

CitizenAI – Intelligent Citizen Engagement Platform

Project Documentation

1. Introduction

Project Title:

CitizenAI – Intelligent Citizen Engagement Platform

Team Members:

1. **Bharath Kumar D**
2. **Anand K**
3. **Chinna**
4. **Gokula Krishnan S**

2. Project Overview

Purpose

The purpose of **CitizenAI** is to enhance citizen–government interaction by leveraging AI and real-time data. The platform allows citizens to:

- Ask questions about government services and policies,
- Submit feedback,
- Report issues or concerns, and
- View responses instantly through a conversational AI assistant.

For government officials, CitizenAI acts as a **decision-making partner** by providing:

- Real-time sentiment analysis,
- Visualization of feedback and reported issues,
- Actionable insights to improve services and public trust.

Ultimately, CitizenAI bridges **technology, governance, and community engagement**, leading to a **more transparent, efficient, and responsive government**.

Features

- **Conversational Interface**
 - Key Point: Real-time natural language interaction
 - Functionality: Citizens can ask questions and get instant answers using an AI-powered assistant.
- **Policy Information & Summarization**
 - Key Point: Easy-to-understand information
 - Functionality: Converts government guidelines and policies into simplified, citizen-friendly responses.
- **Sentiment Analysis**

- Key Point: Public opinion tracking
 - Functionality: Analyzes citizen feedback and classifies it as Positive, Neutral, or Negative.
- **Concern Reporting**
 - Key Point: Issue escalation
 - Functionality: Allows citizens to report civic issues directly through the platform for government follow-up.
- **Dynamic Dashboard**
 - Key Point: Data-driven decision making
 - Functionality: Displays live sentiment distribution, engagement trends, and recent issues for administrators.
- **User-Friendly Interface (Flask/Gradio)**
 - Key Point: Accessibility for both citizens and government officials
 - Functionality: Clean, responsive web interface that works across devices.

3. Architecture

Frontend (Flask + HTML/CSS or Gradio for Colab Demo)

- Pages: `index.html`, `chat.html`, `dashboard.html`, `login.html`
- Responsive design for easy navigation
- Provides chat input forms, feedback forms, and dashboards

Backend (Flask Framework)

- Handles user requests (`/ask`, `/feedback`, `/concern`)
- Generates responses by calling IBM Granite model
- Stores in-memory chat history and feedback data (future-ready for database integration)

AI Layer (IBM Granite via Hugging Face)

- Natural Language Understanding & Generation
- Sentiment Analysis Pipeline

Data & Analytics

- Real-time in-memory storage of chats, concerns, feedback
- Aggregated statistics for dashboard visualization

4. Setup Instructions

Prerequisites

- Python 3.8+
- Flask web framework
- PyTorch with CUDA support (if GPU available)
- Hugging Face libraries: `transformers`, `accelerate`, `bitsandbytes`
- Internet connection (to download IBM Granite model)

Installation Steps

1. Clone or create project folder
2. Create virtual environment:

```
python -m venv venv
source venv/bin/activate # Linux/Mac
venv\Scripts\activate    # Windows
```

3. Install dependencies:

```
pip install -r requirements.txt
```

4. Run Flask app:

```
python app.py
```

5. Open `http://127.0.0.1:5000` in a browser

5. Folder Structure

```
CitizenAI/
├── app.py                # Main Flask backend
├── requirements.txt      # Dependencies
├── templates/           # HTML templates
│   ├── index.html
│   ├── chat.html
│   ├── dashboard.html
│   ├── login.html
│   └── about.html
├── static/              # Static assets
│   ├── css/styles.css
│   └── images/
└── .env                 # Environment variables (MODEL_PATH, SECRET_KEY)
```

6. Running the Application

1. Launch Flask backend (or Gradio app if using Colab).
2. Navigate to chat page, enter a query, and receive an AI-generated response.
3. Submit feedback to trigger sentiment analysis.
4. Report a concern and verify it appears on the dashboard.
5. Open dashboard page to view overall sentiment counts and reported issues.

All interactions happen in **real time**.

7. API Documentation

Available Routes

- GET / – Home page
- GET /chat – Chat interface
- POST /ask – Accepts a user query, returns AI-generated response
- POST /feedback – Accepts citizen feedback, returns sentiment analysis
- POST /concern – Logs a reported concern
- GET /dashboard – Displays aggregated sentiment data & concerns
- GET, POST /login – Handles user authentication

8. Authentication

- Session-based login/logout implemented with Flask session.
- Future enhancement: JWT-based authentication for API endpoints, and role-based access (Citizen/Admin).

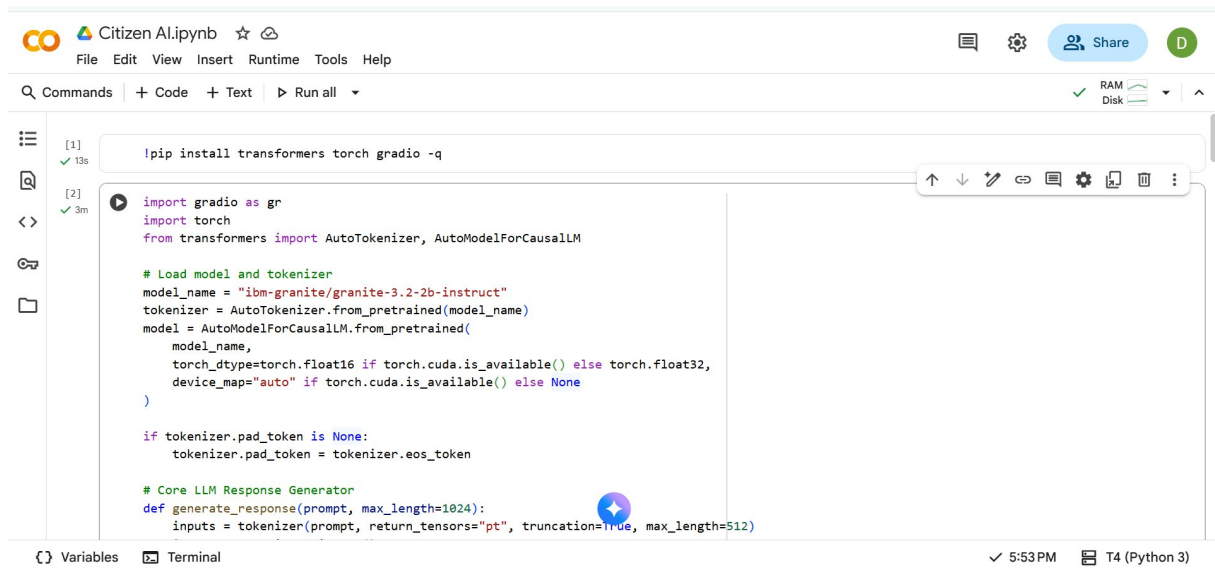
9. User Interface

- **Sidebar Navigation:** Home, Chat, Dashboard, Login/Logout
- **Chat Page:** Input box for queries, feedback form, concern reporting
- **Dashboard:** Real-time sentiment distribution, latest reported issues
- **Responsive Design:** Optimized for desktop and mobile devices

10. Testing

- **Unit Testing:** Verified AI response generation and sentiment classification
- **Integration Testing:** Tested form submissions, login/logout flow
- **Manual Testing:** Checked dashboard updates after feedback & concerns
- **Edge Case Handling:**
 - Empty input detection
 - Invalid login credentials
 - Long query handling without crash

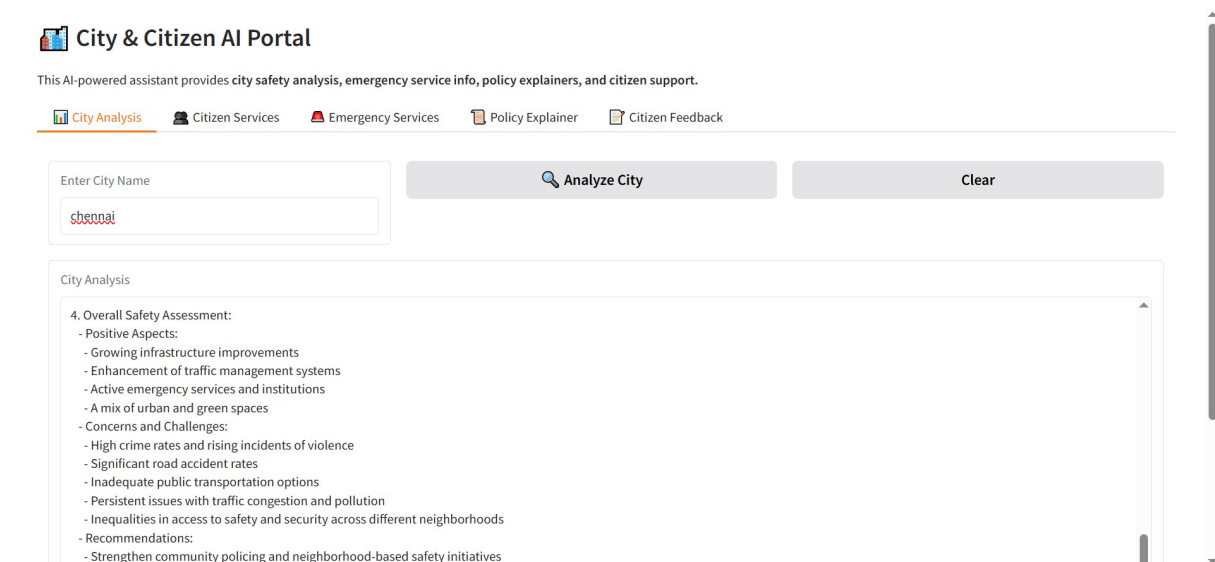
11. Screenshots



The screenshot shows a JupyterLab interface with a code editor and a terminal. The code editor contains a Python script for installing transformers and loading a model. The terminal shows the output of the script.

```
[1] ✓ 13s | !pip install transformers torch gradio -q  
[2] ✓ 3m | import gradio as gr  
import torch  
from transformers import AutoTokenizer, AutoModelForCausalLM  
  
# Load model and tokenizer  
model_name = "ibm-granite/granite-3.2-2b-instruct"  
tokenizer = AutoTokenizer.from_pretrained(model_name)  
model = AutoModelForCausalLM.from_pretrained(  
    model_name,  
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,  
    device_map="auto" if torch.cuda.is_available() else None  
)  
  
if tokenizer.pad_token is None:  
    tokenizer.pad_token = tokenizer.eos_token  
  
# Core LLM Response Generator  
def generate_response(prompt, max_length=1024):  
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
```

User Interface dashboard:



The screenshot shows the City & Citizen AI Portal dashboard. The header includes the portal name and a description. Below the header is a navigation bar with links to City Analysis, Citizen Services, Emergency Services, Policy Explorer, and Citizen Feedback. The main content area shows the City Analysis section, which includes a form to enter a city name and an Analyze City button. The results of the analysis are displayed below the form.

City & Citizen AI Portal

This AI-powered assistant provides city safety analysis, emergency service info, policy explainers, and citizen support.

[City Analysis](#) [Citizen Services](#) [Emergency Services](#) [Policy Explorer](#) [Citizen Feedback](#)

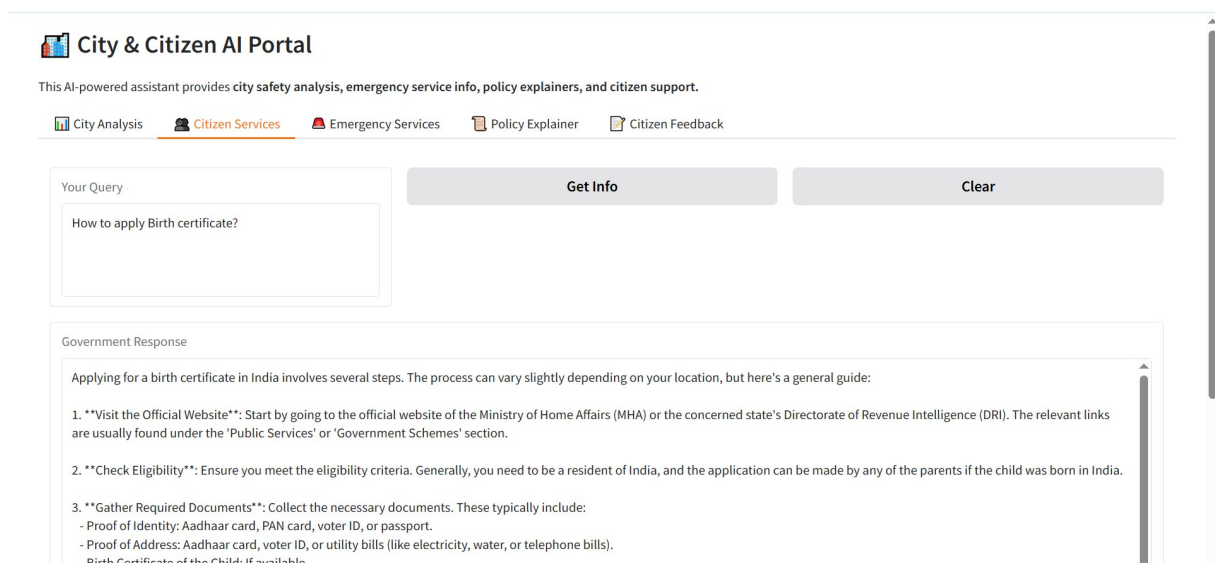
Enter City Name

[Analyze City](#) [Clear](#)

City Analysis

4. Overall Safety Assessment:

- Positive Aspects:
 - Growing infrastructure improvements
 - Enhancement of traffic management systems
 - Active emergency services and institutions
 - A mix of urban and green spaces
- Concerns and Challenges:
 - High crime rates and rising incidents of violence
 - Significant road accident rates
 - Inadequate public transportation options
 - Persistent issues with traffic congestion and pollution
 - Inequalities in access to safety and security across different neighborhoods
- Recommendations:
 - Strengthen community policing and neighborhood-based safety initiatives



The screenshot shows the City & Citizen AI Portal dashboard. The header includes the portal name and a description. Below the header is a navigation bar with links to City Analysis, Citizen Services, Emergency Services, Policy Explorer, and Citizen Feedback. The main content area shows the Citizen Services section, which includes a form to enter a query and a Get Info button. The results of the query are displayed below the form.

City & Citizen AI Portal

This AI-powered assistant provides city safety analysis, emergency service info, policy explainers, and citizen support.

[City Analysis](#) [Citizen Services](#) [Emergency Services](#) [Policy Explorer](#) [Citizen Feedback](#)

Your Query

[Get Info](#) [Clear](#)

Government Response

Applying for a birth certificate in India involves several steps. The process can vary slightly depending on your location, but here's a general guide:

1. **Visit the Official Website**: Start by going to the official website of the Ministry of Home Affairs (MHA) or the concerned state's Directorate of Revenue Intelligence (DRI). The relevant links are usually found under the 'Public Services' or 'Government Schemes' section.
2. **Check Eligibility**: Ensure you meet the eligibility criteria. Generally, you need to be a resident of India, and the application can be made by any of the parents if the child was born in India.
3. **Gather Required Documents**: Collect the necessary documents. These typically include:
 - Proof of Identity: Aadhaar card, PAN card, voter ID, or passport.
 - Proof of Address: Aadhaar card, voter ID, or utility bills (like electricity, water, or telephone bills).
 - Birth Certificate of the Child: If available.

12. Known Issues

- Running Granite model on CPU is slow – GPU recommended
- Currently uses in-memory storage (data lost after restart)
- No persistent user session history yet

13. Future Enhancements

- Migrate to database for permanent storage
- Add voice-based query input and multi-language support
- Integrate SMS/WhatsApp for issue reporting
- Deploy on cloud for production-level scalability
- Role-based access for government departments