Citizen AI – Intelligent Citizen Engagement Platform

Project Documentation

Introduction
Project title : Citizen Ai
Team member : Rithika R
Team member :Shalini S
Team member : Shamyuktha D
Team member :Sharmila M

Project overview:
This project is an AI-powered assistant that provides both city analysis and citizen service support through a simple web-based interface. It is designed to help users gain insights into the safety conditions of cities by generating detailed reports on crime index, accident rates, and overall safety. At the same time, it functions as a government-style assistant that answers queries related to public services, policies, and civic issues in a clear and official tone. The system takes user input in the form of a city name or a citizen query, processes it using a large language model, and produces structured, informative responses that are easy to understand. The interactive application is built to be user-friendly and accessible, making it useful for citizens, travelers, policymakers, and government agencies alike. In the future, the project can be enhanced with real-time data integration, multilingual support, and voice-based interaction to make it more practical and widely applicable.

Architecture:
The new architecture would be broken down into four main layers:

Front-end (Streamlit):

A user-facing web application built with Streamlit. Unlike Gradio, which is excellent for quick demos, Streamlit is better suited for building more polished, feature-rich dashboards and applications. It will handle the user interface for both the city analysis and citizen services functions, sending requests to the backend API.

Back-end (FastAPI):

A high-performance, asynchronous API built with FastAPI. This will be the central hub that receives requests from the front-end. It'll contain all the business logic, including the LLM integration, data processing, and communication with other services. FastAPI's built-in automatic documentation with OpenAPI (Swagger UI) is a huge advantage for development and testing.

LLM Integration Layer:

This layer is managed by the FastAPI backend. It's responsible for interacting with the language model.
IBM watsonx Granite: This is the core LLM that powers the application. The backend will use the Hugging Face transformers library to load and run this model, similar to the original code. For production, you could consider using IBM's official watsonx-python-sdk for a more stable and enterprise-grade connection.
Data & ML Layer:

This layer contains the services for data retrieval and specialized machine learning tasks.
Vector Search (Pinecone): To improve the accuracy and relevance of the LLM's responses, especially for specific, up-to-date information, you can implement a Retrieval-Augmented Generation (RAG) system. A vector database like Pinecone would store embeddings of a knowledge base (e.g., city crime reports, government documents). When a user asks a question, the system retrieves relevant documents from Pinecone and provides them to the LLM as context, reducing "hallucinations."
ML Modules (Forecasting & Anomaly Detection): These modules would be separate services within the backend. They could use
pre-trained models or be trained on relevant data. For example, they could analyze historical crime data to forecast future trends or detect anomalous spikes in accidents. Common algorithms for these tasks include ARIMA for time series forecasting and Isolation Forest or One-Class SVM for anomaly detection.

Setup Instructions
Prerequisites

Python 3.8+
pip package installer
Access to an IBM watsonx account (or Hugging Face hub for the open model)
A Pinecone API key and environmen

Installation Process
Clone the Repository
Set up Virtual Environment
Install Dependencies: Create a requirements.txt file in the root directory and install everything

Folder Structure
A well-organized structure is crucial for a multi-layered application.

Running the Application

Backend: Navigate to the backend directory and run the FastAPI server using
uvicorn
Frontend: Open a new terminal, navigate to the frontend directory, and run the Streamlit app streamlit run app.py .This will launch the Streamlit app in your web browser, which will make requests to the running FastAPI server.

API Documentation

FastAPI automatically generates interactive API documentation. Once the backend server is running, you can access the OpenAPI documentation at
Swagger UI: http://127.0.0.1:8000/docs

Redoc: http://127.0.0.1:8000/redoc

Authentication

For a production environment, you should implement user authentication to secure the API endpoints. A common and secure method with FastAPI is using OAuth2 with JWT (JSON Web Tokens). This would involve:
A user logs in with a username/password, and the backend validates their credentials.
Upon successful validation, the backend generates and returns a JWT.
For subsequent requests, the user's front-end sends the JWT in the
Authorization header.
FastAPI's security dependencies can then automatically validate this token for protected routes.

User Interface

The Streamlit front-end will mirror the original Gradio app's functionality but with a more polished look. It'll use a sidebar for navigation between the "City Analysis" and "Citizen Services" pages. Users can input data, see progress indicators, and view the
AI-generated responses in styled text boxes.

Testing

Unit Tests: Use pytest to test individual functions, such as the
generate_response helper or the functions that interact with Pinecone.
Integration Tests: Test the API endpoints to ensure they correctly handle requests and return the expected responses, including proper error handling for invalid inputs or authentication failures.

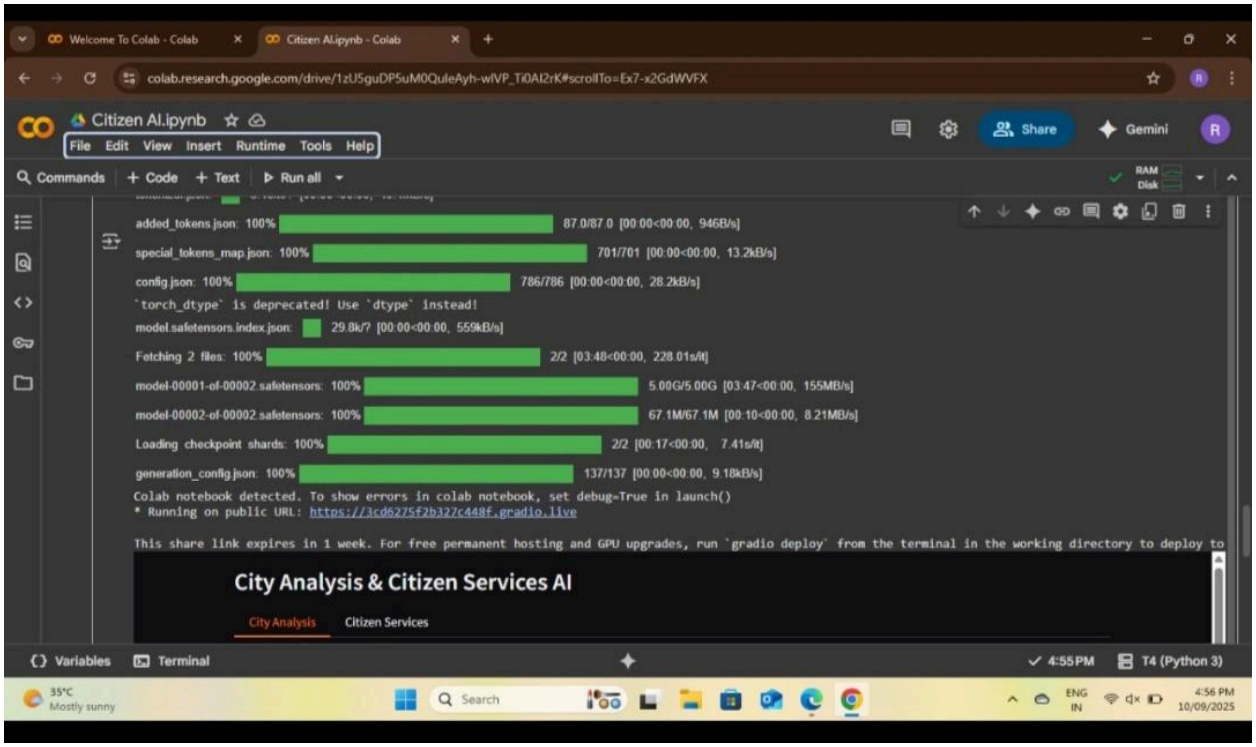
Known Issues

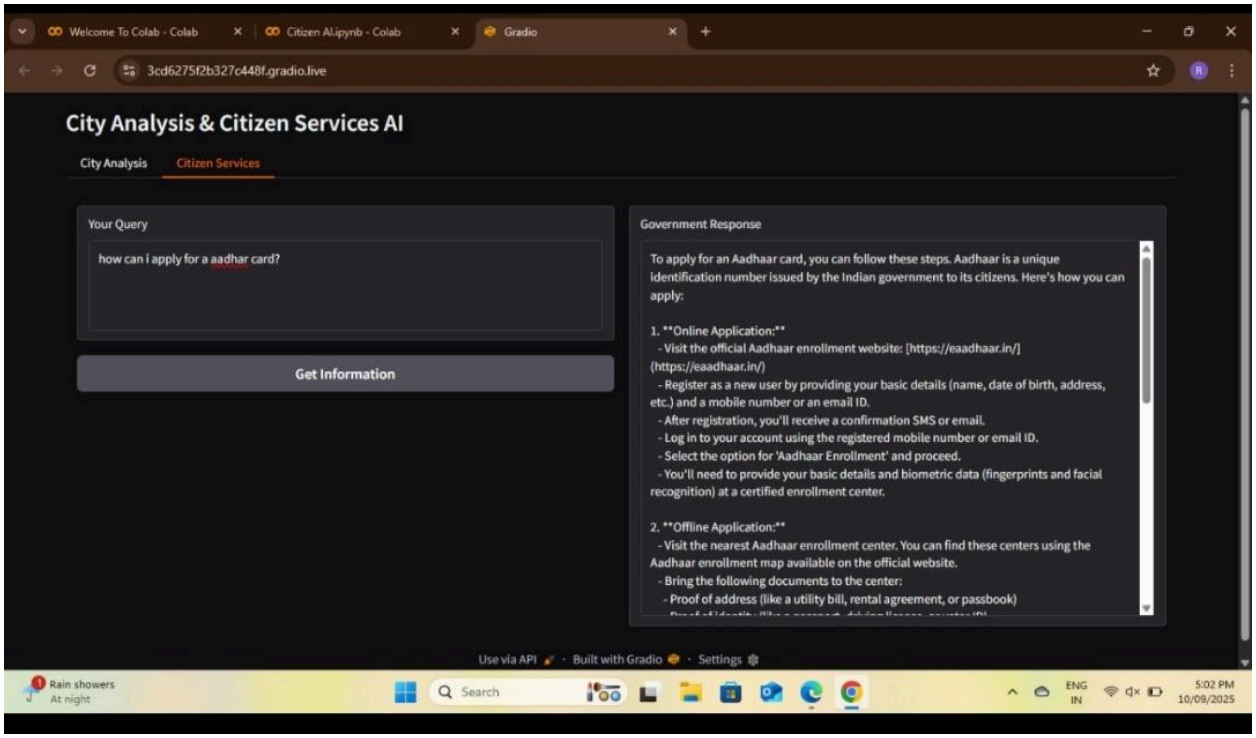
LLM Hallucinations: Despite the RAG setup, the LLM may still generate factually incorrect information. This is an inherent risk of current LLMs.

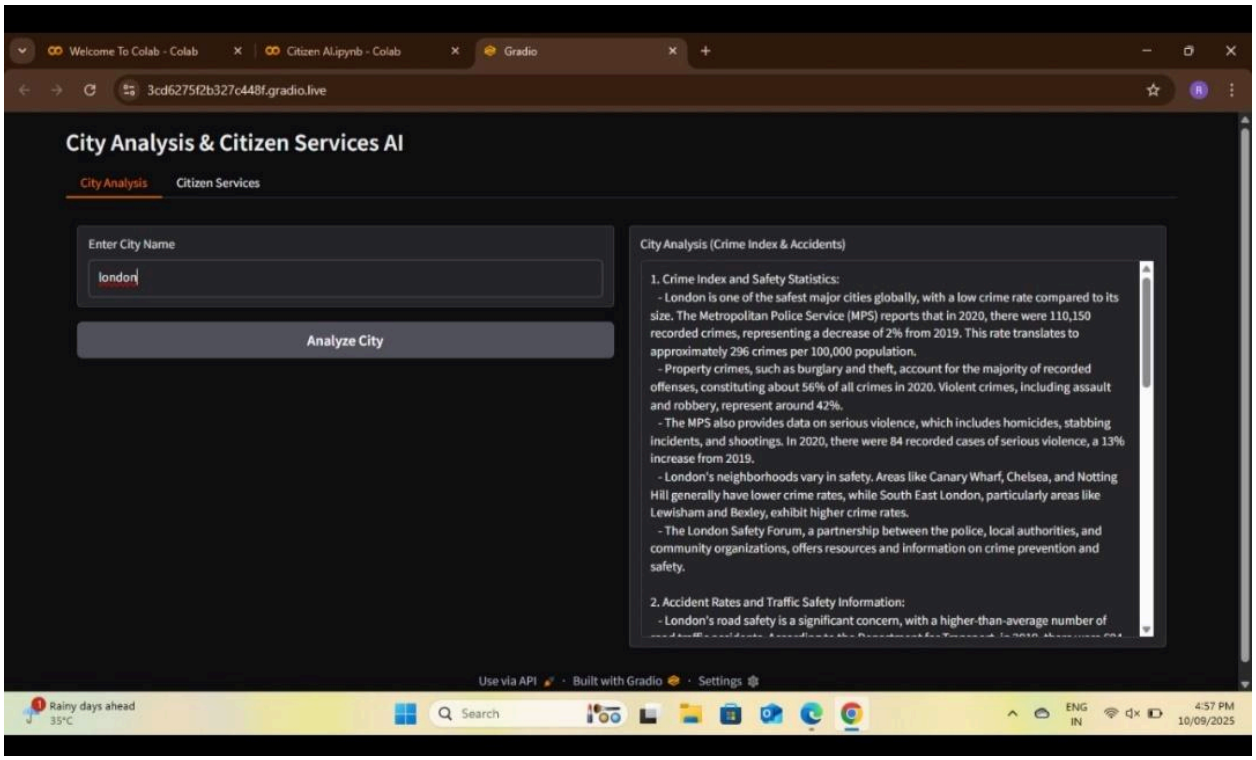
Performance: Loading a 2B-parameter model into memory can be slow and resource-intensive, especially on CPU. This can be mitigated with quantizing the model, or using GPUs.
API Latency: The time taken for the LLM to generate a response can lead to high latency.

Future Enhancements
Real-time Data Integration: Connect the backend to real-time data sources (e.g., public crime APIs, traffic data) to provide up-to-the-minute information.
Chat History: Implement a chat feature that maintains context for a more conversational experience in the "Citizen Services" tab. This would involve a session management system.
User Feedback Loop: Add a feature where users can rate the quality of the AI's response (e.g., thumbs up/down). This data can be used to fine-tune the model or improve the prompting strategy over time.
Advanced ML Modules: Integrate more complex ML models, like time-series models for forecasting crime rates or deep learning models for detecting specific anomalies in traffic patterns.

OUTPUT:




Welcome To Colab - Colab
Citizen AI.ipynb - Colab
Gradio
3cd6275f2b327c448f.gradio.live
City Analysis & Citizen Services AI
City Analysis
Citizen Services
Your Query
how can i apply for a aadhar card?
Get Information
Government Response
To apply for an Aadhaar card, you can follow these steps. Aadhaar is a unique identification number issued by the Indian government to its citizens. Here's how you can apply:
1. **Online Application:**
- Visit the official Aadhaar enrollment website: https://eaadhaar.in/
- Register as a new user by providing your basic details (name, date of birth, address, etc.) and a mobile number or an email ID.
- After registration, you'll receive a confirmation SMS or email.
- Log in to your account using the registered mobile number or email ID.
- Select the option for 'Aadhaar Enrollment' and proceed.
- You'll need to provide your basic details and biometric data (fingerprints and facial recognition) at a certified enrollment center.
2. **Offline Application:**
- Visit the nearest Aadhaar enrollment center. You can find these centers using the Aadhaar enrollment map available on the official website.
- Bring the following documents to the center:
- Proof of address (like a utility bill, rental agreement, or passbook)
Use via API · Built with Gradio · Settings
Rain showers
At night
Search
ENG
IN
5:02 PM
10/09/2025


Welcome To Colab - Colab
Citizen AI.ipynb - Colab
Gradio
3cd6275f2b327c448f.gradio.live
City Analysis & Citizen Services AI
City Analysis
Citizen Services
Enter City Name
london
Analyze City
City Analysis (Crime Index & Accidents)
1. Crime Index and Safety Statistics:
- London is one of the safest major cities globally, with a low crime rate compared to its size. The Metropolitan Police Service (MPS) reports that in 2020, there were 110,150 recorded crimes, representing a decrease of 2% from 2019. This rate translates to approximately 296 crimes per 100,000 population.
- Property crimes, such as burglary and theft, account for the majority of recorded offenses, constituting about 56% of all crimes in 2020. Violent crimes, including assault and robbery, represent around 42%.
- The MPS also provides data on serious violence, which includes homicides, stabbing incidents, and shootings. In 2020, there were 84 recorded cases of serious violence, a 13% increase from 2019.
- London's neighborhoods vary in safety. Areas like Canary Wharf, Chelsea, and Notting Hill generally have lower crime rates, while South East London, particularly areas like Lewisham and Bexley, exhibit higher crime rates.
- The London Safety Forum, a partnership between the police, local authorities, and community organizations, offers resources and information on crime prevention and safety.
2. Accident Rates and Traffic Safety Information:
- London's road safety is a significant concern, with a higher-than-average number of
Use via API · Built with Gradio · Settings
Rainy days ahead
35°C
Search
ENG
IN
4:57 PM
10/09/2025