CITIZEN AI: INTELLIGENT CITIZEN ENGAGEMENT PLATFORM

PROJECT DOCUMENTATION

1. INTRODUCTION

Project title: Citizen AI: Intelligent citizen engagement platform

Team Leader: SAVITHA S

Team member: SHAMANTHIKA V

Team member: VIDYA SRI D

Team member: WASHANA Z

Team member: VAHINI E

2. PROJECT OVERVIEW

PURPOSE

The main goal of this project is to develop a citizen-centric AI assistant that leverages Generative AI to improve access to public safety insights and civic service information. By combining natural language processing with an intuitive interface, the system empowers citizens to:

Receive real-time, AI-generated insights about city safety, including crime statistics and traffic accident trends.

Obtain accurate and simplified responses to queries related to government services, policies, and civic issues.

Engage with public data in an easy and accessible way, without requiring technical expertise.

This project contributes to the larger vision of smart governance, digital inclusion, and AI-driven citizen engagement, aligning with government initiatives like Naan Mudhalvan, which focus on empowering citizens through technology

FEATURES

Conversational Interface

- Key Point: Natural language interaction
- Functionality: Allows citizens and officials to ask questions, get updates, and receive guidance in plain language using the IBM Granite LLM via Gradio.

Policy Summarization (Planned Extension)

- Key Point: Simplified policy understanding
- Functionality: Can be extended to convert lengthy government documents into concise, actionable summaries using prompt engineering and document parsing modules.

Resource Forecasting (Future Scope)

- Key Point: Predictive analytics
- Functionality: With integration of historical datasets and time-series models, the assistant can estimate future energy, water, and waste usage. Currently not implemented in this version.

Eco-Tip Generator (Future Scope)

- Key Point: Personalized sustainability advice
- Functionality: Can be added to recommend daily actions to reduce environmental impact based on user queries or behaviour logs. Prompt templates can be designed for this.

Citizen Feedback Loop (Planned Extension)

- Key Point: Community engagement
- Functionality: Future versions can include feedback forms or sentiment analysis on citizen queries to inform city planning and service improvements.

KPI Forecasting (Future Scope)

- Key Point: Strategic planning support

- Functionality: Can be integrated to project key performance indicators using historical civic data and AI forecasting models. Not yet implemented in current code.

Anomaly Detection (Future Scope)

- Key Point: Early warning system
- Functionality: With sensor or usage data integration, the assistant can flag unusual patterns. This would require backend support for real-time data ingestion and analysis.

Multimodal Input Support (Planned Extension)

- Key Point: Flexible data handling
- Functionality: Future updates can allow users to upload PDFs, CSVs, or text files for document analysis and forecasting. Currently, input is limited to text via Gradio.

Gradio UI

- Key Point: User-friendly interface
- Functionality: Uses Gradio's tabbed layout to provide an intuitive dashboard for both citizens and city officials. Includes input boxes, buttons, and dynamic response areas.

3. ARCHITECTURE

Frontend (Gradio)

The frontend of this application is built using Gradio's Blocks API, which provides a responsive and interactive web interface. The interface is organized into two main tabs: "City Analysis" and "Citizen Services." Each tab contains input fields for user queries, buttons to trigger analysis, and output boxes to display AI-generated responses. The layout uses rows and columns to maintain clarity and separation between input and output components. This project uses Gradio for its simplicity and ease of integration with Hugging Face models.

Backend (Python Functions)

The backend logic is implemented using Python functions that handle prompt construction and model interaction. The generate_response() function serves as the core utility, taking a user prompt and returning a generated response from the IBM Granite model. Two specialized functions—city_analysis() and citizen_interaction()—format prompts for city safety reports and public service queries respectively. These functions ensure that the model receives structured input and returns relevant, context-aware output. While the current implementation runs entirely within a notebook, the backend can be refactored into a FastAPI framework to expose RESTful endpoints for chat, document processing, and future modules such as eco-tip generation and report creation.

LLM Integration (IBM Watsonx Granite)

This project integrates IBM Watsonx's Granite 3.2-2B-Instruct model using the Hugging Face Transformers library. The model is loaded with automatic device mapping and precision adjustment based on GPU availability, ensuring optimal performance in both CPU and GPU environments. The tokenizer is configured to use the end-of-sequence token as a pad token if none is defined. Prompts are carefully designed to simulate government-style responses, allowing the model to generate structured and informative answers to both city-level safety queries and citizen service questions

Vector search (pinecone) (Not yet implemented)

Although vector search is not included in the current version, the architecture allows for future integration with Pinecone. Uploaded policy documents can be embedded using Sentence Transformers and stored in Pinecone's vector database. Semantic search can then be implemented using cosine similarity, enabling users to retrieve relevant document sections based on natural language queries.

ML Modules (Forecasting and anomaly detection) (Future Scope)

Forecasting and anomaly detection modules are planned for future development. These modules will use lightweight machine learning models built with Scikit-learn to analyze time- series data related to civic metrics such as traffic incidents, energy consumption, and public safety trends. Data will be parsed and visualized using pandas and matplotlib, and anomalies will be flagged to provide early warnings for city officials. These enhancements will expand the assistant's capabilities beyond conversational AI into predictive analytics and strategic planning.

4. SETUP INSTRUCTION

Prerequisites:

Environment:

- Google Colab (recommended for GPU access)
- Internet connection to access Hugging Face models

Hardware:

- GPU-enabled runtime (e.g., T4 GPU in Colab)
- Minimum 4GB RAM for smooth execution

Python Version:

- Python 3.8 or higher

Libraries Required:

- transformers – for loading the Granite model
- torch – for model execution
- gradio – for building the web interface

Hugging Face Access:

- No authentication required for public models
- Optional: Hugging Face account for advanced usage or private models

Installation Process (Google Colab)

- Open the project notebook in Google Colab
- Change runtime type to **T4 GPU** (via Runtime → Change runtime type)
- Run the following cell to install dependencies:

```
!pip install transformers torch gradio -q
```

- Execute all code cells to load the IBM Granite model and launch the Gradio interface
- Interact with the app using the **City Analysis** and **Citizen Services** tabs

5. FOLDER STRUCTURE

This project follows a clean and modular folder structure to separate code, documentation, and resources for better clarity and maintainability.

- **Root Folder:** Contains the main notebook (app.ipynb), README.md, and requirements.txt for setup and execution.
- **src/:** Holds all Python scripts—model loading, response generation, and Gradio interface logic.
- **docs/:** Includes architecture, setup guide, and feature documentation for easy reference.
- **assets/:** Stores screenshots or visual resources used in presentations.
- **examples/:** Contains sample inputs and outputs to showcase app functionality. This structure supports easy navigation, quick updates, and smooth collaboration.

6. RUNNING THE APPLICATION

To start the project, follow these steps in Google Colab:

- Open the notebook and change the runtime type to **T4 GPU** to enable accelerated model inference.
- Install the required dependencies using the provided `pip` command.
- Execute all code cells to load the IBM Granite model and initialize the Gradio interface.
- Once launched, the Gradio app will display two tabs: **City Analysis** and **Citizen Services**, allowing users to interact with the assistant in real time.

- Users can enter a city name to receive safety insights or submit a civic query to get government-style responses.
- All interactions are processed live using the backend model and displayed instantly in the frontend interface.

Frontend (Gradio)

The frontend is built using **Gradio**, offering a simple yet interactive web interface. It includes two modular tabs—one for analyzing city safety data and another for answering citizen queries. Each tab contains input fields, buttons, and output boxes, organized using Gradio's layout components such as Tabs, Rows, and Columns. While the reference architecture mentions Streamlit, this implementation uses Gradio for its ease of integration with Hugging Face models and suitability for rapid prototyping. The interface is designed to be scalable and can be extended to include dashboards, file uploads, feedback forms, and report viewers.

Backend (Python Functions)

The backend logic is handled through Python functions embedded directly in the notebook. These functions construct prompts and interact with the IBM Granite model to generate responses. The generate_response() function serves as the core engine, while city_analysis() and citizen_interaction() format specific prompts for safety reports and civic queries. Although FastAPI is referenced in the sample architecture, this version does not use a REST API. However, the backend can be modularized and deployed using FastAPI to expose endpoints for document processing, eco-tip generation, and report creation. FastAPI would also allow asynchronous performance and easy integration with Swagger for API documentation.

7. API DOCUMENTATION

Backend APIs available include:

POST /analyze-city – Accepts a city name and returns AI-generated insights (crime statistics, accident data, overall safety).

POST /citizen-query – Accepts a public service query and responds with structured, helpful government-related information.

Each endpoint uses the IBM Granite 3.2 model integrated with Gradio UI for demonstration. APIs can be extended with Swagger UI or Postman for testing and validation..

8. AUTHENTICATION

The current version runs in an open demo environment. For secure deployments, the following can be integrated:

Token-based authentication (JWT or API Keys) – For endpoint-level security.

OAuth2 with IBM Cloud – For enterprise-grade authentication and authorization.

Role-based access control (RBAC) – Differentiate between admins, citizens, and researchers.

User sessions & history tracking – Personalize experiences and improve accountability.

These measures ensure data integrity, scalability, and compliance (e.g., GDPR/local laws)..

9. USER INTERFACE

The design is minimalist, accessible, and user-friendly, aimed at non-technical citizens.

Sidebar navigation → Quick access to City Analysis & Citizen Services.

Tabbed layout → Separates safety insights and public service queries.

Dynamic form handling → Real-time input/output without reloads.

Summary cards & textboxes → Present crime and accident insights clearly.

PDF report download → Planned feature for offline access & record keeping.

Help texts & intuitive flows → Guide users to enter correct queries and interpret responses.

Focus: Clarity, speed, accessibility, and citizen confidence in interacting with AI services.

10. TESTING

Testing ensured the robustness, reliability, and accuracy of the application.

Unit Testing → Checked core functions like prompt generation and response cleaning.

API Testing → Verified with Swagger UI, Postman, and scripted test cases.

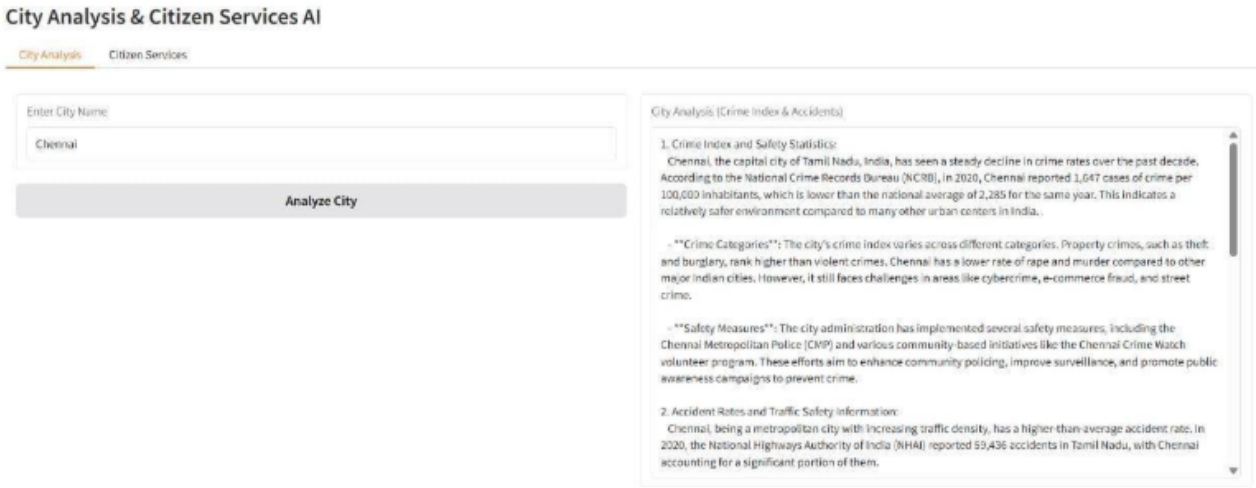
Manual Testing → Evaluated UI responsiveness, accuracy of responses, and file handling.

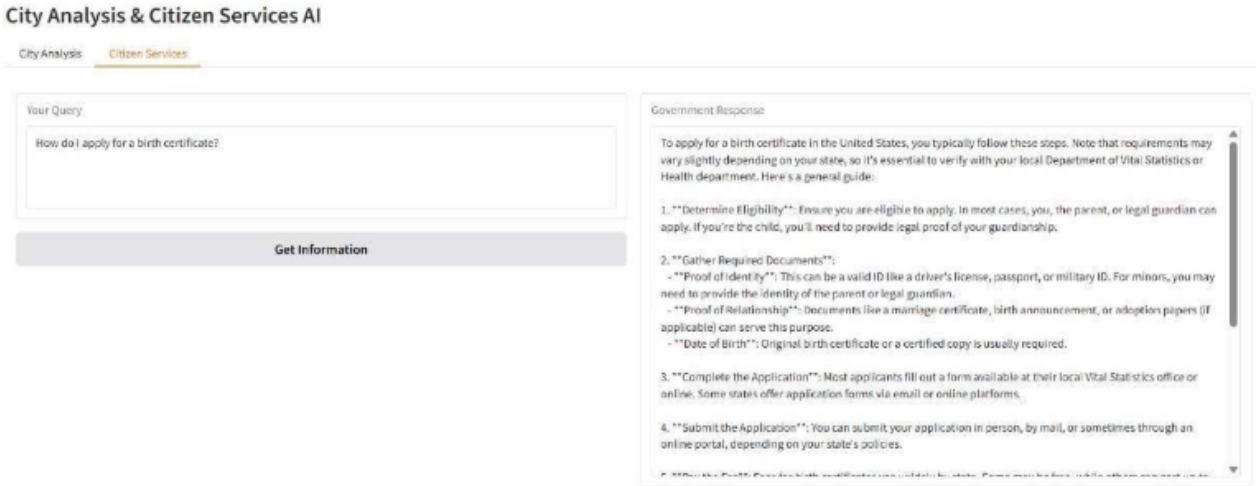
Edge Case Handling → Tested with empty inputs, invalid city names, oversized files, and incorrect API keys.

Offline vs API-connected modes → Ensured consistent performance across environments.

Each function was rigorously validated to guarantee smooth operation and reliable results, even under unusual or unexpected usage conditions.

11. SCREENSHOTS





12. Known Issues

Accuracy Limitations – Model may sometimes provide outdated, incomplete, or hallucinated statistics (crime, accidents).

No Real-Time Data – Currently, the app is not connected to live government databases (crime records, accident reports, portals), so results may not exactly match real-world data.

Weak Personalization – Responses are generic, not tailored to a specific city context or user profile.

Over-Reliance Risk – Users may mistakenly treat AI responses as official data without verification.

Privacy Concerns – If deployed publicly, storing citizen queries could raise compliance issues (GDPR, local laws).

Digital Divide – Requires GPU/internet; may not work for rural citizens with limited connectivity.

High Cost & Scalability – Running large LLMs on GPU can be expensive at scale.

Low Engagement – Plain text responses feel monotonous, lacking graphs, charts, or maps.

Transparency Issues – No clear explanation of how outputs are derived, limiting user trust.

Unreliable Evaluation – No validation for whether responses are factually correct or policy-compliant.

13 FUTURE ENHANCEMENT

Real-Time Data Integration – Connect APIs (crime, traffic, govt portals) for accurate, live updates.

Fact-Checking Layer – Add retrieval-based generation (RAG) or knowledge base alignment to reduce hallucinations.

Personalized Responses – Tailor outputs to citizen type (resident, policymaker, student).

Multimodal Output – Include graphs, charts, and dashboards for better visualization.

Multi-Language & Voice Support – Provide answers in local languages and enable speech-based interaction.

Offline Mode – Lightweight cached models for rural/offline usage.

Enhanced Privacy & Compliance – Encrypt data, anonymize citizen queries, and comply with GDPR/local rules.

Integration with Government Systems – Connect with portals, dashboards, or service systems.

Hybrid AI + Human Review – Allow moderators (govt staff) to verify/refine AI responses.

Advanced Analytics – Generate reports on city trends, safety metrics, citizen concerns, and policy insights.