Citizen AI: Intelligent Citizen Engagement Platform

Team Leader: Akash
Team Members: Ajith Kumar, Mano, Selevaperumal

1. Introduction

Citizen AI is designed to help citizens interact with government services through an AI-powered assistant. By using the IBM Granite model from Hugging Face, the system provides quick, reliable answers to civic-related queries and performs city analysis (crime index, accident rates, and safety statistics).

2. Project Overview

The purpose of Citizen AI is to help citizens access accurate information about government services and policies while also analyzing city-level safety data. It bridges the gap between governments and citizens by providing transparent, accessible, and AI-driven engagement.

3. Problem Statement

• Citizens struggle to navigate lengthy government documents. • City safety data is often scattered across multiple sources. • Lack of a simple platform for citizen-government engagement.

4. Objectives

• Build a Gradio-based citizen assistant. • Integrate IBM Granite model for query answering and city analysis. • Provide an interface with two tabs: City Analysis and Citizen Services.

5. Literature Review

AI in Governance: AI improves citizen engagement, automates responses, and provides decision-making support. NLP for Public Data: NLP can summarize large civic policy documents and safety reports. Existing Solutions: Most chatbots provide general help but lack specialized city-level analysis and government integration.

6. Methodology

Tools & Technologies: - Language: Python - Libraries: Transformers, Torch, Gradio - AI Model: IBM Granite (granite-3.2-2b-instruct) - Hosting: Google Colab with T4 GPU Workflow: 1. Input: User enters query / city name 2. AI Model: Granite generates response 3. Output: City analysis OR government service info

7. System Architecture

+------------------+ +-----------------------+ +-------------------+ | User Input | ---> | IBM Granite Model | ---> | AI Response | | (City / Query) | | (LLM Inference) | | (Analysis/Services) | +------------------+ +-----------------------+ +-------------------+

8. Implementation

Step 1: Setup Environment → !pip install transformers torch gradio -q Step 2: Load Model → Uses Hugging Face Granite model with GPU acceleration. Step 3: Functions → city_analysis(city_name): Provides crime & accident data analysis citizen_interaction(query): Answers citizen queries Step 4: Gradio Interface → Two tabs (City Analysis and Citizen Services).

9. Sample Outputs

Example 1 – City Analysis: Input: Mumbai Output: Crime Index: Medium-high, Accident Rate: Increasing in urban areas, Overall Safety: Needs improvement. Example 2 – Citizen Query: Input: What is the procedure for getting a voter ID? Output: Visit official election commission portal, Fill Form 6, Upload proofs, Get voter ID after review.

10. Results & Discussion

• Successfully integrates Granite LLM for real-time responses. • Provides dual functionality: city analysis + citizen query resolution. • Helps both citizens and officials by improving engagement and transparency.

11. Advantages

• Lightweight and runs on Google Colab. • Easy to use with Gradio interface. • Covers both city-level safety analysis and citizen services.

12. Limitations

• Relies on Hugging Face Granite model availability. • City analysis depends on AI's knowledge (not live data). • May not capture all government policy nuances.

13. Future Enhancements

• Add real-time city data integration (crime/traffic APIs). • Support regional languages (Tamil, Hindi, etc.). • Deploy on Hugging Face Spaces / Cloud for 24x7 availability. • Add dashboards with visualizations for officials.

14. Conclusion

Citizen AI demonstrates how AI + NLP can support citizens and governments in improving transparency and accessibility. It encourages digital governance, improves city safety awareness, and simplifies public service queries.

15. Program Code

```
!pip install transformers torch gradio -q

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

def generate_response(prompt, max_length=1024):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}
with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
response = response.replace(prompt, "").strip()     return response

def city_analysis(city_name):
    prompt = f"Provide a detailed analysis of {city_name} including:\n1. Crime Index and safety statistics\n2. Accident r
return generate_response(prompt, max_length=1000)

def citizen_interaction(query):
    prompt = f"As a government assistant, provide accurate and helpful information about the following citizen query re
return generate_response(prompt, max_length=1000)
# Create Gradio interface with gr.Blocks() as app:
gr.Markdown("# City Analysis & Citizen Services AI")

    with gr.Tabs():         with
gr.TabItem("City Analysis"):
```

```
with gr.Row():                with
gr.Column():
                city_input = gr.Textbox(label="Enter City Name", placeholder="e.g., New York, London, Mumbai...", lines
analyze_btn = gr.Button("Analyze City")                with gr.Column():
                city_output = gr.Textbox(label="City Analysis (Crime Index & Accidents)", lines=15)
analyze_btn.click(city_analysis, inputs=city_input, outputs=city_output)

        with gr.TabItem("Citizen Services"):
with gr.Row():                with
gr.Column():
                citizen_query = gr.Textbox(label="Your Query", placeholder="Ask about public services, government poli
query_btn = gr.Button("Get Information")                with gr.Column():
                citizen_output = gr.Textbox(label="Government Response", lines=15)

query_btn.click(citizen_interaction, inputs=citizen_query, outputs=citizen_output) app.launch(share=True)
```

16. Output

• Tab 1: City Analysis → Provides crime & accident insights.

• Tab 2: Citizen Services → Provides policy/service-related answers.