Citizen AI Project - Detailed Documentation

City Analysis & Citizen Services AI

Project Report

Project Team Members

1. MANIMEKALAI A
2. KOWSALYA R
3. SRIDHARANI R
4. KIRUTHIKA A

1. Introduction

1.1 Project Title: Citizen AI - City Analysis & Citizen Services

1.2 Background:

In modern smart cities, citizens require real-time access to safety information, public services, a

government policies. At the same time, officials need tools to process, summarize, and an large

volumes of data. Citizen AI aims to bridge this gap.

1.3 Objectives:

- Provide city safety insights (crime and accident data).
- Act as a government assistant for policy-related queries.
- Ensure easy accessibility through a user-friendly interface.

2. Project Overview

2.1 Purpose:

The purpose of the Citizen AI project is to empower cities and residents with an AI-pov assistant

capable of providing real-time safety assessments and responding to citizen queries.

2.2 Features:

- Conversational Interface: Natural language interaction.

- City Analysis: Provides statistics about crime index, accident rates, and safety.

- Policy Summarization: Summarizes long policy documents.

- Citizen Query Response: Answers questions about services and civic issues.

- User-Friendly Interface: Uses Gradio for dashboards.

3. Architecture

3.1 Frontend (Gradio): Provides interactive web-based UI with tabs.

3.2 Backend (Hugging Face Transformers + PyTorch): Runs AI model for responses.

3.3 Model Integration: IBM Granite model is used for text generation and analysis.

3.4 Functions:

- generate_response: Generates AI responses.

- city_analysis: Provides city safety analysis.

- citizen_interaction: Responds to citizen queries.

4. Setup Instructions

4.1 Prerequisites:

- Python 3.8 or later

- Google Colab (T4 GPU preferred)

- Libraries: transformers, torch, gradio

4.2 Installation Process:

1. Open Google Colab.

2. Change runtime to GPU (T4 preferred).

3. Install libraries with !pip install transformers torch gradio -q.

4. Paste the project code into Colab.

5. Run all cells to launch.

6. Access the public link for interaction.

5. Folder Structure

app/ – Backend logic

app/api/ – API routes

ui/ – Frontend components

smart_dashboard.py – Entry script

granite_llm.py – Model integration

document_embedder.py – Document embeddings

6. Running the Application

6.1 Steps to Run:

➢ Launch Colab notebook.

➢ Run installation and model setup.

➢ Start Gradio interface.

➢ Use City Analysis tab for safety data.

➢ Use Citizen Services tab for queries.

7. API Documentation

7.1 Available Functions:

generate_response(prompt)

city_analysis(city_name)

citizen_interaction(query)

8. Authentication & Security

Future deployments can include:

- Token-based authentication (JWT, API Keys).
- OAuth2 integration.
- Role-based access control (Admin, Citizen, Researcher).

9. User Interface

9.1 City Analysis Tab: Input city name → Safety insights.

9.2 Citizen Services Tab: Input query → Government response.

9.3 Output: Clear textboxes with results.

10. Testing

10.1 Unit Testing: Functions tested independently.

10.2 API Testing: Using test inputs.

10.3 Manual Testing: Validating analysis and queries.

10.4 Edge Case Handling: Empty queries, invalid city names.

11. Screenshots

[Placeholder for Gradio interface screenshots]

12. Known Issues

- Requires stable internet connection.
- AI-generated outputs may vary.
- Limited accuracy without real-time datasets.

13. Future Enhancements

- Integration with real-time city data.
- Advanced analytics and forecasting.
- Multilingual support.
- Mobile app interface.

14. Conclusion

The Citizen AI Project demonstrates how AI models and user-friendly interfaces ca
and officials in decision-making, safety monitoring, and public service managemen
foundation for future smart city applications.