to leverage the IBM Granite language model for providing AI-driven insights into city safety and government-related services. The application is built with Gradio to create a user-friendly interface where users can interact with the system seamlessly. The project is structured around two major features:

• City Analysis – Provides crime index, accident rates, and overall safety assessment of any city

• Citizen Services – Acts as a government assistant, answering public queries about civic issues, policies, and services.

# 2. Purpose:

To empower citizens with instant AI-driven insights about city safety and governance.To assist government bodies in responding to queries in a structured, AI-backed wayTo serve as an educational/demo project showing how LLMs + Gradio can be applied to civic technology.

# 3. Features

✅ City Analysis Tab – Users enter a city name to receive AI-generated insights on:

Crime index

Accident rates

Overall safety evaluation

✅ Citizen Services Tab – Users enter a query about public services or government policies and receive structured, assistant-style responses.

✅ Automatic Device Optimization – Uses GPU if available (float16 precision), otherwise falls back to CPU.

✅ Gradio UI – Clean, two-tab interface for easy navigation.

✅ Custom Prompting – Prompts are engineered to provide structured, helpful outputs.

# 4. Conversational Interface

The system provides a chat-like conversational experience where:

The user types in natural language queries.

The AI generates a human-like, structured response.

Unlike traditional FAQs, answers are dynamic and context-aware.

This conversational flow makes Citizen-AI feel like interacting with a virtual civic officer rather than browsing static information.

# 5. System Architecture

🔹 Backend (AI Processing)

Model: IBM Granite 3.2 2B Instruct (Causal LM).

Tokenizer: Converts user input → tokens → processed by model.

Response Generator:

Prepares prompt based on tab (city analysis or citizen query).

Runs inference with controlled parameters (temperature, sampling).

Post-processes output to remove redundancies.

🔹 Frontend (User Interface)

Built using Gradio Blocks and Tabs.

# Two Tabs:

1. City Analysis Tab → Input: City name → Output: Structured safety analysis.

2. Citizen Services Tab → Input: Query → Output: Government-like response.

UI Components:

Textboxes (input & output).

Buttons (trigger analysis or query).

Tabs (switch between features).

🔹 High-Level Flow

User Input → Gradio Frontend → Tokenizer → Granite Model →

Response Generator → Output Processing → Gradio Frontend → User

# 6. Further Enhancements

🔮 Data Integration

Connect to real-time crime statistics APIs or government open data portals.

Enhance city analysis with live data + AI reasoning.

🔮 User Personalization

Add user profiles to track queries and provide personalized suggestions.

🔮 Multi-language Support

Extend beyond English to support local languages (e.g., Hindi, Tamil).

🔮 Voice Support

Integrate speech-to-text for queries and text-to-speech for responses.

🔮 Mobile-friendly UI

Optimize Gradio frontend for mobile apps or PWA (Progressive Web App).

🔮 Advanced Features

Add visualization (charts, maps) for crime/accident trends.

Build admin dashboard for government officials.

# System Architecture The architecture of Citizen-AI is relatively straightforward and modular:

. It includes the following layers:

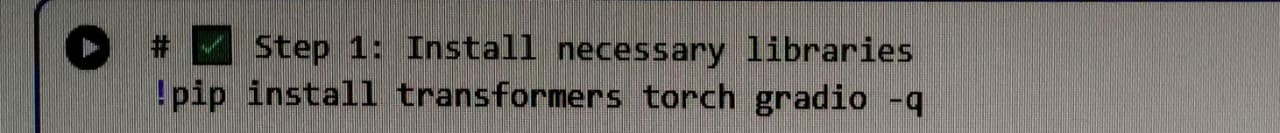
• Model Layer: Uses IBM Granite-3.2-2B Instruct model for text generation.

• Processing Layer: Includes tokenizer and response generation functions that handle text inputs and outputs.

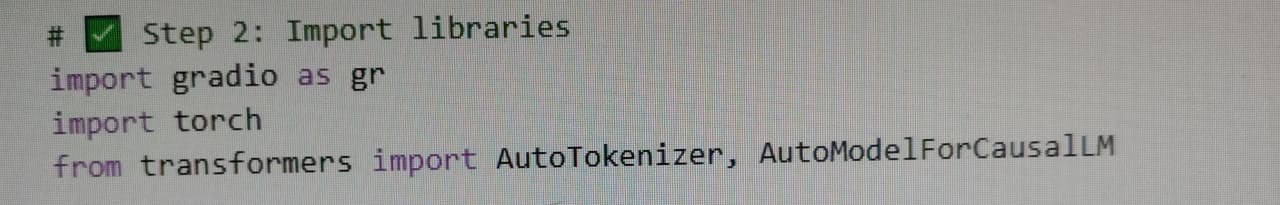
• Interface Layer: Built with Gradio Blocks and Tabs, allowing easy interaction via a web-based interface.

## Step-by-Step Code Explanation:

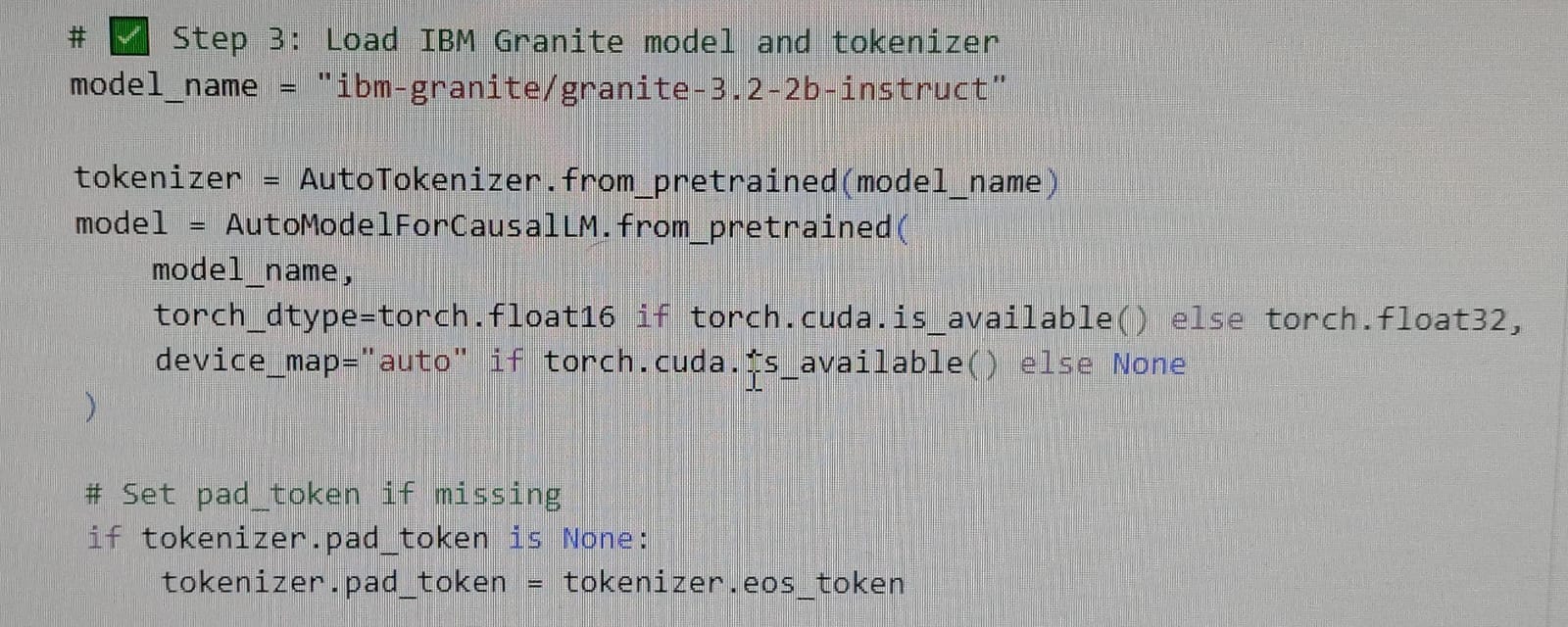
Step 1: Installing Dependencies The project requires three primary libraries – Transformers for accessing the Granite model, Torch for running the model computations on CPU/GPU, and Gradio for building the web interface.

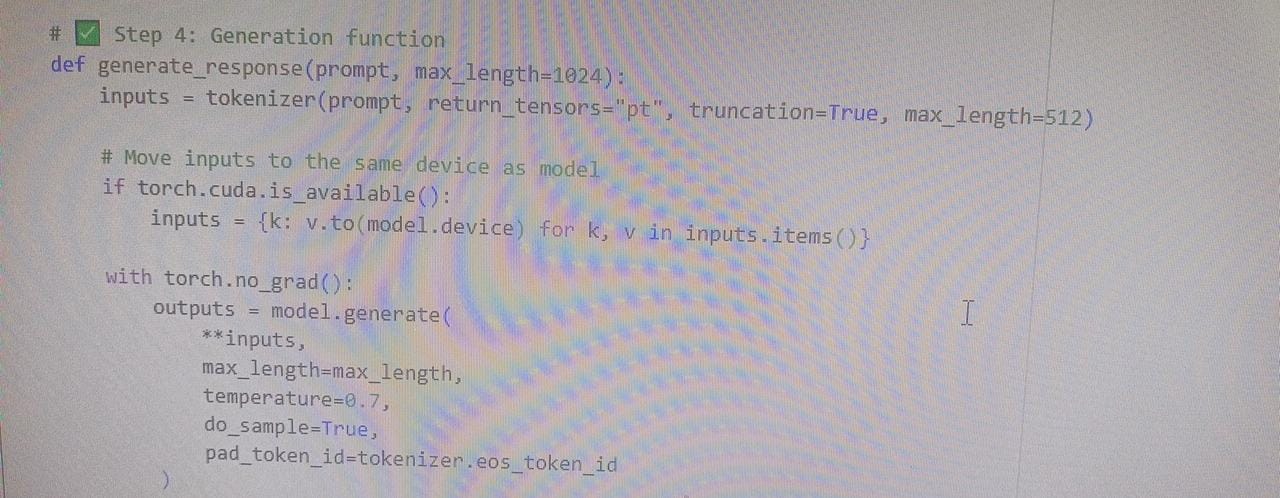


Step 2: Importing Libraries The essential Python packages are imported: Gradio for UI, Torch for model execution, and Transformers for the Granite model.

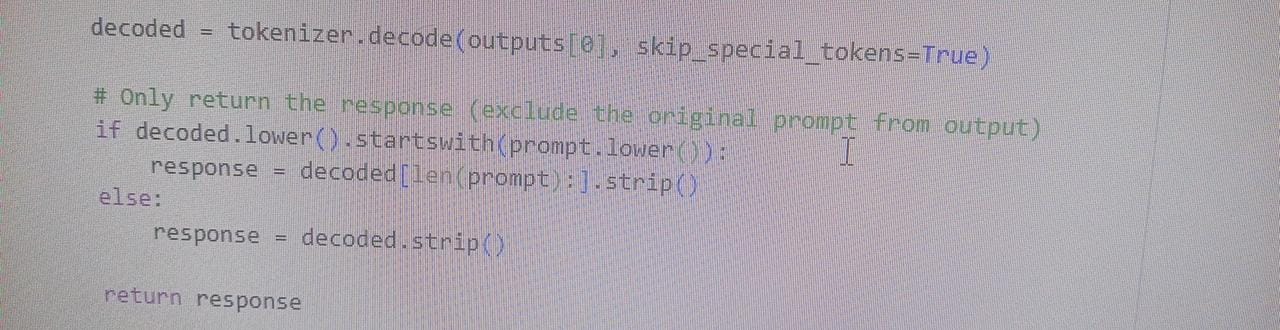


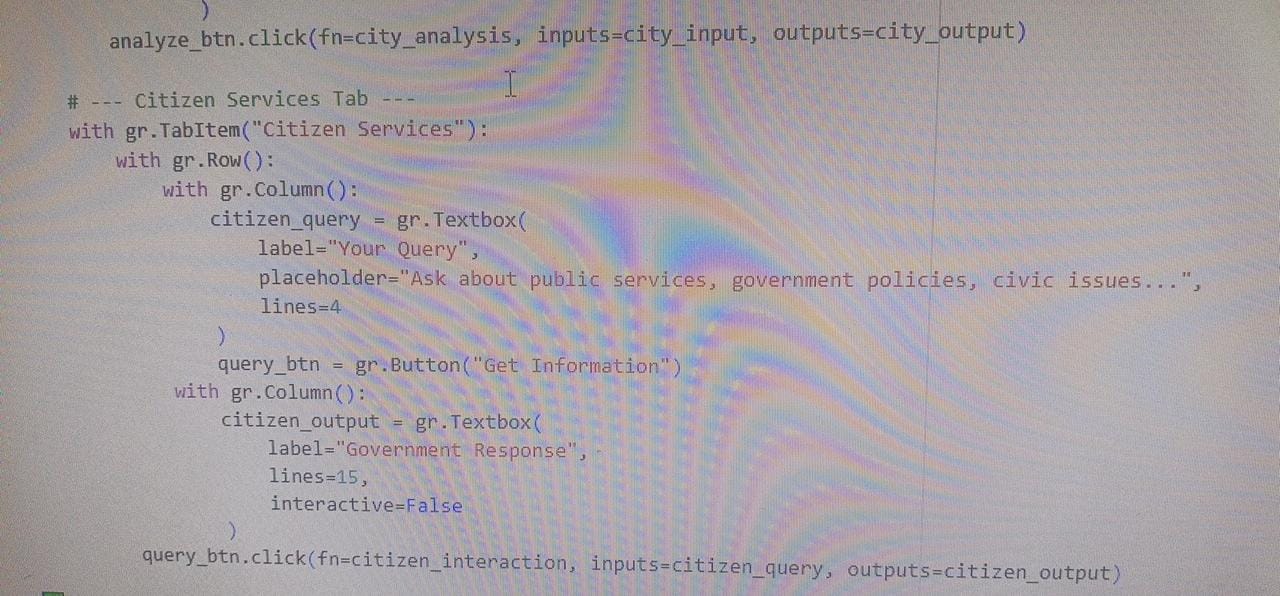
Step 3: Loading the Granite Model The IBM Granite model and its tokenizer are loaded from Hugging Face. The code also detects if a GPU is available and sets the model to use optimized data types (float16 for GPU, float32 for CPU). A padding token is assigned to ensure the model processes inputs correctly.



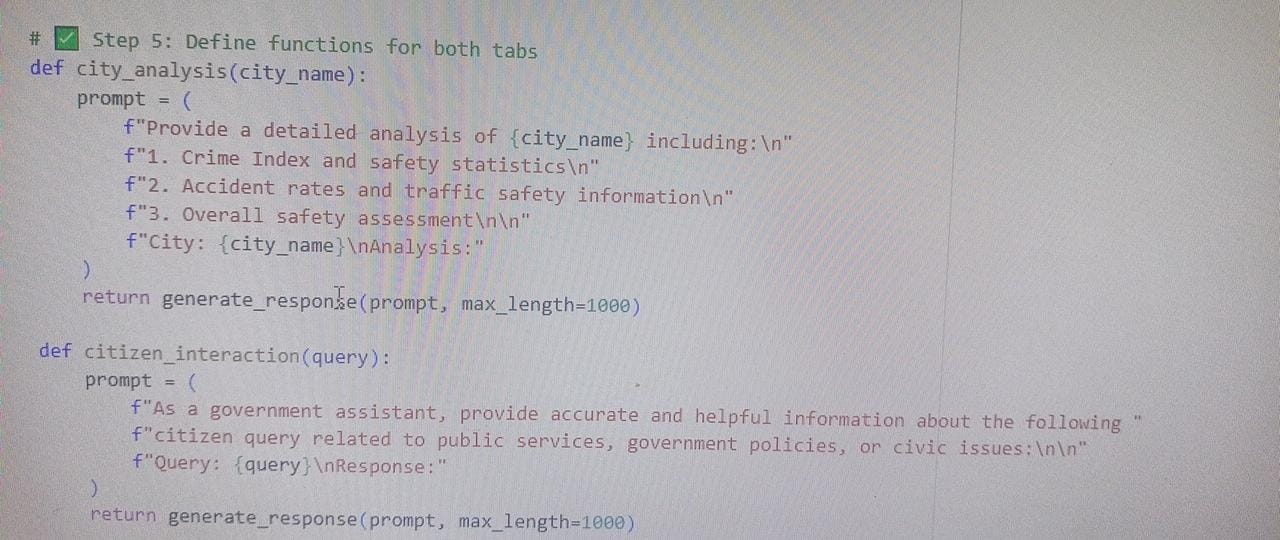
Step 4: Response Generation Function The `generate\_response` function handles the process of taking user input, tokenizing it, passing it through the model, 

and decoding thegenerated output. It also ensures that unnecessary repetition of the input prompt is removed from the output.

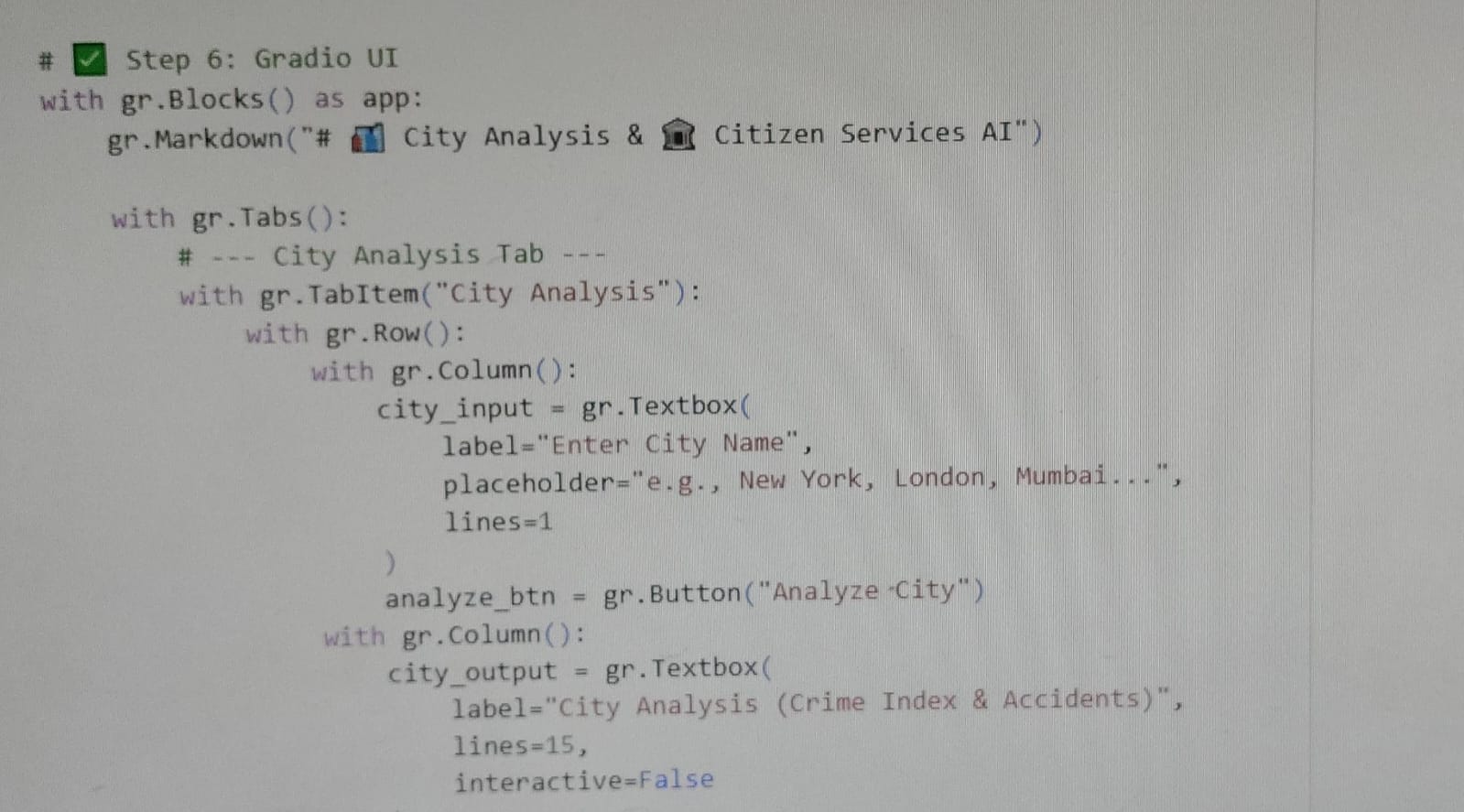




Step 5: Feature Functions Two specialized functions are built on top of the response generator: `city\_analysis` for analyzing safety aspects of cities, and `citizen\_interaction` for answering queries related to governance and civic issues.

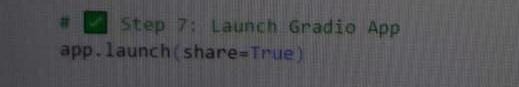


Step 6: Gradio Interface The interface is built using Gradio Blocks and Tabs. It contains two separate tabs for the two features.



Each tab includes input textboxes, action buttons, and output areas for displaying the model’s responses.

Step 7: Launching the App Finally, the Gradio app is launched with `share=True`, which generates a public link that can be shared with others. This makes the application accessible outside of the development environment.



# Key Features:

• Integrates IBM Granite model for natural language understanding and generation.

• Provides structured analysis of cities regarding crime and accidents.

• Acts as a civic assistant for answering government-related queries.

• Supports GPU acceleration automatically if available.

• Uses a clean, tabbed interface built with Gradio for easy navigation.

# Potential Use Cases :

• Urban planning departments analyzing city safety trends.

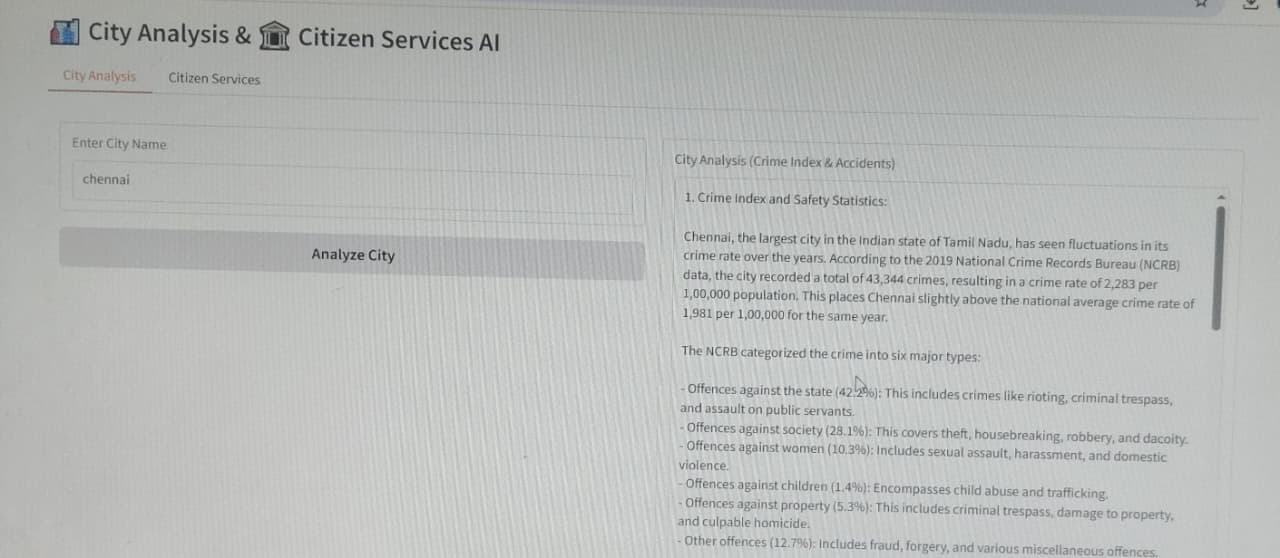
• Citizens seeking reliable information about local governance and public services

• Journalists and researchers gathering quick AI-assisted insights on civic issues.

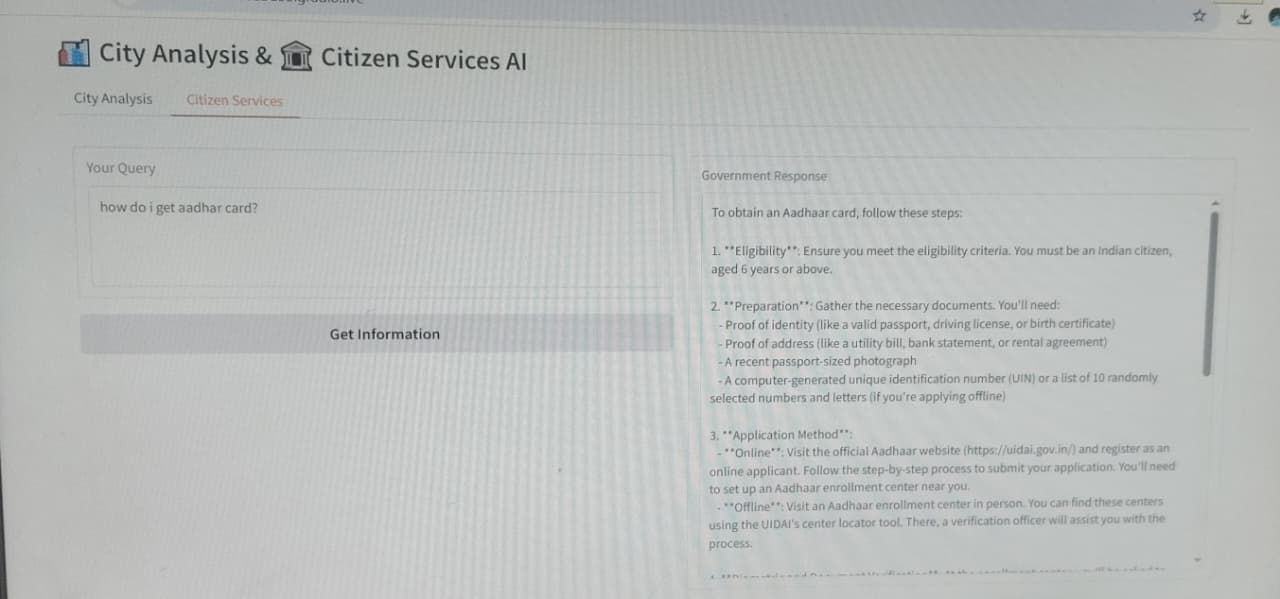
• Educational tools for teaching AI applications in governance and public administration.

# TESTING OUTPUT:

## City Analysis:



## City services:



# CONCLUSION:

Citizen-AI demonstrates how large language models like IBM Granite can be integrated into civic technology solutions. By combining robust NLP capabilities with a simple web-based interface, the system empowers both individuals and organizations to access structured, AI-generated insights into city safety and governance. This project can serve as a foundation for more advanced civic AI systems with added data sources and real-time updates.