1. INTRODUCTION:

PROJECT TITLE: CITIZEN AI: INTELLIGENT CITIZEN ENGAGEMENT PLATFORM

TEAM MEMBER:POOJA B
TEAM MEMBERS: MADHUMITHA S
TEAM MEMBERS:SHAKILA G
TEAM MEMBERS :KAVIYA S

2.PROJECT OVERVIEW:

Based on multiple recent and ongoing projects, "Citizen AI" refers to the development of artificial intelligence systems that are human-centered and designed to enhance public services. Rather than viewing citizens as passive data sources, the approach prioritizes transparency, ethics, and engagement to build trustworthy AI applications for government and other sectors.

"Citizen AI" also describes the process of empowering "citizen developers"—non-technical users—to create their own AI solutions within secure frameworks.

Key Features

Automation:

AI automates routine administrative tasks, reducing human workload and speeding up service delivery by handling queries and processing applications more efficiently.

Personalization:

AI systems can analyze citizen data to provide tailored information, relevant service recommendations, and personalized experiences, increasing user satisfaction.

Data-Driven Insights:

AI analyzes vast datasets to identify trends, patterns, and citizen needs, enabling governments to make more informed, evidence-based decisions and allocate resources effectively.

Natural Language Processing (NLP):

NLP powers AI-driven tools like chatbots and virtual assistants to understand and process human language, facilitating natural and intuitive interactions between citizens and public services.

3.ARCHITECTURE: The architecture of a Citizen AI is a socio-technical framework designed to empower citizens and improve public services by integrating AI with citizen-centric principles. The structure goes beyond mere technological components, incorporating ethical guidelines, data governance, and user-centric design to ensure AI systems are transparent, fair, and accountable.

Core architectural layers and components

A robust Citizen AI architecture is composed of interconnected layers that facilitate data processing, intelligent analysis, and secure interaction with citizens.

1. Data-merging layer

This layer handles the acquisition and consolidation of data from various sources to build a comprehensive view of citizen-related information.

- ☐ **Data sources:** Gathers structured, semi-structured, and unstructured data from:
  - **Governmental sources:** Departmental databases, data warehouses, and public records.
  - **External sources:** IoT sensors, satellite imagery, social media, and websites.
- ☐ **Data governance:** Implements strict data policies to ensure citizen consent, privacy, and security in compliance with regulations like GDPR.
- ☐ **Data processing and storage:** Uses scalable cloud platforms to store and process large volumes of data, often employing a modular, microservices-based approach for flexibility.
- ☐ 4.SET OF INSTRUCTIONS: 1.GRADIO framework knowledge:GRADIO documentation 2.IBM granite models(hugging face):IBM granite models 3.python programming proficiency:python documentation 4.version control with Git:Git documentation 5.Google collab's T4 GPU Knowledge:Google Collab.

- ☐ 5.RUNNING THE APPLICATION:After the AI model is trained and validated, it must be deployed and integrated into the production environment.
- ☐ 6.USERFACE:The Citizen AI's UI must be designed for maximum accessibility and usability, as it will be used by a wide range of people with varying levels of technical proficiency.
- ☐ 7.TESTING:Testing an AI system goes beyond traditional software testing and must account for the probabilistic and data-dependent nature of AI.

Key aspects of AI in citizen services include enhanced personalization, automation of tasks like responding to queries, improved data analysis for better decision-making, increased accessibility through 24/7 virtual assistants, and more efficient, data-driven public safety applications. These applications aim to make government services more efficient, responsive, and tailored to individual needs, ultimately improving citizen satisfaction and engagement.

Here are the key aspects in detail:

•Personalized Services:

AI can analyze citizen data to provide tailored recommendations, information, and services, enhancing user experience and satisfaction.

•Chatbots & Virtual Assistants:

These AI tools offer instant, 24/7 support, answering questions, guiding citizens through processes, and handling routine tasks to reduce wait times and improve accessibility.

Efficiency and Automation

•Task Automation:

AI automates repetitive tasks such as data entry, document processing, and handling common citizen inquiries, freeing up public servants for more strategic responsibilities.

•Cost Reduction:

By automating tasks and optimizing resource allocation, AI can significantly lower operational costs for government agencies.

Informed Decision-Making

Data Analysis: AI systems process vast amounts of data to identify trends, predict future needs, and provide data-driven insights for more informed policy development and service delivery.

Improved Public Safety

Predictive Analytics: AI can identify crime hotspots and predict future demand for services, allowing for more effective resource allocation and faster emergency response.

Increased Citizen Engagement & Trust

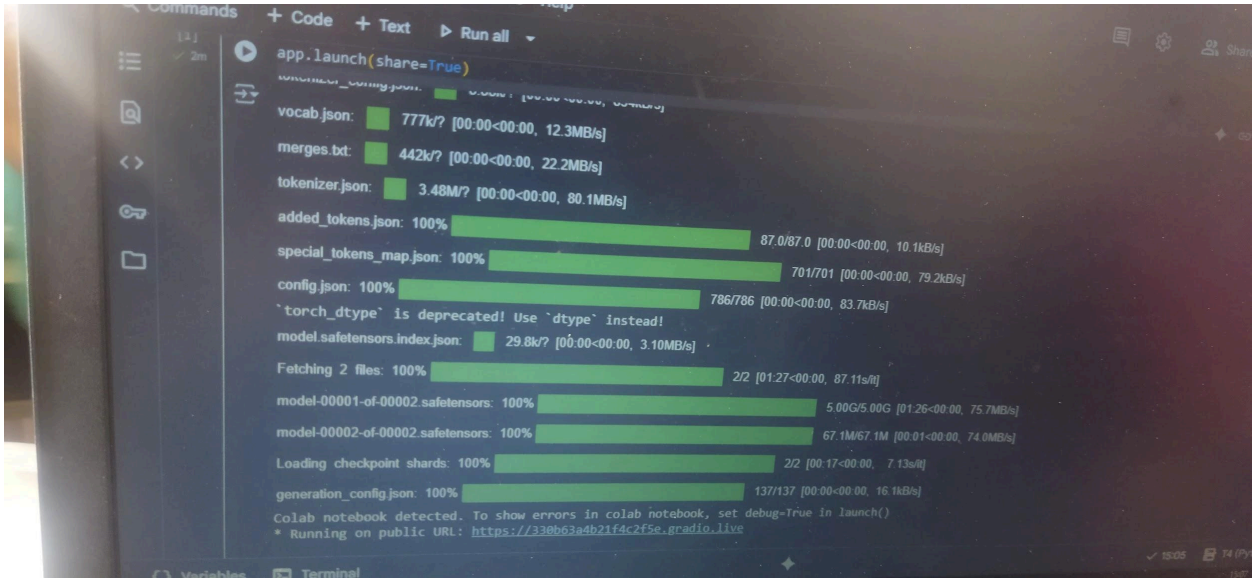
Accessible Platforms:

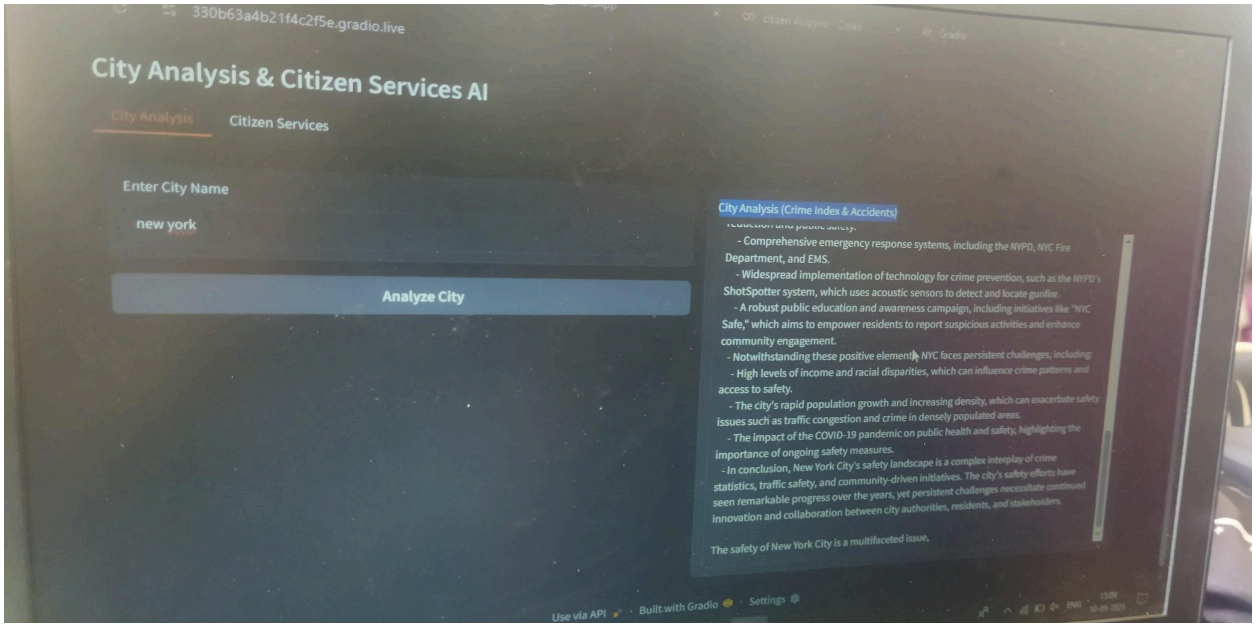
AI-powered interfaces and tools make government services more user-friendly and accessible to a wider range of citizens.

Building Trust:

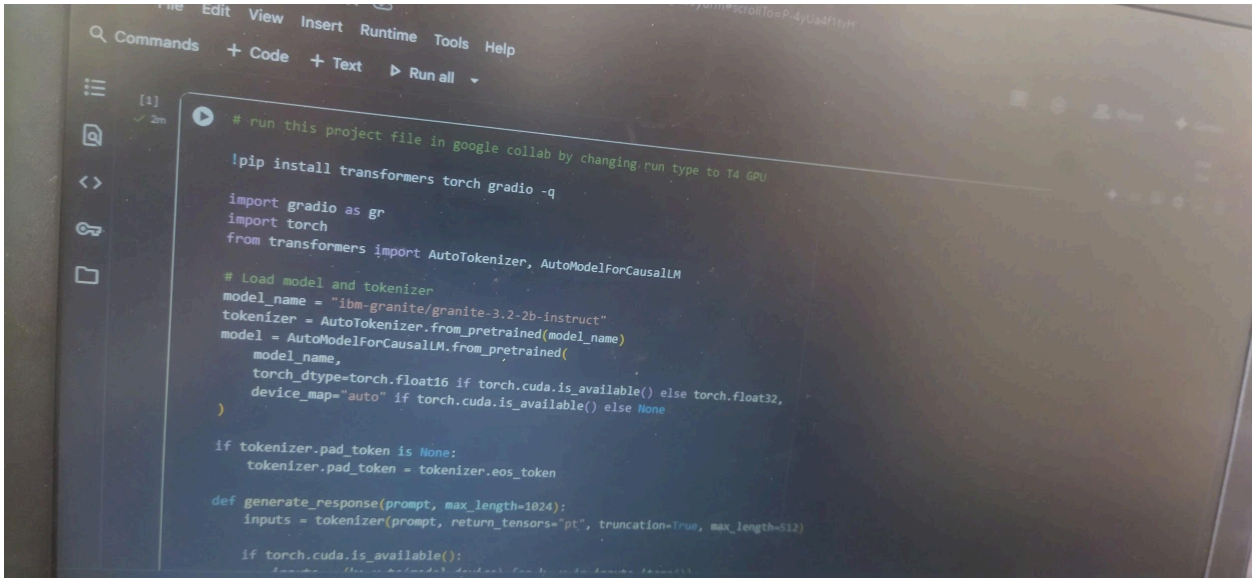
- By providing personalized assistance, handling feedback quickly, and increasing transparency, AI can foster greater trust between citizens and government.

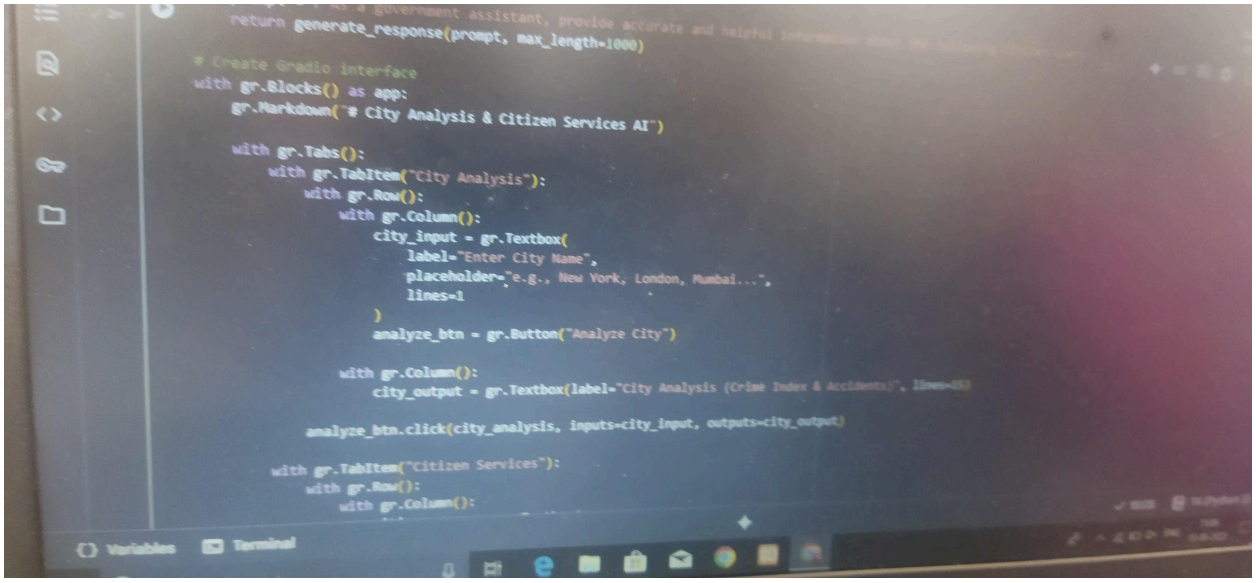
9.Screenshot:


+ Code
+ Text
Run all
app.launch(share=True)
vocab.json: 777k/? [00:00<00:00, 12.3MB/s]
merges.txt: 442k/? [00:00<00:00, 22.2MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 80.1MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 10.1kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 79.2kB/s]
config.json: 100% 786/786 [00:00<00:00, 83.7kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 3.10MB/s]
Fetching 2 files: 100% 2/2 [01:27<00:00, 87.11s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:26<00:00, 75.7MB/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 74.0MB/s]
Loading checkpoint shards: 100% 2/2 [00:17<00:00, 7.13s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 16.1kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://330b63a4b21f4c2f5e.gradio.live
Variables
Terminal

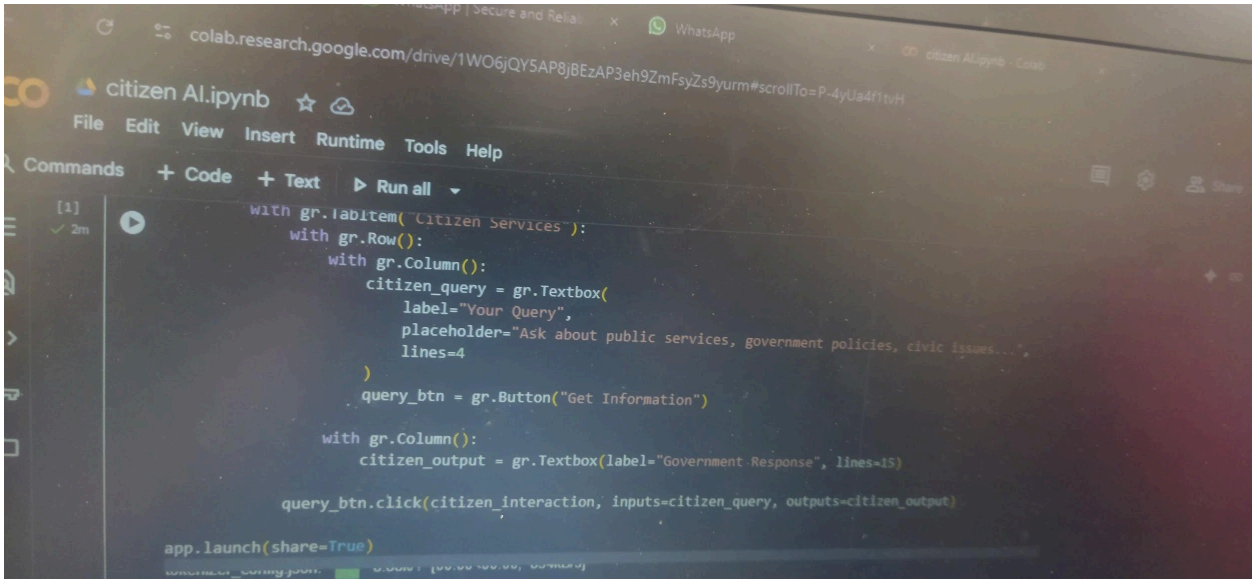

330b63a4b21f4c2f5e.gradio.live
City Analysis & Citizen Services AI
City Analysis
Citizen Services
Enter City Name
new york
Analyze City
City Analysis (Crime Index & Accidents)
- Comprehensive emergency response systems, including the NYPD, NYC Fire Department, and EMS.
- Widespread implementation of technology for crime prevention, such as the NYPD's ShotSpotter system, which uses acoustic sensors to detect and locate gunfire.
- A robust public education and awareness campaign, including initiatives like "NYC Safe," which aims to empower residents to report suspicious activities and enhance community engagement.
- Notwithstanding these positive elements, NYC faces persistent challenges, including:
- High levels of income and racial disparities, which can influence crime patterns and access to safety.
- The city's rapid population growth and increasing density, which can exacerbate safety issues such as traffic congestion and crime in densely populated areas.
- The impact of the COVID-19 pandemic on public health and safety, highlighting the importance of ongoing safety measures.
- In conclusion, New York City's safety landscape is a complex interplay of crime statistics, traffic safety, and community-driven initiatives. The city's safety efforts have seen remarkable progress over the years, yet persistent challenges necessitate continued innovation and collaboration between city authorities, residents, and stakeholders.
The safety of New York City is a multifaceted issue,
Use via API
Built with Gradio
Settings

10.CODING:


File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
[1]
2m
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():



```
        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():
```



```
colab.research.google.com/drive/1WO6jQY5AP8jBEzAP3eh9ZmFsyZs9yurm#scrollTo=P-4yUa4f1tvH

citizen AI.ipynb
File  Edit  View  Insert  Runtime  Tools  Help
Commands  + Code  + Text  Run all

[1]  2m

        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)
```