Citizen AI – Intelligent Citizen Engagement Plateform

1.Introduction

Project Title: Citizen AI – Intelligent Citizen Engagement Plateform

Team Member(s): E.Meenakshi

Team Member(s): R.Monika

Team Member(s): B.Pavithra

Team Member(s): S.NithyaShree

Technologies Used: Python, Gradio, Hugging Face Transformers, PyTorch, Google Colab

Description

This project demonstrates the integration of an AI-powered language model into a web-based application. It helps in analyzing city safety statistics (crime index, accident rates) and provides AI-generated responses to citizen queries related to government policies and services.

2. Project Overview

Purpose:

The purpose of this application is to show how AI can be used in civic applications to provide quick, meaningful, and structured information.

Core Functionalities:

1.City Analysis– Enter a city name and the AI generates an analysis including crime index, accident statistics, and overall safety assessment.

2. Citizen Services – Works as a virtual government assistant that provides answers to queries about public services, civic issues, or policies.

Advantages:

- ❖ Saves time by quickly generating reports and responses.
- ❖ Simple two-tab interface.
- ❖ Can be extended with real-time data sources**.**

3. Architecture

Frontend (Gradio):

using Gradio's Blocks API.

Provides an interactive web-based interface with two tabs (City Analysis and Citizen Services).

Backend (Google Colab + Python):

application runs in Google Colab, which provides GPU acceleration.

Python handles model inference and text processing.

LLM Integration (IBM Granite):

Uses Hugging Face-hosted IBM Granite model.

Responsible for natural language understanding and text generation.

Deployment:

The app is launched using app.launch(share=True).

Generates a shareable Gradio link accessible from any device.

4. Setup Instructions

Prerequisites:

Google account for Colab.

Hugging Face account with access token.

Stable internet connection.

Steps to Run:

- Open Google Colab.
- Change runtime type to T4 GPU.
- Install dependencies:
  - !pip install transformers torch gradio -q
- Log in to Hugging Face:

  ```
  from huggingface_hub import login
  login("YOUR_HF_TOKEN")
  ```
- Copy and paste the project code into a Colab notebook.
- Run all cells sequentially.
- Launch the app → Colab will display a public Gradio link.

5. Folder Structure

Since this project is notebook-based, the structure is minimal:

project/

└── city_analysis_ai.ipynb

6. Running the Application

Run the Colab notebook.

Launch the app using app.launch(share=True).

Two tabs will be available:

- **City Analysis Tab:**Input city → get AI-generated safety analysis.
- **Citizen Services Tab:** Input query → get AI-generated government-style response.

7. API Documentation

Conceptual – currently inside notebook but extendable to APIs

POST /city-analysis

Input: City name.

Output: Report on crime, accidents, and safety.

POST /citizen-query

Input: Public service/civic query.

Output: Government-style AI-generated response.

8. User Interface

- **Tabs:** City Analysis and Citizen Services.
- **Textboxes:** Input fields for city name and queries, output fields for responses.
- **Buttons:** “Analyze City” and “Get Information”.
- **Shareable Gradio link:** Makes the app accessible on any browser or device.

9. Testing

Functional Testing:

Verified that City Analysis produces contextual outputs.

Checked Citizen Services with queries about healthcare, education, and policies.

Interface Testing:

Ensured both tabs and buttons work correctly.

Cross-Device Testing:

Confirmed app works smoothly on mobile and desktop browsers.

10. Known Issues

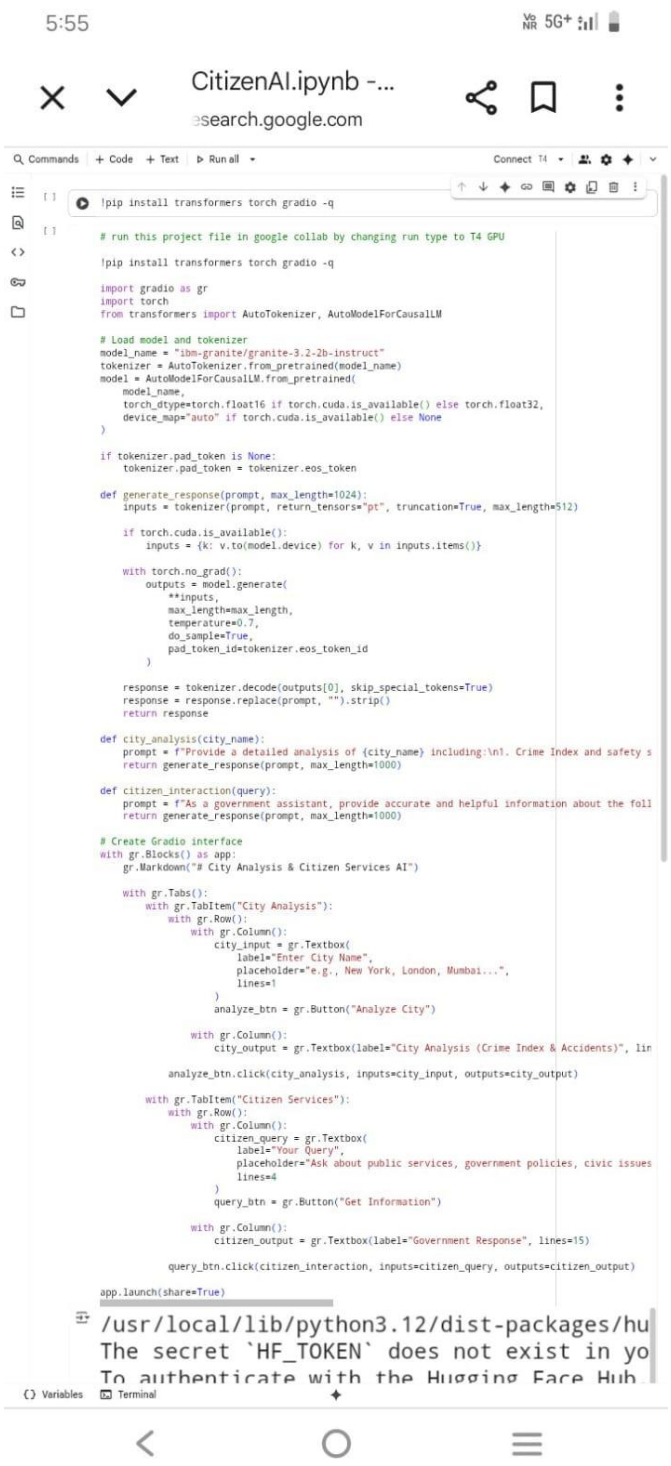
- ❖ Slow if GPU resources are limited.
- ❖ Responses may not reflect real statistics (AI-generated, not connected to live databases).
- ❖ Requires Hugging Face token.

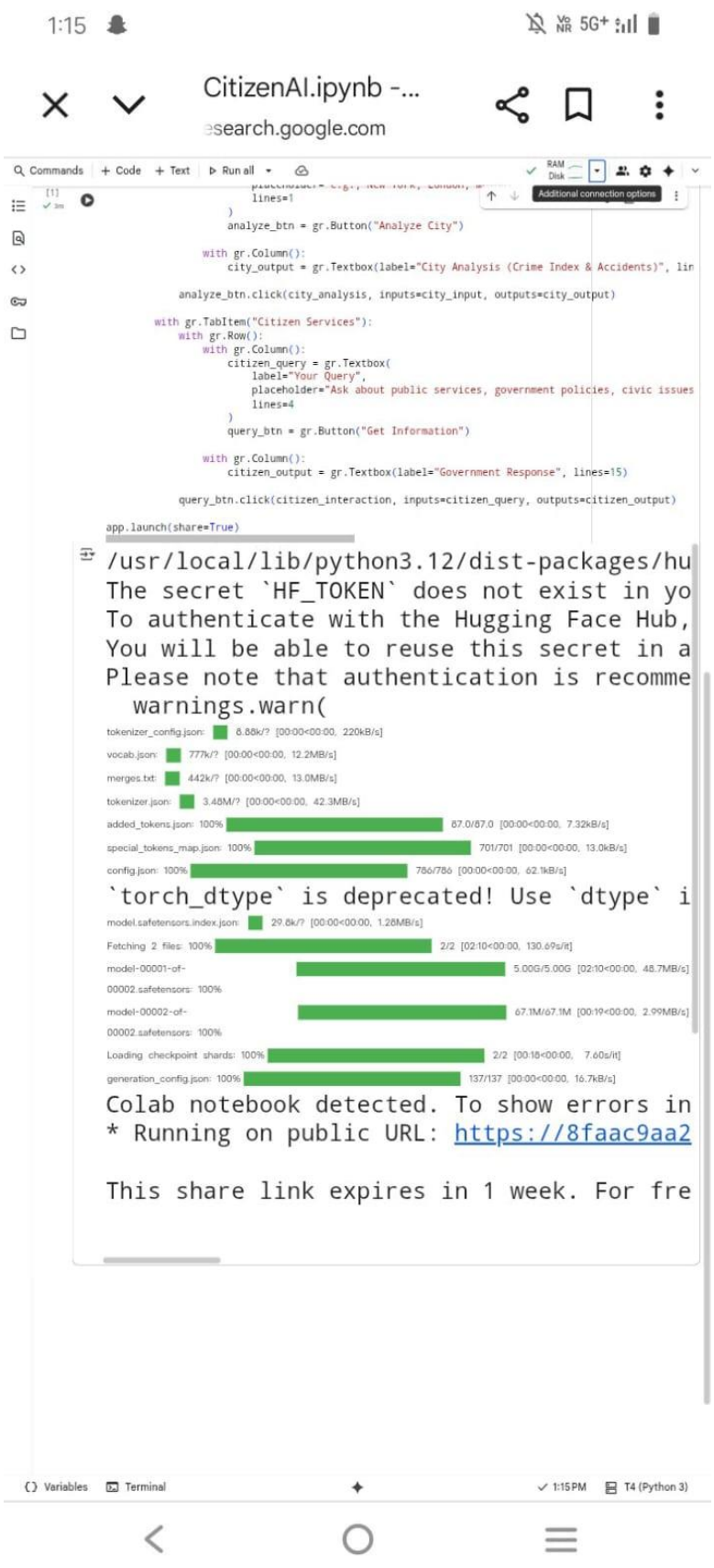
- ❖ Limited to text output (no visual charts yet).
- ❖ Internet connection required to run in Colab.

11. Future Enhancements

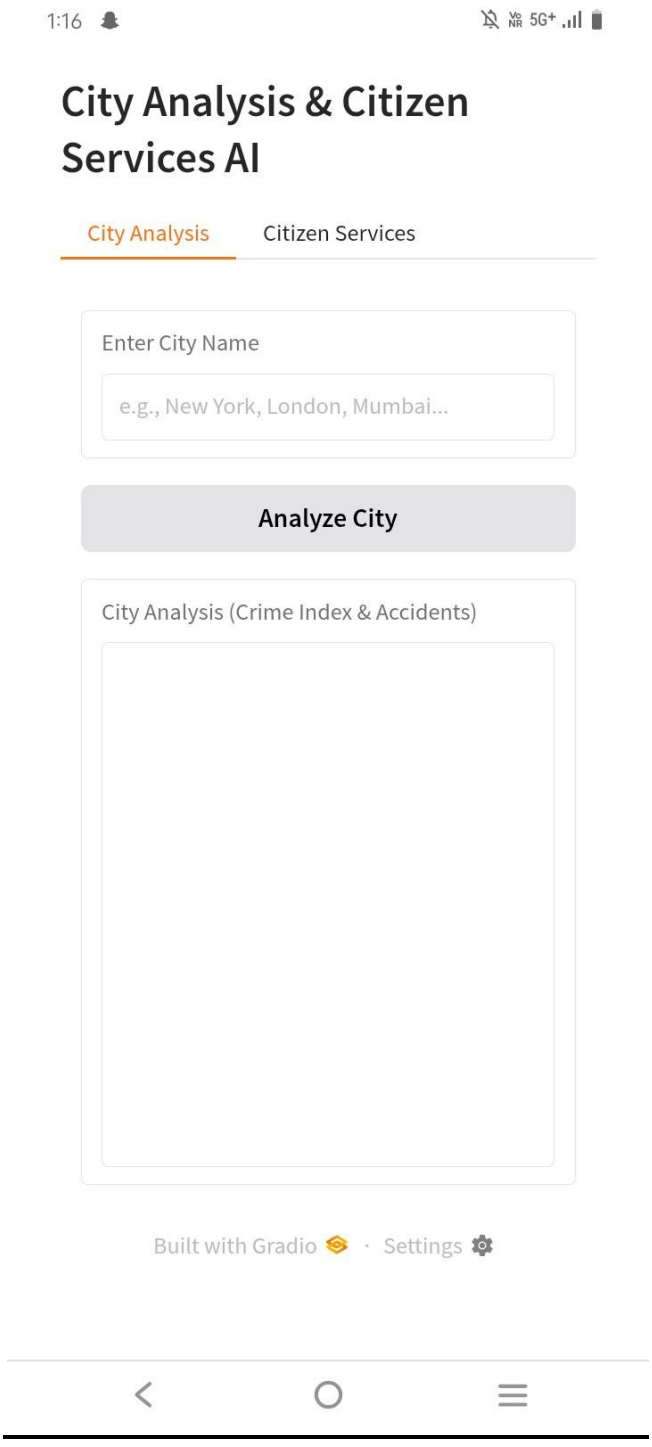
- ❖ Connect to real datasets (crime/accident reports, government databases).
- ❖ Add user authentication for secure access.
- ❖ Provide multi-language support.
- ❖ Enable download option for generated reports.
- ❖ Deploy as a standalone web app or mobile app.
- ❖ Improve UI with charts, maps, and graphs.
- ❖ Add real-time collaboration features

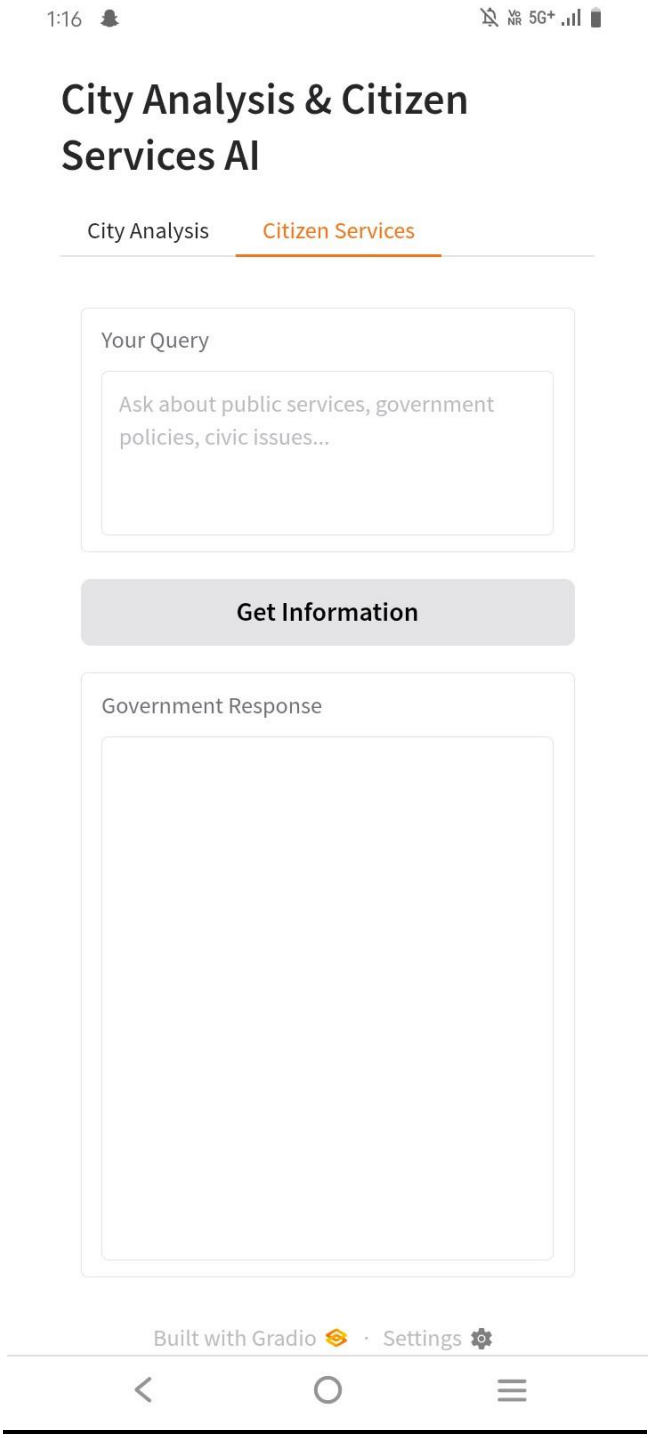
Coding


5:55
CitizenAI.ipynb -...
search.google.com
Commands
+ Code
+ Text
Run all
Connect
!pip install transformers torch gradio -q
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 safety s
return generate_response(prompt, max_length=1000)
def citizen_interaction(query):
prompt = f"As a government assistant, provide accurate and helpful information about the foll
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)", lin
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/hu
The secret `HF_TOKEN` does not exist in yo
To authenticate with the Hugging Face Hub
Variables
Terminal


1:15
CitizenAI.ipynb -...
esearch.google.com
Commands
+ Code
+ Text
Run all
RAM
Disk
Additional connection options
lines=1
analyze_btn = gr.Button("Analyze City")
city_output = gr.Textbox(label="City Analysis (Crime Index & Accidents)", lin
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/hu
The secret `HF_TOKEN` does not exist in yo
To authenticate with the Hugging Face Hub,
You will be able to reuse this secret in a
Please note that authentication is recomme
warnings.warn(
tokenizer_config.json: 6.88k/? [00:00<00:00, 220kB/s]
vocab.json: 777k/? [00:00<00:00, 12.2MB/s]
merges.txt: 442k/? [00:00<00:00, 13.0MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 42.3MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 7.32kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 13.0kB/s]
config.json: 100% 786/786 [00:00<00:00, 62.1kB/s]
`torch_dtype` is deprecated! Use `dtype` i
model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.28MB/s]
Fetching 2 files: 100% 2/2 [02:10<00:00, 130.69s/it]
model-00001-of-
00002.safetensors: 100%
5.00G/5.00G [02:10<00:00, 48.7MB/s]
model-00002-of-
00002.safetensors: 100%
67.1M/67.1M [00:19<00:00, 2.99MB/s]
Loading checkpoint shards: 100% 2/2 [00:16<00:00, 7.60s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 16.7kB/s]
Colab notebook detected. To show errors in
* Running on public URL: https://8faac9aa2
This share link expires in 1 week. For fre
Variables
Terminal
1:15 PM
T4 (Python 3)

Output


1:16
City Analysis & Citizen Services AI
City Analysis
Citizen Services
Enter City Name
e.g., New York, London, Mumbai...
Analyze City
City Analysis (Crime Index & Accidents)
Built with Gradio
Settings


1:16
5G+
City Analysis & Citizen Services AI
City Analysis
Citizen Services
Your Query
Ask about public services, government policies, civic issues...
Get Information
Government Response
Built with Gradio
Settings