Citizen AI: Intelligent Citizen Engagement Platform

Project Documentation

1.Introduction:

- Project Title: Citizen AI: Intelligent Citizen Engagement Platform
- Team Member: Alnasreen Fathima R
- Team Member: Berlin M
- Team Member: Bhagyasree P
- Team Member: Brindha K

2. Project Overview:

Purpose:

The project's main objective is to create an AI assistant that can serve two important roles:

1.City Safety Analysis – Provide detailed insights into a city's safety profile (crime index, accident statistics, overall safety).

2. Citizen Services Assistant – Allow citizens to interact with the system by asking questions about public services, civic policies, or governance.

Features:

City Analysis

- Generates structured analysis about crime, accidents, and safety ratings.
- Uses AI-driven text generation to make the data readable and informative.

Citizen Services Chatbot

- Responds to queries about public services, policies, or civic issues.
- Provides helpful and natural responses.
- Interactive Gradio Interface
- Two tabs: “City Analysis” and “Citizen Services.”
- Clean and simple UI for non-technical users.

AI-Powered Responses

- Powered by IBM Granite LLM (granite-3.2-2b-instruct).
- Generates contextual, human-like answers.

3. Architecture

The system has a modular design consisting of three main parts:

Frontend (Gradio):

Provides a web-based interface where users can enter inputs and view outputs.

Offers two modes of interaction: City Analysis and Citizen Services.

Handles user-friendly layouts with textboxes and buttons.

Backend (Transformers + PyTorch):

Manages communication with the Granite LLM.

Handles tokenization, model inference, and response decoding.

Includes logic for generating prompts for both city analysis and queries.

Model Integration (IBM Granite):

Uses IBM Granite 3.2-2b-instruct model for natural language processing.

The model is capable of understanding queries and generating detailed responses.

4. Setup Instructions:

Prerequisites:

- Python 3.9 or later
- pip and virtual environment tools
- Libraries: torch, transformers, gradio

- Internet connection (to fetch and run the IBM Granite model)

Installation Process:

1. Clone or download the project files.

2. Install dependencies:

pip install torch transformers gradio

3. Run the project:

python app.py

4. Gradio will generate a local link and a shareable link to access the app in a browser.

5. Folder Structure:

project/

```
├── app.py              # Main application code
├── requirements.txt    # List of dependencies
```

This is a simple project with only one main script file. In larger projects, the folder can be extended into modules for analysis, UI, and data handling.

6. Running the Application:

1. Start the application with python app.py.

2. Open the Gradio interface in a browser.

3. Use the City Analysis tab to enter a city name.

The model will generate crime index, accident data, and safety evaluation.

4. Use the Citizen Services tab to ask queries.

The assistant will provide detailed answers about policies or civic issues.

5. Both outputs are generated in real-time and displayed as text.

7. API Documentation:

Even though this project is mainly a Gradio app, internally it uses functions as APIs:

city_analysis(city_name)

Input: City name (string).

Output: Text analysis with crime index, accident rates, and safety assessment.

citizen_interaction(query)

Input: User query (string).

Output: AI-generated response about public services or policies.

8. User Interface:

The UI is designed to be minimal and functional:

- Navigation Tabs: Separate tabs for City Analysis and Citizen Services.
- Textboxes: Input boxes for user queries or city names.
- Output Area: Large textboxes that display AI responses.
- Dark Theme: Easy on the eyes and professional in appearance.

9. Testing:

Testing was carried out at multiple levels:

Unit Testing: Verified AI response generation functions.

Interface Testing: Checked Gradio buttons, textboxes, and outputs.

Manual Testing: Entered multiple cities (e.g., Mumbai, London) to verify consistency.

10. Known Issues:

- The analysis depends on AI-generated responses; no real-time crime/traffic database is used.
- Requires internet connection to load the IBM Granite model.

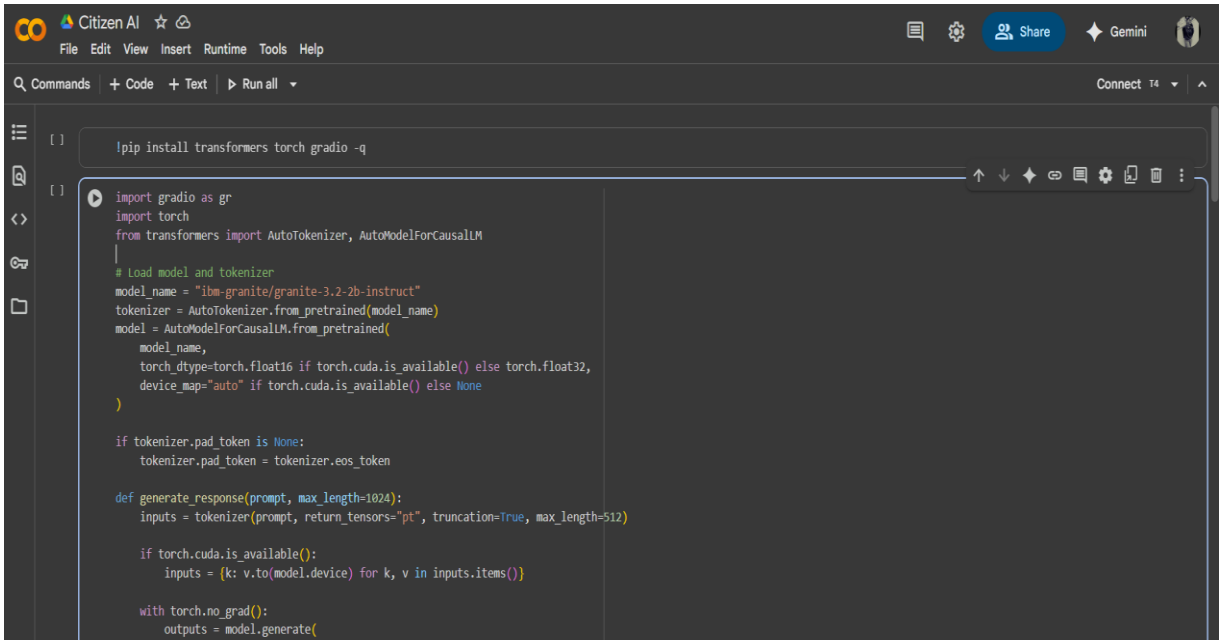
- Responses may vary slightly each time due to AI's generative nature.

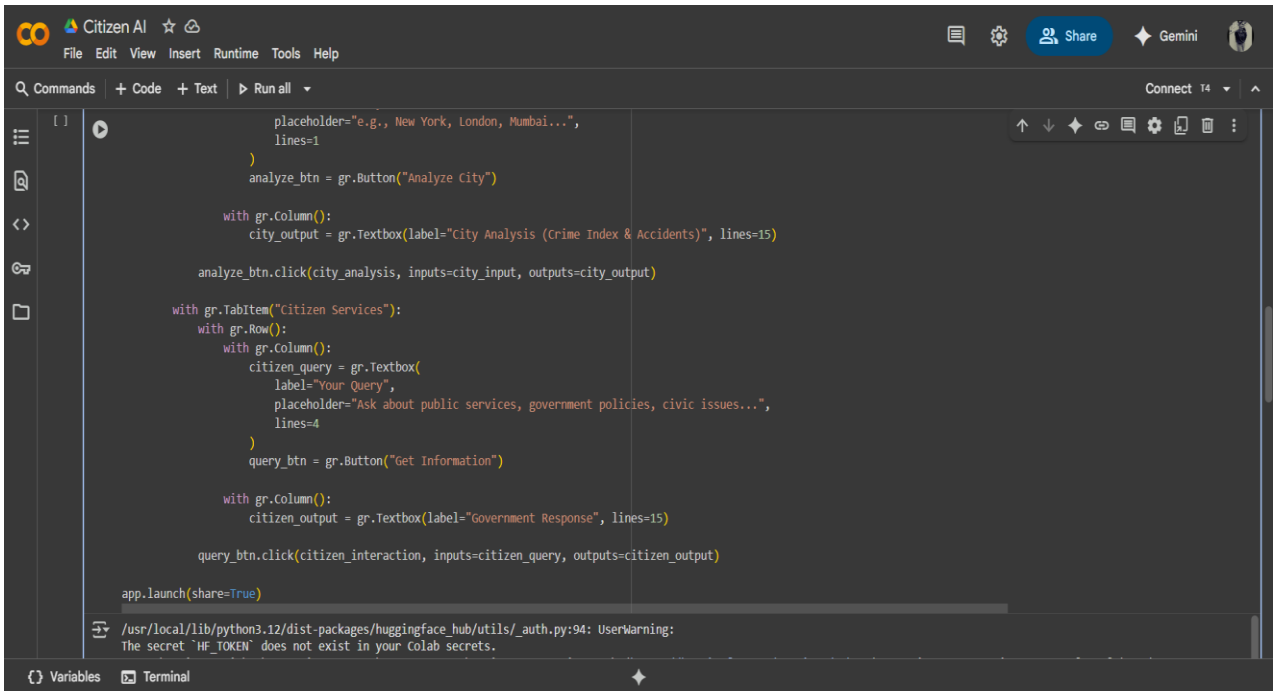
13. Future Enhancements:

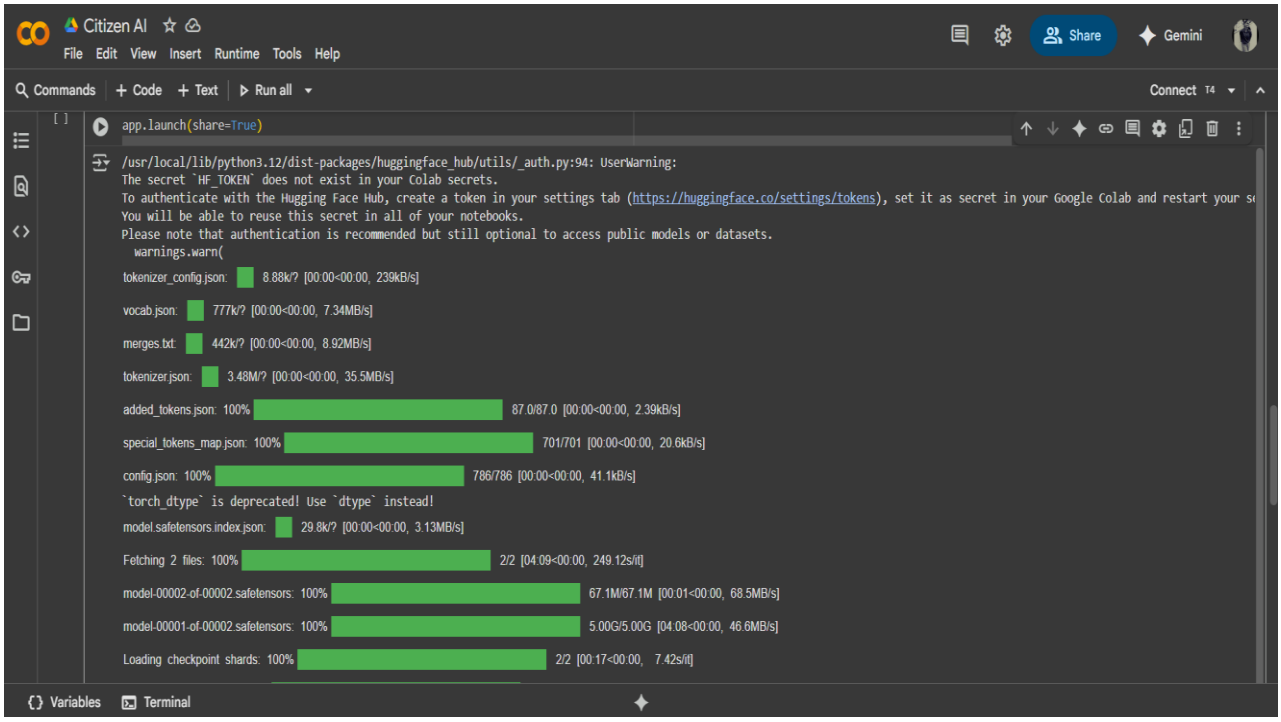
- Connect to real-time data APIs (crime reports, traffic statistics).
- Add data visualization (graphs, maps, charts).
- Support multiple languages for wider accessibility.
- Implement query history and analytics for government use.
- Deploy on cloud platforms (AWS, IBM Cloud, Azure) for larger-scale access.

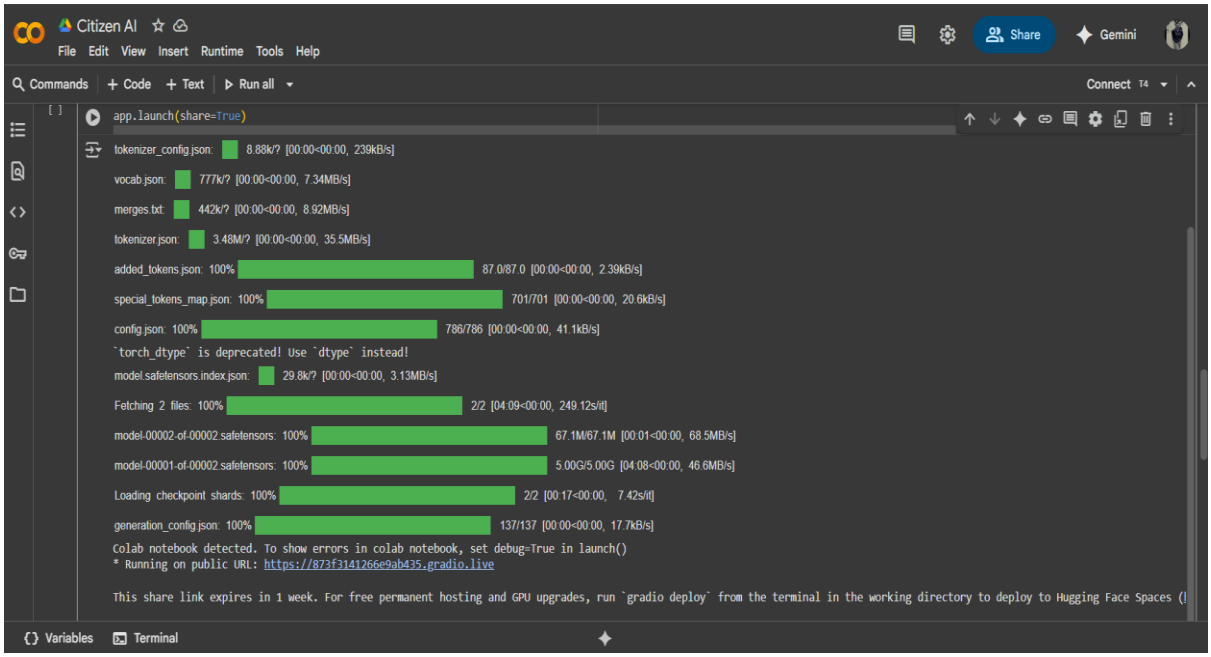
12.Screenshots:

Program:


Citizen AI
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect
!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(

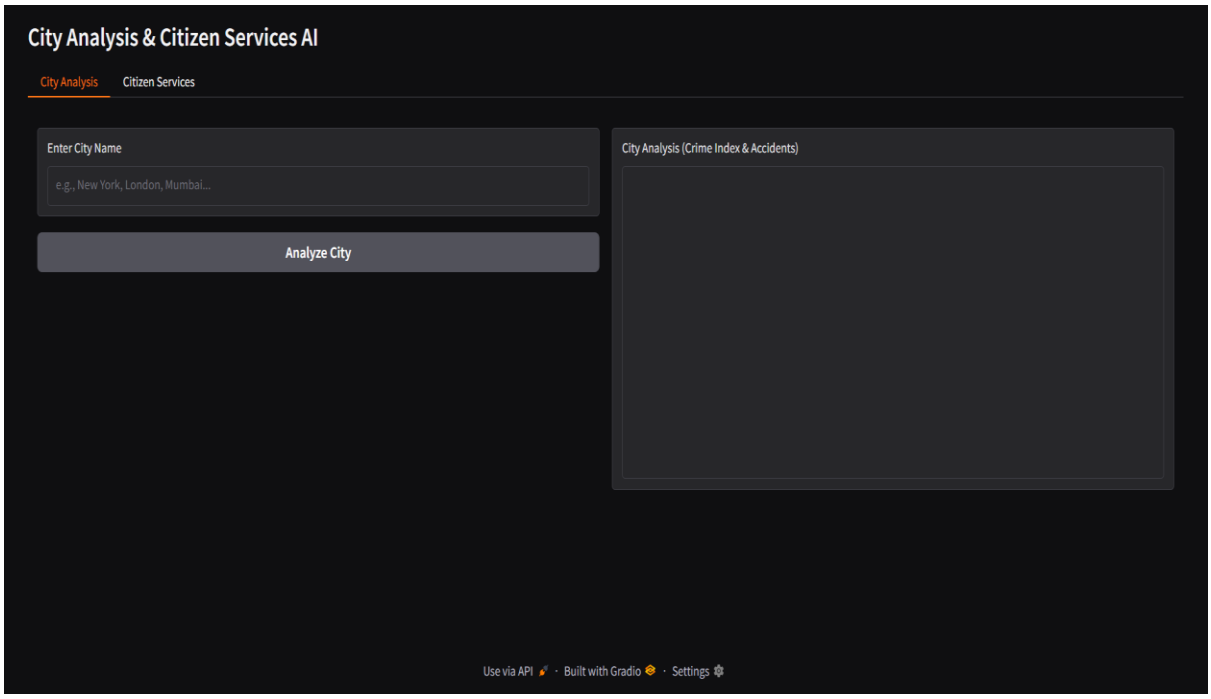

Citizen AI
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect
placeholder="e.g., New York, London, Mumbai...",
lines=1
)
analyze_btn = gr.Button("Analyze City")
with gr.Column():
city_output = gr.Textbox(label="City Analysis (Crime Index & Accidents)", lines=15)
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)
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
Variables
Terminal


Citizen AI
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect
app.launch(share=True)
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 239kB/s]
vocab.json: 777k/? [00:00<00:00, 7.34MB/s]
merges.txt: 442k/? [00:00<00:00, 8.92MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 35.5MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 2.39kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 20.6kB/s]
config.json: 100% 786/786 [00:00<00:00, 41.1kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 3.13MB/s]
Fetching 2 files: 100% 2/2 [04:09<00:00, 249.12s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 68.5MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [04:08<00:00, 46.6MB/s]
Loading checkpoint shards: 100% 2/2 [00:17<00:00, 7.42s/it]
Variables
Terminal

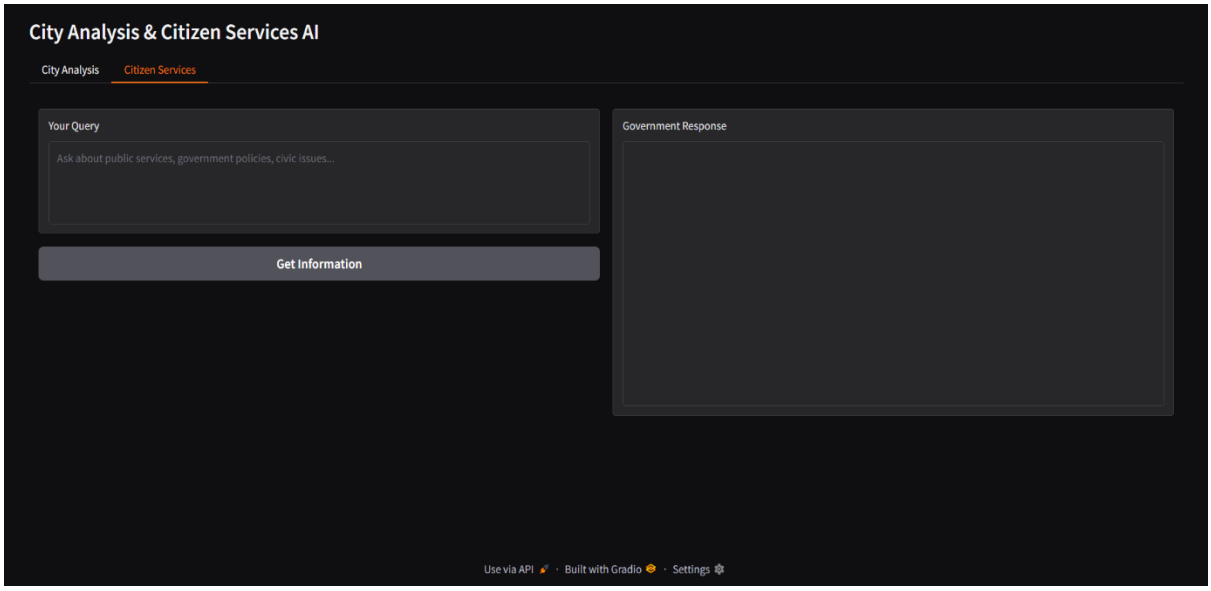

Citizen AI
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect
app.launch(share=True)
tokenizer_config.json: 8.88k/? [00:00<00:00, 239kB/s]
vocab.json: 777k/? [00:00<00:00, 7.34MB/s]
merges.txt: 442k/? [00:00<00:00, 8.92MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 35.5MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 2.39kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 20.6kB/s]
config.json: 100% 786/786 [00:00<00:00, 41.1kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 3.13MB/s]
Fetching 2 files: 100% 2/2 [04:09<00:00, 249.12s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 68.5MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [04:08<00:00, 46.6MB/s]
Loading checkpoint shards: 100% 2/2 [00:17<00:00, 7.42s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 17.7kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://873f3141266e9ab435.gradio.live
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (
Variables
Terminal

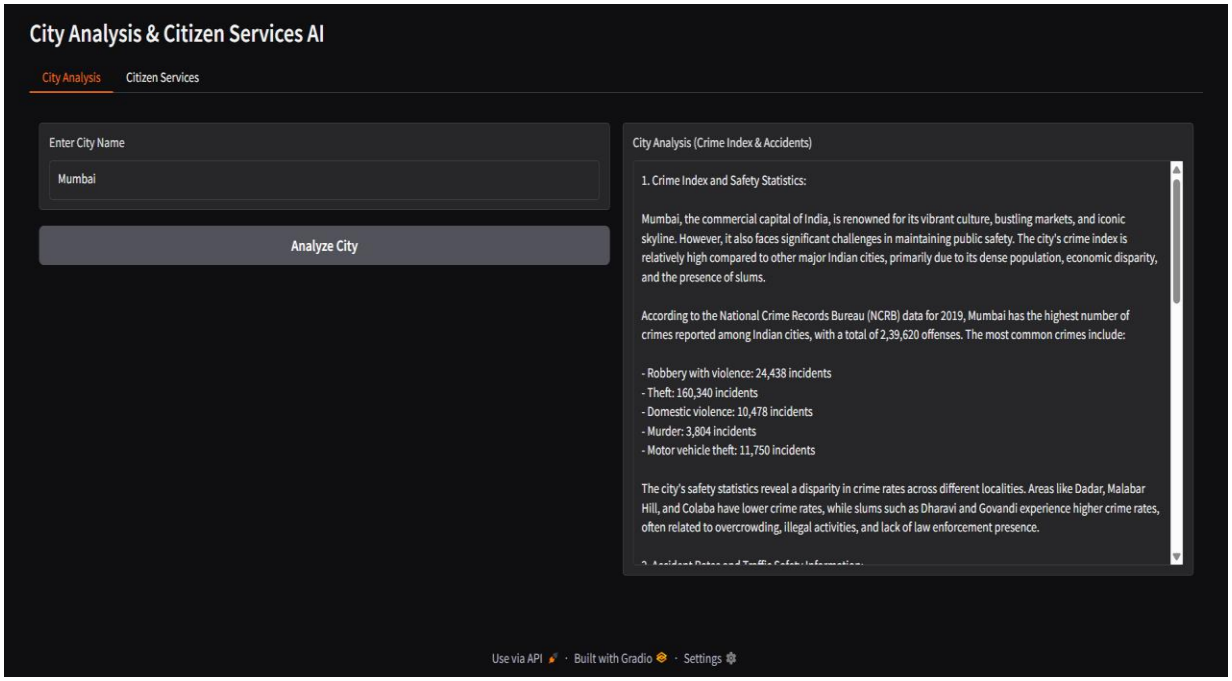
Output:

First view of the screen,



Second view of the output screen, Citizen Services.




City Analysis & Citizen Services AI
City Analysis
Citizen Services
Enter City Name
Mumbai
Analyze City
City Analysis (Crime Index & Accidents)
1. Crime Index and Safety Statistics:
Mumbai, the commercial capital of India, is renowned for its vibrant culture, bustling markets, and iconic skyline. However, it also faces significant challenges in maintaining public safety. The city's crime index is relatively high compared to other major Indian cities, primarily due to its dense population, economic disparity, and the presence of slums.
According to the National Crime Records Bureau (NCRB) data for 2019, Mumbai has the highest number of crimes reported among Indian cities, with a total of 2,39,620 offenses. The most common crimes include:
- Robbery with violence: 24,438 incidents
- Theft: 160,340 incidents
- Domestic violence: 10,478 incidents
- Murder: 3,804 incidents
- Motor vehicle theft: 11,750 incidents
The city's safety statistics reveal a disparity in crime rates across different localities. Areas like Dadar, Malabar Hill, and Colaba have lower crime rates, while slums such as Dharavi and Govandi experience higher crime rates, often related to overcrowding, illegal activities, and lack of law enforcement presence.
Use via API · Built with Gradio · Settings