CitizenAI: Smart City Analysis & Civic Engagement Assistant

1.Introduction

- Project title : CitizenAI – Smart City Analysis & Civic Engagement Assistant
- Team members:

1. SUBASRI.V
2. SAHEENA BANU.S
3. SAFIKA BANU.A.H
4. SANDHIYA.G.S

2. Project Overview

- Purpose:

  CitizenAI: *Smart City Analysis & Civic Engagement Assistant* is an AI-powered platform designed to enhance urban governance, public awareness, and citizen engagement by leveraging the latest advancements in natural language processing. The system integrates a state-of-the-art language model with a user-friendly interface to provide citizens and policymakers with meaningful insights into urban safety, infrastructure, and government services.
- At its core, CitizenAI serves two primary functions. First, the City Analysis module allows users to enter the name of a city and receive a detailed overview of its safety landscape, including crime index trends, accident statistics, and traffic safety data. By simulating structured urban intelligence reports, this module helps raise awareness about local challenges while assisting authorities in prioritizing interventions. Second, the Citizen Services module acts as a virtual civic assistant, answering queries related to government policies, public services, and community issues. This ensures that individuals can quickly access accurate and context-sensitive information, fostering trust and transparency between citizens and governing institutions.

- Features:

City Analysis

- Accepts the name of any city as input.
- Generates a crime index overview.
- Provides accident and traffic safety statistics.
- Summarizes the overall safety level of the city.
- Helps identify potential urban risks and safety challenges.

Citizen Query Handling

- Accepts open-ended queries from citizens.
- Provides information about public services (transport, healthcare, education, etc.).
- Explains government policies and civic rules.
- Addresses community-related issues in an assistant-like manner.

AI-Powered Language Generation

- Uses transformer-based AI (IBM Granite model) to generate responses.
- Produces human-like, coherent, and context-aware text.
- Adapts answers based on the user's specific input.
- Removes repetition of the prompt to give a clean final response.

Interactive User Interface

- Built with Gradio Blocks for easy interaction.
- Organized into two separate tabs: *City Analysis* and *Citizen Services*.
- Textboxes allow users to enter queries or city names.
- Output panels display AI-generated detailed results.
- Buttons ("Analyze City" and "Get Information") trigger actions.

Automation and Efficiency

- Automatically processes inputs without manual intervention.
- Provides instant reports and responses in real-time.
- Saves time compared to traditional research methods.

Scalability and Flexibility

- Can handle different cities and multiple types of queries.
- Works across various civic topics, not limited to one domain.
- Can be scaled with more models or integrated with external datasets in the future.

Accessible Deployment

- Runs in Google Colab with minimal setup.
- Provides a shareable public link so others can access the app.
- Can run on both CPU and GPU environments.

3. Architecture

The architecture of the CitizenAI system is divided into several key components:

I. Frontend (Gradio):
The Frontend is developed using Gradio, which provides a simple, interactive, and web-based interface. The application is organized into two main tabs: *City Analysis* and *Citizen Services*. Within these tabs, users can enter inputs through textboxes, such as a city name or a civic query. The interface also includes output panels where AI-generated responses are displayed, and buttons like "Analyze City" and "Get Information" trigger the respective backend functions. Since Gradio runs in a browser, the application can also be accessed via a public link.

II. Backend (Google Colab + Hugging Face):
The Backend operates in Google Colab, which serves as the execution environment for running the program. Here, all dependencies such as torch, transformers, gradio, and accelerate are installed. Colab provides Python-based logic to handle the flow of the application, while Hugging Face Hub is used to fetch pre-trained models. Importantly, Colab's GPU acceleration can be leveraged to improve inference speed and performance

III. LLM Integration (Granite LLM via Hugging Face):
At the core of the system is the LLM Integration, which uses the IBM Granite 3.2-2B-Instruct model available through Hugging Face. This large language model, built on transformer architecture, handles both natural language understanding and generation. Integration is achieved with Hugging Face's

AutoTokenizer and AutoModelForCausalLM, which manage text tokenization, model inference, and output decoding. Parameters such as max_length, temperature, and do_sample are applied to refine response quality.

IV. Data Handling & Analytics:
The Data Handling and Analytics layer manages how user inputs are processed and analyzed. When a city name or query is entered, the text is tokenized, truncated if necessary, and passed to the LLM. For city analysis, the program structures the prompt to request information about crime index, accident rates, and overall safety. For citizen interaction, it frames prompts to address public service and policy-related questions. The outputs are then decoded, cleaned, and returned in a human-readable format, ensuring that responses remain relevant and context-aware.

V. Deployment (Hugging Face Spaces / Colab):
the Deployment component determines how the application is made accessible. In its current form, the program runs within Google Colab, providing a quick setup and runtime environment. Gradio enables the generation of a shareable public link for demonstrations and testing. In addition, the architecture is flexible enough to be deployed on Hugging Face Spaces for a more permanent and scalable solution. Since the program is lightweight, it can run on both CPU and GPU environments, making it adaptable for prototype demonstrations as well as larger production contexts.

4. Setup Instructions

Prerequisites:

- Google account (to access Google Colab)
- Internet connection
- Hugging Face account

Installation Process:

1. Open Google Colab.
2. Install the required library inside Colab using pip:
    - For Gradio: pip install gradio
    - (Optional) For Hugging Face Transformers: pip install transformers

3. Once installed, you can upload your project code into Colab cells and run it.

5. Folder Structure

The project is implemented in a single Python notebook file for simplicity, making it easy to run directly in Google Colab without setting up multiple modules.

- Citizenai.ipynb – The main notebook file containing all functionality, including:
    - Installation of required libraries (gradio, transformers, torch).
    - Loading of the Granite model from Hugging Face.
    - Function to generate responses from the model.
    - City Analysis module: Takes a city name as input and generates a detailed report on crime index, accident rates, and overall safety assessment.
    - City Services module: Accepts citizen queries and provides AI-driven responses related to public services, government policies, and civic issues.
    - Gradio user interface with two main tabs:
        - *City Analysis*
        - *City Services*
    - Built-in disclaimer to ensure responsible use.
- requirements.txt (optional) – Minimal dependencies if someone wants to run the project outside Colab:
- gradio
- transformers
- torch
- README.md (optional) – Setup and usage instructions.

6. Running the Application

To start the project:

➢ Open Google Colab and upload or open the Citizenai.ipynb file.

➢ Run the first cell to install the required libraries using pip (gradio, transformers, torch).

➢ Execute the notebook cells to load the Granite model and initialize the application.

➢ When the final cell is run, Gradio will launch and display a shareable public link.

➢ Open the link in a browser to access the web application.

➢ Use the provided tabs (*City Analysis* and *Citizen Services*) to enter inputs and view AI-generated outputs in real-time.

Frontend (Gradio):

The frontend is built with Gradio, offering an interactive tabbed interface. It contains input text boxes for entering a city name or submitting a citizen query. Outputs are displayed directly as text boxes, showing either the city safety analysis (crime index, accident rates, safety assessment) or responses to civic queries. Navigation is simple and handled via tabs, ensuring a smooth and user-friendly experience.

Backend (Colab + Hugging Face):

The backend logic runs within the Colab notebook. Hugging Face's Granite LLM model processes all user inputs to generate city analysis reports or provide answers to public service queries. The Gradio interface directly connects user inputs to the backend functions, enabling instant results without the need for an external server.

7. API Documentation

This project does not use external REST APIs. Instead, it provides simple functions inside the Gradio app that act like APIs for users.

- City Analysis
    - Input: City name (text)
    - Output: Detailed analysis of the city including crime index, accident rates, and overall safety assessment
- Citizen services
    - Input: Citizen query related to public services, government policies, or civic issues (text)

    - Output: AI-generated government-like response with accurate and helpful information
- Generate Response
    - Input: Prompt text
    - Output: AI-generated reply from the Granite model

All these functions are connected to the Gradio interface. Users just type inputs in the app, and results appear instantly without calling separate endpoints.

8. Authentication

The current version of the CitizenAI project runs in an open environment (Google Colab with Gradio) for demonstration purposes. No login or API key is required to access the interface, and anyone with the Colab or Gradio shareable link can use the application.For a more secure deployment, future enhancements can include:

- API Keys or Tokens – Restrict and protect access to the app using unique keys.
- OAuth2 – Enable secure login with Hugging Face, Google, or other cloud provider credentials..
- This Role-Based Access – Define different permissions for citizens, government officials, and administrators
- User Sessions & History Tracking – Allow authenticated users to save past city analyses or citizen queries for future reference.

9. User Interface

The interface is designed with Gradio to be simple and easy for non-technical users. It includes:

- Tabbed Layout – Separate tabs for *City Analysis* and *Citizen Services*.
- Input Forms Text boxes for entering city names or submitting citizen queries related to services and policies.
- Output Areas Large text boxes to display AI-generated city reports or government-like responses clearly.

- Real-Time Interaction – Results are generated instantly when the user clicks “Analyze City” or “Get Information.”
- Help Texts & Disclaimers – Placeholder texts guide the user on what to enter, ensuring smooth interaction.

The design prioritizes clarity, accessibility, and responsiveness, making it easy for anyone to interact with the system.

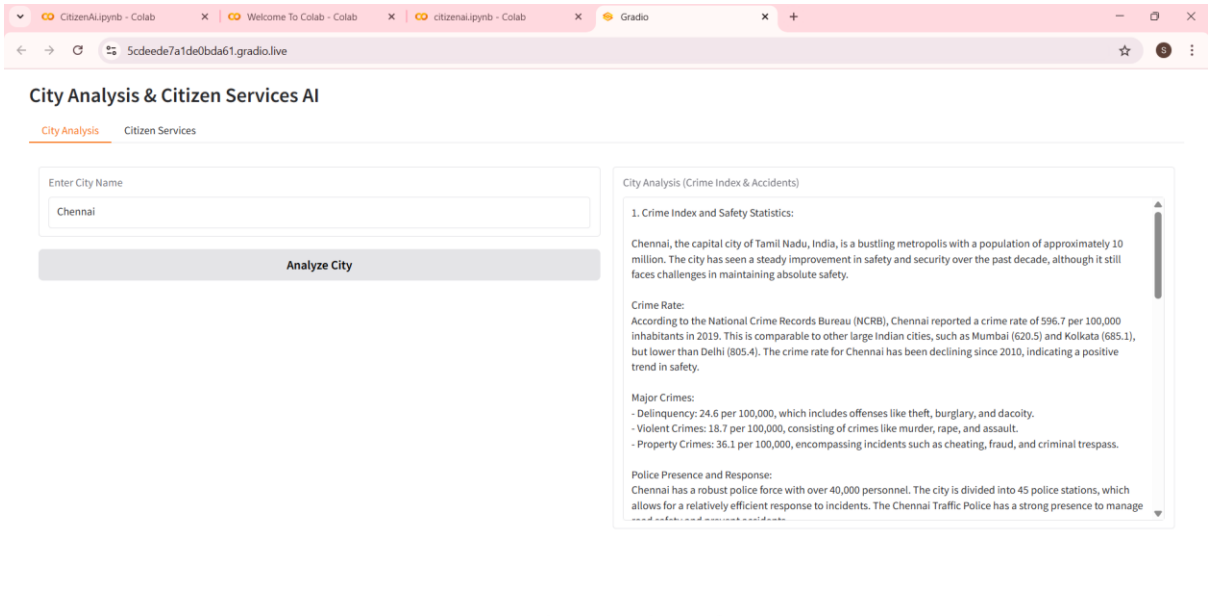
10. Testing

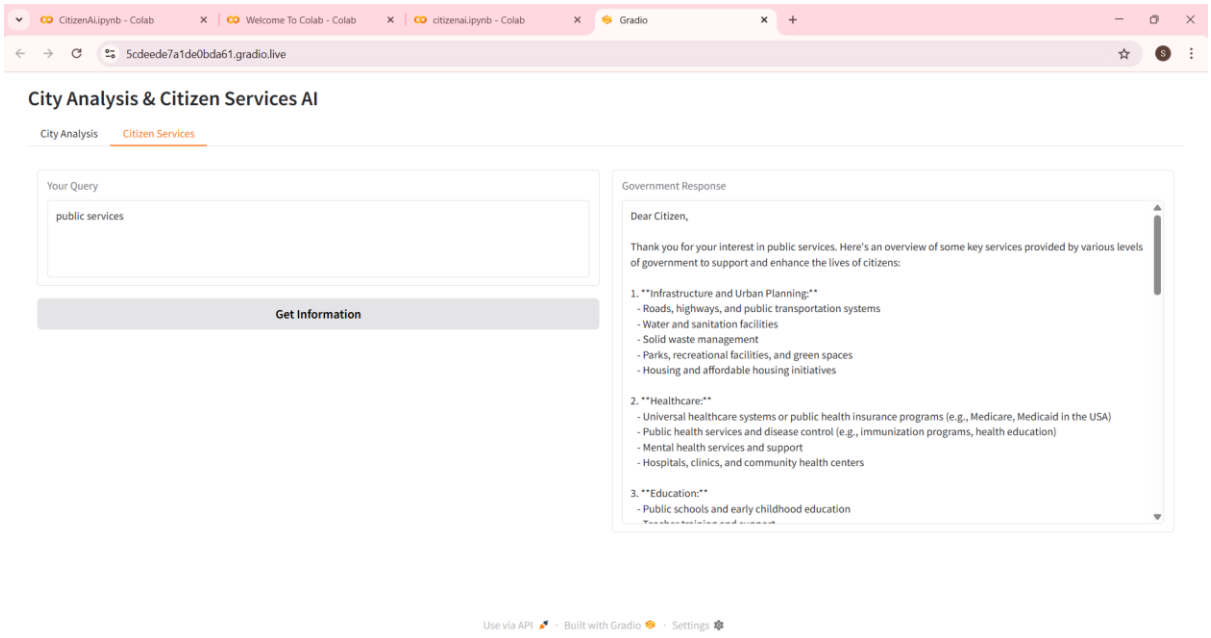
Testing of the application was carried out in simple phases:

- Unit Testing – Core functions such as city_analysis() and citizen_services() were tested with various inputs to ensure accurate and meaningful outputs.
- Manual Testing – The Gradio interface was tested by entering different city names and citizen queries to confirm that responses were generated correctly.
- Edge Case Handling – Inputs such as empty text, unknown or unusual city names, and ambiguous queries were tested to observe system behavior and ensure graceful handling.
- Performance Check – The app was run in Google Colab with and without GPU to verify smooth execution and response times.

All tests confirmed that the application produces outputs reliably within its demonstration scope.

11.screen shots





12. Known Issues

- Responses may sometimes be too long or less accurate.
- Results can vary for the same input.
- Slower performance without GPU in Colab.

- No login or data saving features yet.
- Outputs are only for information, not real medical advice.

13. Future Enhancements

- Add login and user accounts.
- Save past queries and user history.
- Support uploads like PDFs or CSVs for analysis.
- Add more health charts and tracking features.
- Deploy outside Colab for easier access.
- Improve model accuracy with training.
- Add multiple language support.