

# Citizen AI: Intelligent Citizen Engagement Platform

## Project Documentation

### 1. Introduction

- Project title : Citizen AI: Intelligent Citizen Engagement Platform
- Team member : Tarika S
- Team member : Tawffeqa U
- Team member : Thulasi Sre H G

### 2. Project overview

Purpose :

The purpose of this project is to build an AI-powered citizen service and city analysis system that helps users:

- Get safety insights (crime rates, accident statistics, traffic safety) for any city.
- Interact with a government-like assistant to ask questions about public services, policies, and civic issues.

It aims to serve as a citizen support tool that provides quick, AI-generated, and informative responses through a simple web interface.

Features:

#### ☐ City Analysis Module

- Accepts a city name as input.
- Provides a detailed AI-generated report including:
  - Crime Index & safety statistics.
  - Accident rates & traffic safety information.

#### Citizen Services Module

- Allows users to enter queries related to government services.
- AI responds as a government assistant, offering clear and relevant guidance.

#### AI-Powered Text Generation

- Uses the IBM Granite 3.2 Instruct model for generating reliable, human-like responses.
- Handles prompts dynamically and adapts answers based on the input context.

### **Interactive Web Interface**

- Built with Gradio for simplicity.
- Organized with two tabs:
  - City Analysis tab for city safety reports.
  - Citizen Services tab for government/civic queries.
- Easy-to-use textboxes and buttons for interaction.

### **Scalable & Device-Friendly**

- Runs on GPU (if available) for faster responses.
- Automatically adjusts to CPU execution if GPU is not available.

### **Conversational Interface**

Key Point: Provides a natural chat-like experience between citizens and the AI.

Functionality:

- Allows continuous back-and-forth conversations.
- Retains context of previous messages for smoother dialogue.
- Helps users ask multiple follow-ups without restarting queries.

### **Policy Summarization**

Key Point: Summarizes complex government policies into simple, citizen-friendly language.

Functionality:

- Takes long policy text as input.
- Outputs a concise summary with key rules, benefits, and eligibility.
- Improves accessibility of government regulations for the public.

### **Resource Forecasting**

Key Point: Predicts future demand and allocation of public resources.

Functionality:

- Uses AI prompts to analyze trends (e.g., water, electricity, healthcare demand).
- Generates forecasts with reasoning and potential challenges.
- Assists in city planning and smart governance.

## **Eco-Tip Generator**

Key Point: Provides actionable environmental sustainability advice.

Functionality:

- Suggests eco-friendly practices for individuals and communities.
- Tips may include energy saving, waste reduction, and pollution control.
- Encourages sustainable lifestyle choices.

## **Citizen Feedback Loop**

Key Point: Collects and analyzes citizen feedback for continuous improvement.

Functionality:

- Allows users to submit feedback on city services or policies.
- AI can categorize feedback into themes (e.g., safety, transport, healthcare).
- Provides insights for government decision-makers.

## **KPI Forecasting**

Key Point: Predicts future values of key performance indicators (KPIs) for governance.

Functionality:

- Tracks KPIs such as crime rate trends, traffic congestion, or service response time.
- Uses AI to forecast future trends based on historical patterns.
- Helps authorities plan resource allocation and set improvement targets.

## **Anomaly Detection**

Key Point: Identifies unusual or abnormal patterns in civic data.

Functionality:

- Monitors inputs like citizen complaints, city safety reports, or traffic data.
- Detects spikes (e.g., sudden rise in accidents, water shortages).
- Alerts stakeholders for quick intervention.

## **Multimodal Input Support**

Key Point: Accepts different types of inputs (text, images, audio, etc

Functionality:

- Citizens can upload documents, images (e.g., potholes, pollution spots), or voice queries.
- AI processes multimodal inputs for richer analysis.

- Improves inclusivity for users with different communication preferences.

### **Streamlit or Gradio UI**

Key Point: Provides a simple and interactive interface for citizens.

Functionality:

- Gradio: Best for quick prototypes with tabs, textboxes, and shareable links.
- Streamlit: Allows more flexible dashboards, visual analytics, and KPI charts.
- Both options make the AI system accessible via the web without complex setup.

## **3. Architecture**

### **Frontend (Stream lit):**

Role: Provides the user interface for citizens to interact with the system.

Key Components:

- City Analysis Page → Enter city name → Display AI-generated safety & accident insights.
- Citizen Services Page → Enter queries about policies/services → Show government-style response.
- Extra Modules (optional): Conversational interface, policy summarizer, eco tips, KPI dashboards.

Functionality:

- Collects user input (text, queries, feedback).
- Sends requests to the FastAPI backend.
- Displays AI-generated results, charts, and summaries in a clean UI.

### **Backend (Fast API):**

- Role: Acts as the AI engine and API service layer.
- Key Components:
  - Model Loader → Loads ibm-granite/granite-3.2-2b-instruct with Hugging Face Transformers + PyTorch.
  - Endpoints:
    - city-analysis → Returns crime & accident insights for a given city.
    - citizen-query → Returns government assistant responses to queries.
    - policy-summary → Summarizes given policy text.
    - eco-tip → Generates eco-friendly suggestions.
    - kpi-forecast → Predicts future performance indicators.

- feedback → Collects and categorizes citizen feedback.
- Functionality:
  - Handles incoming requests from Streamlit frontend.
  - Runs AI model inference (using PyTorch + Transformers).
  - Sends structured responses (JSON) back to the frontend.

### **LLM Integration (IBM Watsonx Granite):**

Model Used: ibm-granite/granite-3.2-2b-instruct (Watsonx Granite LLM).

Loading: Done via Hugging Face transformers with GPU/CPU support.

Functionality:

- Tokenizes user input.
- Runs inference with Granite LLM.
- Decodes output into natural language responses.

Applications:

- City Analysis (crime & safety).
- Citizen Services (policy & civic queries).
- Extensions → Policy summarization, eco tips, KPI forecasting, chat.

Benefits: Enterprise-ready, scalable, ethical, customizable for governance tasks.

### **Vector Search (Pinecone):**

Purpose: Adds memory + semantic search for past queries, policies, and reports.

Flow:

1. Convert text to embeddings.
2. Store in Pinecone index.
3. Query Pinecone for similar vectors.
4. Provide retrieved context to Granite LLM.

Use Cases: Policy lookup, city safety history, citizen query relevance, feedback analysis.

Benefits: Scalable, fast, and ensures context-aware, factual responses.

### **ML Modules (Forecasting and Anomaly Detection):**

1. **Forecasting**
  - Purpose: Predicts future trends (crime rates, traffic, resource demand, KPIs).

- **Functionality:** Uses historical data + ML models (e.g., ARIMA, Prophet, LSTM) to generate forecasts for planning and decision-making.

## 2. **Anomaly Detection**

- **Purpose:** Identifies unusual or abnormal patterns in data.
- **Functionality:** Detects spikes or irregularities (e.g., sudden accidents, service failures, unusual feedback trends) using models like Isolation Forest or Autoencoders.

## 4. **Setup Instructions**

### **Prerequisites:**

Python 3.9+ installed

GPU with CUDA (optional, for faster inference)

Accounts/Keys if using Pinecone or IBM Watsonx API

Basic knowledge of command line

### **Installation Process:**

Set up Python → Make sure Python 3.9 or later is installed on your system.

Create a virtual environment → Keeps project dependencies isolated.

Install required libraries → Install packages like PyTorch, Transformers, Gradio, Streamlit, FastAPI, and Pinecone client.

Prepare the project files → Download or clone the project folder into your system.

Run the application → Start the backend (FastAPI) and frontend (Streamlit/Gradio) to launch the app in your browser.

## 5. **Folder Structure**

```
citizen_ai_project/  
├── backend/           # FastAPI backend (AI logic & APIs)  
│   ├── __init__.py  
│   ├── main.py        # FastAPI app entry point  
│   ├── routes/        # API routes  
│   └── city_analysis.py
```

```

|   |   | — citizen_services.py
|   |   | — policy_summary.py
|   |   | — eco_tips.py
|   |   | — forecasting.py
|   |   | — anomaly_detection.py
|   | — services/          # Core ML/LLM logic
|   |   | — granite_model.py    # IBM Granite model loader
|   |   | — vector_search.py    # Pinecone integration
|   |   | — forecasting_model.py # Forecasting logic (ARIMA/Prophet/LSTM)
|   |   | — anomaly_model.py    # Anomaly detection logic
|   |   | — utils.py            # Helper functions
| — frontend/              # Streamlit/Gradio UI
|   | — app.py              # Streamlit app main file
|   | — components/        # UI modules
|   |   | — city_tab.py
|   |   | — citizen_tab.py
|   |   | — policy_tab.py
|   |   | — eco_tab.py
|   |   | — kpi_tab.py
| — data/                  # Sample datasets
|   | — city_data.csv
|   | — policies.json
|   | — feedback.csv
| — models/                # Pretrained/fine-tuned models
|   | — granite/            # Local cache of IBM Granite (if needed)
|   | — forecasting_model.pkl
|   | — anomaly_model.pkl
| — tests/                 # Testing folder
|   | — test_api.py
|   | — test_forecasting.py
|   | — test_anomaly.py
| — requirements.txt        # Python dependencies
| — README.md              # Project documentation
| — config.yaml            # Configurations (API keys, settings)

```

## 6. Running the Application

To start the project:

Set up environment → Install Python and required libraries.

Run backend (FastAPI) → Starts the AI model API service.

Run frontend (Streamlit/Gradio) → Opens the web interface in your browser.

Use the app → Enter a city name or query to get AI-powered responses.

### **Frontend (Stream lit):**

Launch the Streamlit/Gradio app for the user interface.

It will open in your browser with tabs for City Analysis and Citizen Services (plus extra modules if added).

### **Backend (Fast API):**

Run the FastAPI server to handle AI model and API requests.

This will expose endpoints like /city-analysis and /citizen-query.

## **7.API Documentation**

Backend APIs available include:

- POST /city-analysis – Accepts a city name and returns AI-generated insights on crime index, safety statistics, accident rates, and traffic conditions.
- POST /citizen-query – Accepts a citizen query about public services or policies and responds with government-assistant–style information.
- POST /policy-summary – Takes a long policy text and provides a simplified summary for easier understanding.
- GET /eco-tip – Returns eco-friendly sustainability tips for daily practices.
- POST /forecast – Predicts future trends in KPIs such as crime rates, traffic flow, or healthcare demand.
- POST /anomaly-detection – Analyzes numerical data to detect unusual patterns or spikes.

Each endpoint can be tested and explored using Swagger UI for quick inspection and trial during development.

## **8. Authentication**

Purpose: To secure access to the backend APIs and prevent unauthorized use of AI services.

Approach:

1. API Key Authentication → Each request must include a valid API key in the headers (e.g., Authorization: Bearer <API\_KEY>).



2. User Authentication (Optional) → Citizens log in via username/password or OAuth (Google, government ID, etc.) before accessing services.
3. Role-Based Access Control (RBAC) → Different roles (e.g., admin, analyst, citizen) can have restricted permissions for sensitive endpoints like forecasting or anomaly detection.

Token Management: JWT (JSON Web Tokens) can be used for session-based authentication with expiry times for added security.

Integration: Works seamlessly with FastAPI security modules (fastapi.security) and can be extended to use OAuth2 or SSO for government-level integration.

## 9. User Interface

The interface is clean and user-friendly, focusing on accessibility for all citizens. It includes:

- Tabbed navigation for City Analysis, Citizen Services, Policy Summarization, Eco Tips, Forecasting, and Anomaly Detection
- Textboxes and buttons for user input and AI response generation
- Dynamic output areas for AI-generated text, summaries, charts, and alerts
- Real-time form handling for queries, feedback, and file uploads
- Optional charts and visualizations in Streamlit for KPI forecasting and anomaly detection

The design emphasizes clarity, simplicity, and intuitive guidance, allowing users to interact with AI-powered services efficiently without technical knowledge.


## 10. Testing

Testing was done in multiple phases:

- Unit Testing: For AI response functions, forecasting, anomaly detection, and utility scripts
- API Testing: Using Swagger UI, Postman, and automated test scripts for endpoints like /city-analysis and /citizen-query
- Manual Testing: For UI interactions including text inputs, file uploads, tab navigation, and AI output consistency
- Edge Case Handling: Malformed queries, empty inputs, large datasets, invalid API keys, and network interruptions

Each function and module was validated to ensure reliability, accuracy, and robustness in both standalone (offline) and API-connected modes.

## 11.Screen shots

**City Analysis & Citizen Services AI**

City Analysis

Citizen Services

Enter City Name

Chennai

Analyze City

City Analysis (Crime Index & Accidents)

1. Crime Index and Safety Statistics:


Chennai, the capital city of Tamil Nadu, India, has experienced a fluctuation in crime rates over the years. As of 2021, the city ranks moderately high in terms of crime when compared to other Indian metropolises. According to the National Crime Records Bureau (NCRB), Chennai's overall crime index is 183.0, placing it in the category of cities with "moderate" crime levels.

- Property Crimes: Chennai's property crime rate is relatively high at 213.0 incidents per 100,000 inhabitants, which places it in the "high" category. This includes offenses like robbery, theft, and burglary.

- Violent Crimes: The violent crime rate is moderate at 165.0 incidents per 100,000 inhabitants. This encompasses incidents such as murder, assault, and rape, although the frequency is lower compared to property crimes.

- Traffic Crimes: Although not formally categorized in the NCRB data, traffic infractions do occur, and they contribute to the city's overall crime tally.

Safety measures implemented by the Chennai Police include regular patrols, community policing initiatives, and technology-based solutions like CCTV surveillance and the use of mobile apps for reporting incidents. The city also has a robust emergency response system

**City Analysis & Citizen Services AI**

City Analysis

Citizen Services

Your Query

how to change name in aadhar card

Get Information

Government Response

To change your name in the Aadhaar card, follow these steps:

1. **\*\*Visit the Aadhaar Enrollment Authorized Agency or Enrollment Center\*\***: You can do this in person, or if there's no nearby center, you can request a home visit from the enrollment authority.

2. **\*\*Submit Required Documents\*\***: Prepare the following documents to prove your identity, address, and the reason for name change:

- Aadhaar card or any other unique identification number
- Two recent color photographs
- Proof of date of birth (like a birth certificate, passport, or any other government-issued ID)
- A signed statement (on Aadhaar's prescribed form) explaining the reason for the name change
- Two signatures on a self-declaration form
- Two affidavits from a government-recognized authority (like a teacher, priest, or police officer) confirming your identity and the need for the name change

3. **\*\*Biometric Update\*\***: You'll need to provide biometric data (fingerprints and face recognition) for the updated Aadhaar.

## 12.Known Issues

**Response Latency:** AI-generated responses can be slow for large prompts or on CPU-only systems.

**Incomplete Context Handling:** The model may occasionally miss context in long multi-turn conversations.

**Large Input Limits:** Extremely long text inputs may be truncated due to token limits of the Granite model.

**Pinecone Dependence:** Vector search functionality requires a live Pinecone account and API connection.

**Forecasting & Anomaly Accuracy:** Predictions depend on historical data quality and may not capture sudden changes.

**UI Limitations:** Gradio interface has limited visualization capabilities compared to Streamlit; charts and interactive graphs are less dynamic.

**Error Handling:** Some edge cases, like malformed inputs or network interruptions, may return generic errors instead of detailed messages.

### **13.Future enhancement**

The current system provides AI-based city analysis and citizen services through natural language interaction. While effective, several enhancements can be introduced in the future to improve accuracy, usability, and scalability:

1. **Real-Time Data Integration** o Connect the system with live government databases, open data APIs, and crime/traffic records to provide updated and accurate statistics instead of static AI generated insights.
2. **Multilingual Support** o Extend the model to handle multiple languages (e.g., Hindi, Tamil, Marathi, etc.) to make the system accessible to a wider population.
3. **Voice-Based Interaction** o Integrate speech-to-text and text-to-speech modules for voice-enabled citizen queries and responses.
4. **Mobile Application Deployment** o Develop Android/iOS apps for broader accessibility, in addition to the Gradio web interface.
5. **Personalized Recommendations** o Provide personalized insights based on user profiles, such as safety alerts for specific areas or customized government schemes.
6. **Enhanced Visualization** o Add graphs, charts, and dashboards for crime and accident statistics to improve understanding through data visualization.
7. **Feedback & Continuous Learning** o Allow users to rate responses, enabling the system to improve accuracy over time through fine-tuning and feedback loops.