

PROJECT REPORT
ON
Citizen AI
Intelligent Citizen Engagement Platform

Organized by
Smart Bridge - IBM Skillsbuild

During the academic year

June 2025 to October 2025

Submitted by

TEAM ID : NM2025TMID00120

ALJUVAIREYA P [212302667]

NITHYA SHREE S [212302689]

DHIVYA DHARSHINI M [212302673]

USHA A [212302712]

DEPARTMENT OF COMPUTER APPLICATIONS



VALLIAMMAL COLLEGE FOR WOMEN

E-9, Anna Nagar East, Chennai – 60010

INDEX

S NO	TITLE	PG NO
1	INTRODUCTION	1
2	ARCHITECTURE	3
3	TECHNICAL PREREQUISITES	5
4	MODULES AND FEATURES	7
5	PROJECT WORKFLOW	9
6	EXECUTION AND RESULTS	13
7	CONCLUSION	17

Citizen AI
Intelligent Citizen
Engagement
Platform

1. INTRODUCTION

1.1 PROJECT DESCRIPTION :

Citizen AI is an intelligent citizen engagement platform designed to revolutionize how governments interact with the public. Leveraging flask, ibm granite models, and ibm watson, citizen ai provides real-time, ai-driven responses to citizen inquiries regarding government services, policies, and civic issues. The platform integrates natural language processing (nlp) and sentiment analysis to assess public sentiment, track emerging issues, and generate actionable insights for government agencies. A dynamic analytics dashboard offers real-time visualizations of citizen feedback, helping policymakers enhance service delivery and transparency. By automating routine interactions and enabling data-driven governance, citizen ai improves citizen satisfaction, government efficiency, and public trust in digital governance.

Real-Time Conversational AI Assistant :

The Real-Time Conversational AI Assistant in Citizen AI serves as the primary interface for citizen interaction. It allows users to engage with public services naturally by typing questions or requests. The system captures user input in real-time and immediately sends it to a powerful underlying AI model, such as IBM Granite. This model processes the query and generates a relevant, human-like response on the fly. The assistant then displays this response back to the user almost instantly, facilitating quick access to information, support, and the ability to perform tasks like reporting issues, 24/7. It aims to provide a seamless and efficient conversational experience for civic engagement.

Citizen Sentiment Analysis :

Citizen Sentiment Analysis in Citizen AI is a core feature designed to understand the public's feelings about government services and related topics. It works by analysing text input, whether from direct citizen feedback submitted through the platform or potentially from other digital interactions (though the current implementation focuses on submitted text). Using AI (like the simple analyse_sentiment function in app.py), the system classifies the sentiment of the text as Positive, Neutral, or Negative. This process helps the government quickly identify areas of public satisfaction or concern. By aggregating sentiment data, the platform provides valuable insights into overall citizen mood and highlights specific issues that may need

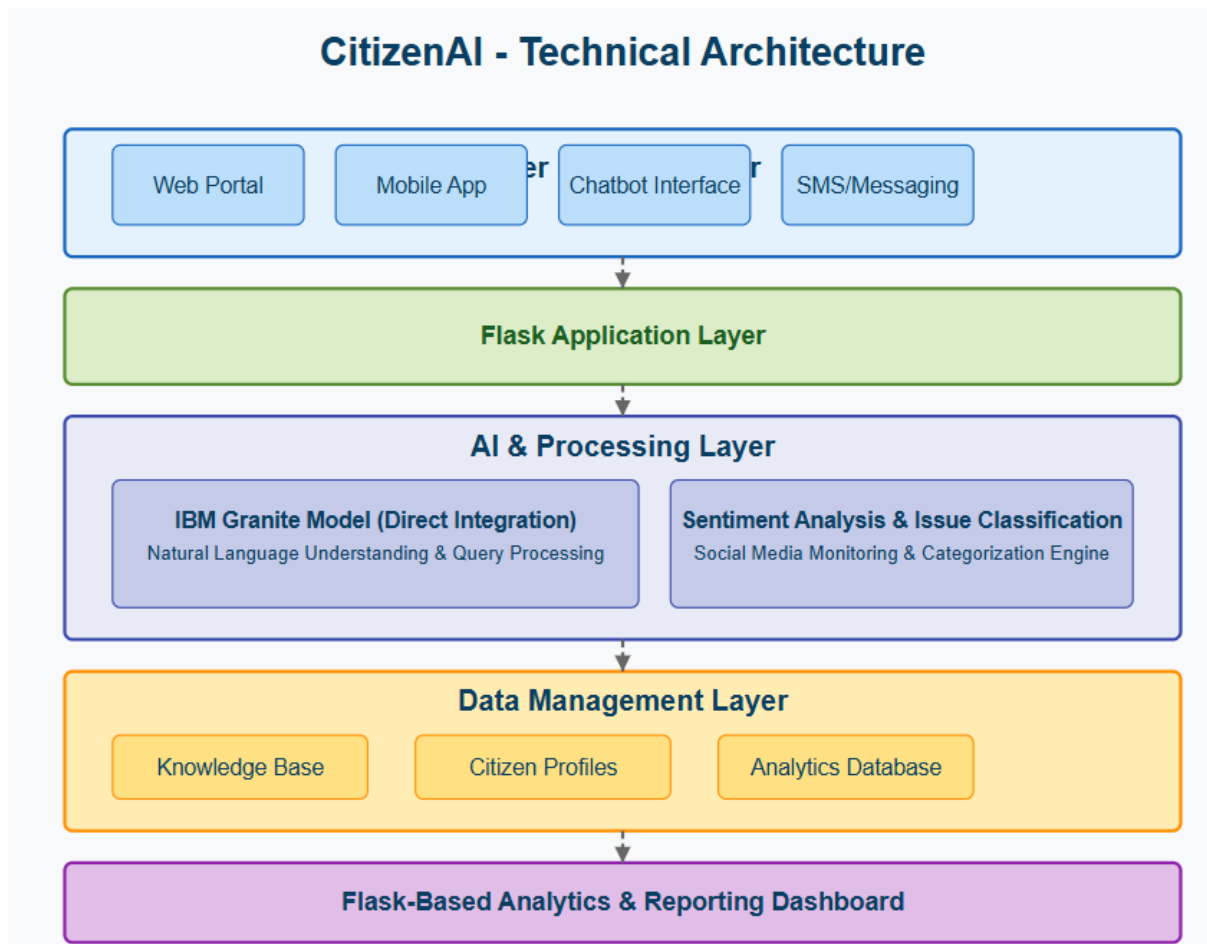
attention, ultimately aiming to improve service delivery and citizen satisfaction. The results are presented on the dashboard for easy monitoring.

Dynamic Dashboard :

The Dynamic Dashboard in Citizen AI serves as a central hub for government officials to gain real-time insights into citizen feedback and interactions. It visualizes key data points, including the overall citizen sentiment (positive, neutral, negative) derived from submitted feedback. The dashboard also tracks interaction trends over time, showing peak periods of activity. Furthermore, it can display aggregated government service ratings or issues reported by citizens. By presenting this information dynamically through charts and clear metrics, the dashboard empowers government departments to quickly understand public perception, identify areas needing improvement, and make data-driven decisions to enhance public services and citizen satisfaction. It transforms raw interaction data into actionable intelligence for a more responsive government.

2. ARCHITECTURE

The Citizen AI – Intelligent Citizen Engagement Platform is designed with a three-layer architecture: the AI Model Layer, the Application Logic Layer, and the User Interface Layer. At the AI Model Layer, the system integrates the IBM Granite-3.2-2B Instruct model from Hugging Face, which acts as the backbone for natural language understanding and text generation. The tokenizer encodes raw user input into tokenized tensors, and the model processes these inputs to generate meaningful responses. The design allows for dynamic hardware adaptation by checking for CUDA availability; if a GPU is present, computation is automatically mapped to it, ensuring faster inference, while falling back to CPU if GPU is unavailable.



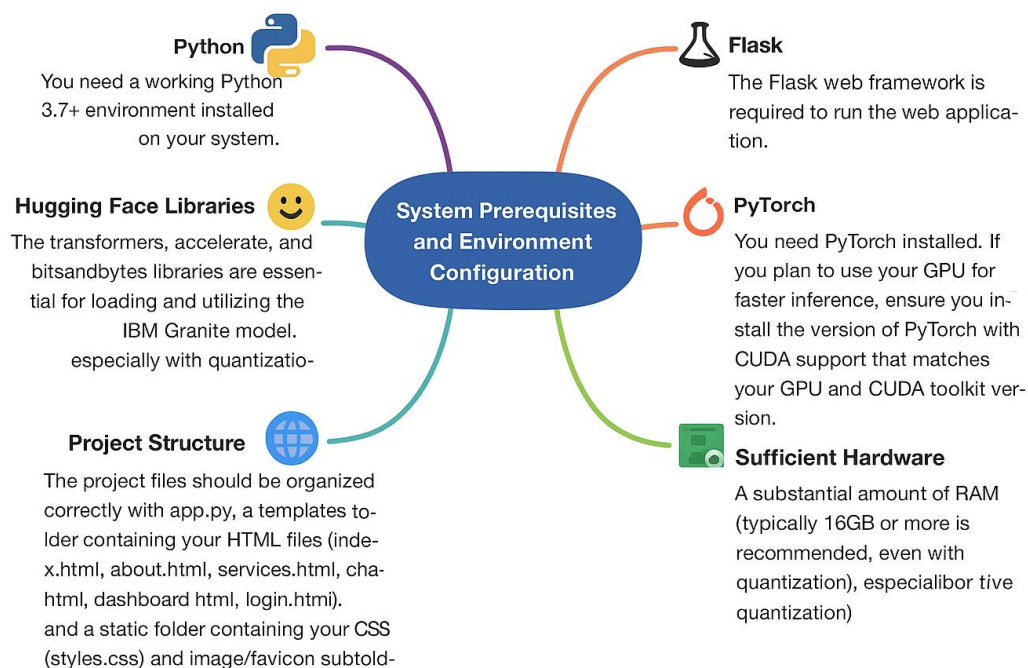
The Application Logic Layer handles the core functional modules, encapsulated in Python functions. The `generate_response` function acts as a generic response generator that manages prompt construction, model inference, and output decoding. Two domain-specific modules—`city_analysis` and `citizen_interaction`—leverage this generic function. The City Analysis module takes a city name, builds a structured prompt, and produces insights such as crime index, accident statistics, and safety evaluations. The Citizen Interaction module accepts user queries about public services, policies, or civic issues and responds with contextually relevant government-assistant-style answers. This layer essentially separates business logic from the AI model, making it reusable and extensible for future modules.

At the User Interface Layer, the system uses Gradio's Blocks API to create a structured, interactive web-based interface. The UI is organized into two tabs: "City Analysis" and "Citizen Services." Each tab contains textboxes for user inputs and multi-line output fields for displaying responses, with buttons (Analyze City, Get Information) connected to their respective backend functions through Gradio's `.click` event binding. This event-driven design ensures real-time interaction, as user inputs are directly passed to the backend logic, and results are rendered instantly in the frontend. Finally, the application is deployed through `app.launch(share=True)`, which not only allows local execution but also generates a publicly shareable link, enabling external access without the need for complex server setups. The modularity of this architecture allows for easy enhancements, such as integrating a database for logging queries, adding authentication for secure access, or expanding to a multilingual interface.

3. TECHNICAL PREREQUISITES

The Citizen AI – Intelligent Citizen Engagement Platform requires certain prerequisites to function smoothly. At its foundation, the system depends on a stable Python environment, preferably version 3.7 or higher, since both the artificial intelligence model and the application logic are implemented in Python. The web interface is powered by the Flask framework, which acts as the backbone for routing, rendering HTML templates, and managing interactions between the user and the backend. Since the project integrates a deep learning model, PyTorch is a critical requirement, as it provides the computational framework necessary for running inference. When a GPU is available, PyTorch must be installed with CUDA support so that model computations can be accelerated; otherwise, the application will rely on the CPU, where response times may be considerably slower.

System Prerequisites and Environment Configuration



The platform also relies on several Hugging Face libraries to enable smooth integration of the IBM Granite model. The Transformers library provides access to pretrained models and handles tokenization and text generation tasks. Accelerate optimizes device allocation and ensures that resources are used efficiently, while Bitsandbytes supports quantization, which reduces memory consumption and makes it possible to run large models on machines with

more modest hardware. Because of the size of the IBM Granite model, the application demands significant hardware resources. At least 16 GB of RAM is recommended to handle model loading and inference without performance issues, and while CPU execution is possible, an NVIDIA GPU with at least 8 GB of VRAM is strongly preferred for practical speeds. To make use of GPU acceleration, compatible CUDA drivers must be correctly installed and configured.

Another important requirement is a reliable internet connection, particularly during the initial setup. The IBM Granite model files are downloaded directly from the Hugging Face Hub the first time the program is executed, and these files are then cached locally for future use. Finally, the project must adhere to a clear file structure for organization and maintainability. The main application script is named `app.py`, while the `templates` directory contains the HTML files that make up the user interface, including pages such as `index.html`, `about.html`, `services.html`, `chat.html`, `dashboard.html`, and `login.html`. In parallel, the `static` directory is reserved for styling and assets, holding the CSS file (`styles.css`) as well as subfolders like `static/Images` and `static/Favicon` for storing images and favicon files. This structured setup ensures that the application runs reliably, remains scalable, and can be easily extended in the future.

4. MODULES AND FEATURES

4.1 Purpose of the Platform :

The primary purpose of Citizen AI – Intelligent Citizen Engagement Platform is to act as a bridge between citizens and government systems, ensuring that reliable information is easily accessible without unnecessary complexity. In today's fast-paced environment, citizens often struggle to navigate multiple portals, government websites, and official documents to find accurate information about policies, safety measures, and public services. Citizen AI addresses this challenge by providing a single, intelligent, AI-powered platform that simplifies access to information.

Purpose and Scope of Citizen AI – Intelligent Citizen Engagement Platform

PURPOSE



Empower citizens with real-time insights about their city





Provide a virtual government assistant





Enhance transparency and trust between governments and citizens

SCOPE

City Analysis Module

-  Safety indicators such as crime index, accident rate, etc.
-  Structured reports

Citizen Services Module

-  Answers to queries on govt. schemes and civic services
-  Contextually relevant responses

This platform aims to :

- It empower citizens with real-time insights about their city, including crime statistics, accident trends, and safety assessments.
- It provide a virtual government assistant that can answer queries about public services, government policies, welfare schemes, and civic issues in a conversational manner.
- It enhance transparency and trust between governments and citizens by offering accurate, unbiased, and consistent information.

- It reduce the information gap for citizens in rural, semi-urban, and urban areas, making governance more inclusive.
- It serve as a prototype model for integrating AI into e-governance systems, with potential for expansion into mobile applications, multilingual interfaces, and real-time data integration.

4.2 Scope of the Platform :

The scope of Citizen AI is designed to cover two major functional areas while keeping room for future enhancements:

1. City Analysis Module :

- It provides a comprehensive overview of any city by analysing key safety indicators such as crime index, accident rates, and traffic safety statistics.
- It generates structured reports to help citizens assess safety conditions before traveling, relocating, or making policy-related decisions.
- It can be extended to integrate live datasets from government crime databases, transport departments, and smart city initiatives.

2. Citizen Services Module :

- It acts as a virtual civic assistant that responds to citizen queries about government schemes, public services, and civic issues.
- It delivers information in natural, easy-to-understand language using the IBM Granite AI model.
- It has potential to evolve into a multilingual, voice-enabled assistant, allowing citizens across regions to interact in their native languages.

5. PROJECT WORKFLOW

5.1 Library Requirements :

1. Gradio :

- A Python library for creating interactive UIs for machine learning models.
- Allows building web-based apps with text boxes, buttons, sliders, etc.

2. PyTorch (torch) :

- An open-source deep learning framework used for training and running AI models.
- Provides tensor operations, GPU acceleration, and model deployment tools.

3. Transformers (Hugging Face) :

- A library for using pretrained NLP models like IBM Granite, GPT, BERT, etc.
- Provides AutoTokenizer (for input text processing) and AutoModelForCausalLM (for text generation).

4. Accelerate :

- A Hugging Face library that makes it easier to run models on multiple devices (CPU/GPU) efficiently.

5. Bitsandbytes :

- A library for quantization (e.g., 4-bit, 8-bit) to reduce memory usage of large models.
- Useful for running big AI models on limited GPU/CPU hardware.

5.2 Code Explanation :

1. Imports :

```
import gradio as gr
```

```
import torch
```

```
from transformers import AutoTokenizer, AutoModelForCausalLM
```

- Loads the required libraries : Gradio (for UI), Torch (for running the model), and Hugging Face Transformers (for tokenization and model loading).

2. Model & Tokenizer Loading :

```
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

)
```

- Loads IBM Granite model from Hugging Face Hub.
- Uses GPU (with float16 precision) if available, otherwise falls back to CPU (float32).
- device_map="auto" automatically assigns model layers to the best available hardware.

3. Pad Token Handling :

```
if tokenizer.pad_token is None:

    tokenizer.pad_token = tokenizer.eos_token
```

- Ensures that the model can handle padding properly.

4. Response Generation Function :

```
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

```

- Tokenizes input text → runs model inference → decodes model output.
- Uses temperature=0.7 for balanced creativity and relevance.
- Cleans up output by removing prompt text.

5. City Analysis Function :

```

def city_analysis(city_name):

    prompt = f"Provide a detailed analysis of {city_name} ..."

    return generate_response(prompt, max_length=1000)

```

- Prepares a structured prompt for the AI to analyze crime, accidents, and safety for a given city.

6. Citizen Services Function :

```

def citizen_interaction(query):

    prompt = f"As a government assistant, provide information about the following..."

    return generate_response(prompt, max_length=1000)

```

- Formats a query for the AI to answer citizen-related questions (policies, services, issues).

7. Gradio Interface :

```
with gr.Blocks() as app:
    gr.Markdown("# City Analysis & Citizen Services AI")
    with gr.Tabs():
        with gr.TabItem("City Analysis"):
            ...
        with gr.TabItem("Citizen Services"):
            ...
    app.launch(share=True)
```

- Creates a two-tab interface :
 - **City Analysis Tab** → User enters city → AI provides analysis.
 - **Citizen Services Tab** → User enters query → AI provides government-like response.
- share=True generates a temporary public URL so others can test it.

6. EXECUTION AND RESULTS

6.1 Program Code :

```
# run this project file in google collab by changing run type to T4 GPU

!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\nsafety statistics\n2. Accident rates and traffic safety information\n3. Overall safety\nassessment\n\nCity: {city_name}\nAnalysis:"

    return generate_response(prompt, max_length=1000)

def citizen_interaction(query):

    prompt = f"As a government assistant, provide accurate and helpful information about the\nfollowing citizen query related to public services, government policies, or civic\nissues:\n\nQuery: {query}\nResponse:"

    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=1

                )

                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 policies, civic issues...",

                    lines=4

                )

```

```
query_btn = gr.Button("Get Information")
```

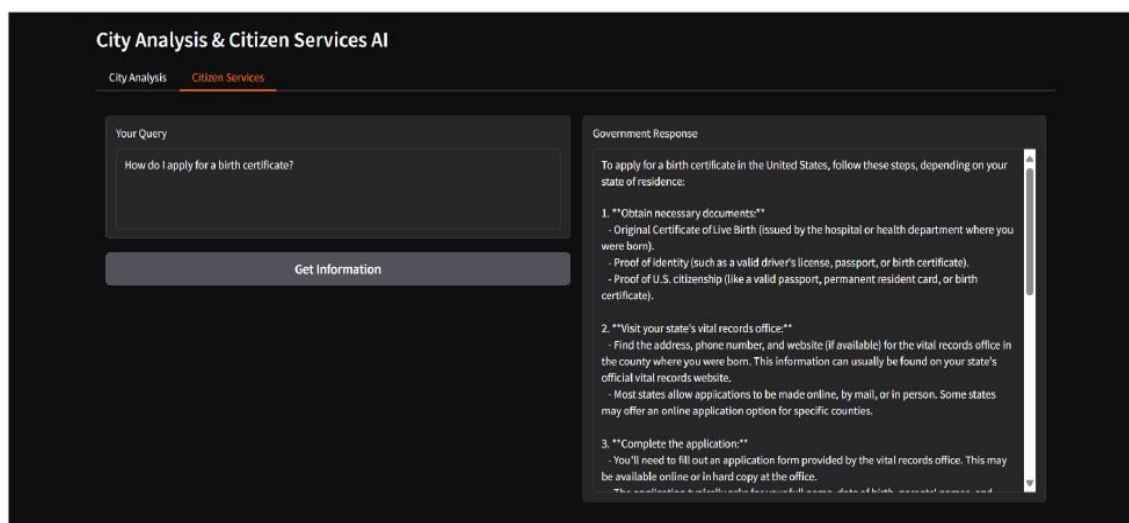
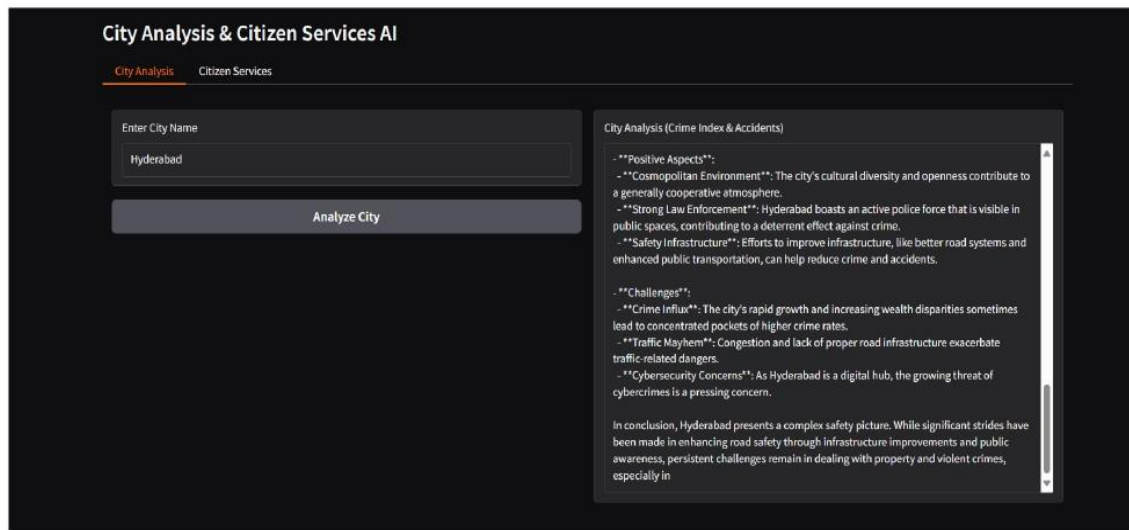
```
with gr.Column():
```

```
citizen_output = gr.Textbox(label="Government Response", lines=15)
```

```
query_btn.click(citizen_interaction, inputs=citizen_query, outputs=citizen_output)
```

```
app.launch(share=True)
```

6.2 Output :



7. CONCLUSION

The Citizen AI – Intelligent Citizen Engagement Platform is a significant step toward integrating artificial intelligence into civic administration and public interaction. The platform not only facilitates efficient communication between citizens and authorities but also empowers users to access services, provide feedback, and stay informed in real-time.

By incorporating advanced AI algorithms, natural language processing, and an intuitive interface, the platform ensures accuracy, responsiveness, and ease of use. It reduces administrative delays, enhances transparency, and promotes participatory governance. Moreover, the system's modular and scalable design allows for future upgrades, integration with additional services, and adaptation to different regions or governmental needs.

Overall, Citizen AI demonstrates the transformative potential of technology in public service. It encourages citizen engagement, strengthens trust between authorities and the community, and provides a sustainable solution for smarter urban management. With continuous improvements and user-focused innovations, Citizen AI can serve as a model for next-generation digital governance platforms, ultimately fostering a more informed, connected, and proactive society.