CITIZEN AI:INTELLIGENT CITIZEN ENAGEGEMENT PLATFORM

Project Documentation

1. Introduction :

Project title : Citizen Ai:Intelligent Citizen Enagagement Platform.

- Team leader : SHADIK BASHA.H
- Team member :KRISHNA KUMAR.R
- Team member : DINESH.G
- Team member :JEROME.J

2. Project Overview:

- **Purpose:** The goal of the Citizen AI Project is to empower city residents by leveraging AI and real-time data to create a more eco-conscious and connected urban environment. It helps optimize resources like energy, water, and waste, and provides personalized eco-tips to encourage sustainable behaviors among citizens. For city officials, the project serves as a decision-making tool by providing insights, forecasting capabilities, and summaries of complex policies. The project aims to connect technology, governance, and community to build more efficient, resilient, and greener cities.
- **Features:**
    - **Conversational Interface:** This allows for natural language interaction, enabling citizens and officials to ask questions and receive guidance.
    - **Policy Summarization:** Converts long government documents into clear, actionable summaries for easier understanding.
    - **Resource Forecasting:** Uses historical and real-time data to predict future usage of energy, water, and waste.
    - **Eco-Tip Generator:** Recommends daily actions to help users reduce their environmental impact based on their behavior.
    - **Citizen Feedback Loop:** Gathers and analyzes public input to assist with city planning and service enhancements.
    - **KPI Forecasting:** Projects key performance indicators to help officials monitor progress and plan strategically.
    - **Anomaly Detection:** Acts as an early warning system by identifying unusual patterns in sensor or usage data to flag potential issues.

  - **Multimodal Input Support:** Can handle different data types, including text, PDFs, and CSVs, for analysis and forecasting.
  - **User-friendly Interface:** An intuitive dashboard built with Streamlit or Gradio UI that allows both citizens and city officials to easily interact with the assistant.

3. Architecture:

- **Frontend (Streamlit):** The frontend is an interactive web UI with multiple pages for dashboards, file uploads, a chat interface, feedback forms, and report viewers. It uses the Streamlit-option-menu library for sidebar navigation, and each page is modularized for scalability.
- **Backend (FastAPI):** This serves as the REST framework for API endpoints that handle document processing, chat, eco-tip generation, and more. It is optimized for asynchronous performance and easy Swagger integration.
- **LLM Integration (IBM Watsonx Granite):** The project uses Granite LLM models from IBM Watsonx for natural language understanding and generation. Prompts are specifically designed to produce summaries, reports, and sustainability tips.
- **Vector Search (Pinecone):** Uploaded policy documents are converted into embeddings using Sentence Transformers and stored in Pinecone. Semantic search is enabled via cosine similarity, letting users search documents using natural language queries.
- **ML Modules (Forecasting and Anomaly Detection):** Lightweight ML models from Scikit-learn are used for forecasting and anomaly detection. Time-series data is parsed, modeled, and visualized using pandas and matplotlib.

4. Setup Instructions:

- **Prerequisites:**
  - Python 3.9 or later
  - pip and virtual environment tools
  - API keys for IBM Watsonx and Pinecone
  - Internet access for cloud services
- **Installation Process:**
  - Clone the repository.
  - Install dependencies from

    requirements.txt.

  - Create and configure a

    .env file with credentials.

    - Run the backend server using FastAPI.
    - Launch the frontend via Streamlit.
    - Upload data and interact with the modules.

5. Folder Structure:

- app/ - Contains all FastAPI backend logic, including routers, models, and integration modules.
- app/api/ - Subdirectory for modular API routes like chat, feedback, and document vectorization.
- ui/ - Contains frontend components for Streamlit pages and form UIs.
- smart_dashboard.py - The entry script for the main Streamlit dashboard.
- granite_llm.py - Handles all communication with the IBM Watsonx Granite model.
- document_embedder.py - Converts documents to embeddings and stores them in Pinecone.
- kpi_file_forecaster.py - Forecasts future trends for energy/water using regression.
- anomaly_file_checker.py - Flags unusual values in uploaded KPI data.
- report_generator.py - Constructs AI-generated sustainability reports.

6. Running the Application:

- To start the project, launch the FastAPI server and then run the Streamlit dashboard.
- Navigate through the pages using the sidebar.
- Users can upload documents or CSVs, interact with the chat assistant, and view outputs like reports, summaries, and predictions.
- All interactions are real-time, with the frontend dynamically updating via backend APIs.

7. API Documentation:

- The backend APIs include:
    - POST /chat/ask - Accepts a user query and returns an AI-generated message.
    - POST /upload-doc - Uploads and embeds documents in Pinecone.
    - GET /search-docs - Returns semantically similar policies to a user query.
    - GET /get-eco-tips - Provides sustainability tips on selected topics.
    - POST /submit-feedback - Stores citizen feedback.

- Each endpoint is documented and tested in Swagger UI.

8. Authentication:

- For demonstration purposes, this version of the project runs in an open environment.
- Secure deployments can include:
  - Token-based authentication (JWT or API keys).
  - OAuth2 with IBM Cloud credentials.
  - Role-based access for different user types (admin, citizen, researcher).
- Future enhancements will include user sessions and history tracking.

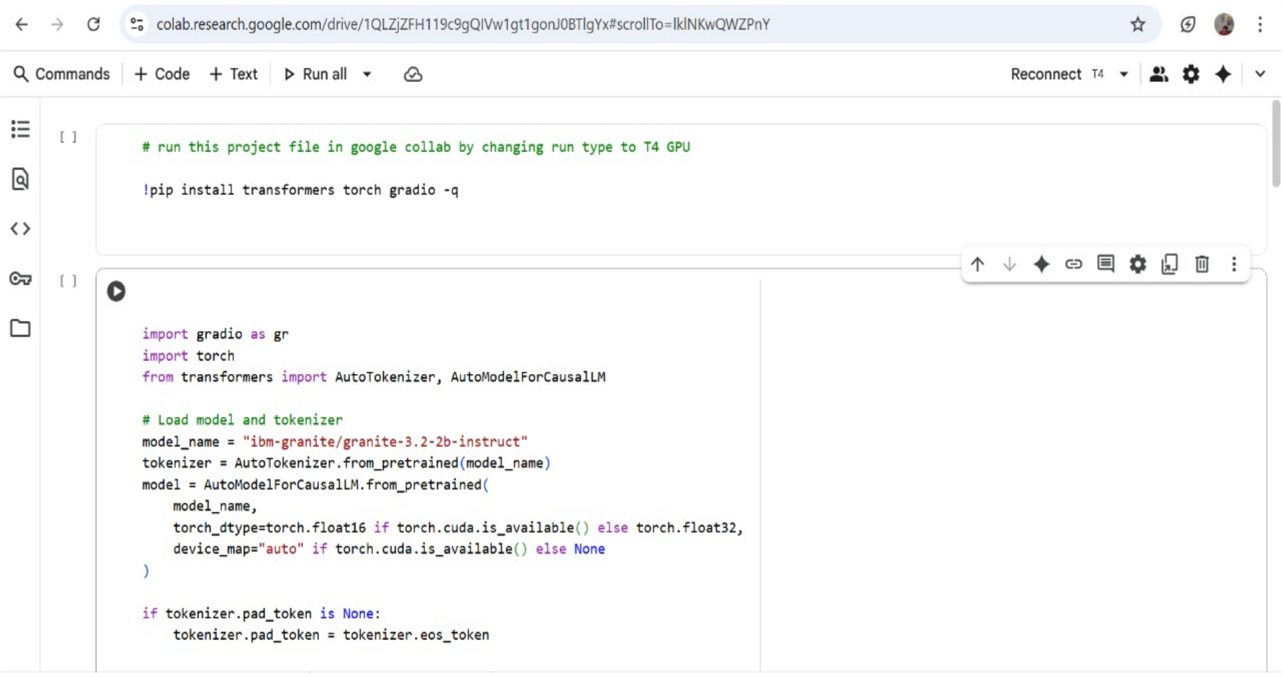
9. User Interface:

- The interface is minimalist and designed for accessibility for non-technical users.
- Key elements include:
  - A sidebar for navigation.
  - KPI visualizations with summary cards.
  - Tabbed layouts for chat, eco tips, and forecasting.
  - Real-time form handling.
  - PDF report download capability.

10. Testing:

- Testing was conducted in several phases:
  - **Unit Testing:** For prompt engineering functions and utility scripts.
  - **API Testing:** Done via Swagger UI, Postman, and test scripts.
  - **Manual Testing:** To validate file uploads, chat responses, and output consistency.
  - **Edge Case Handling:** To address malformed inputs, large files, and invalid API keys.
- Each function was validated to ensure reliability in both offline and API-connected modes.

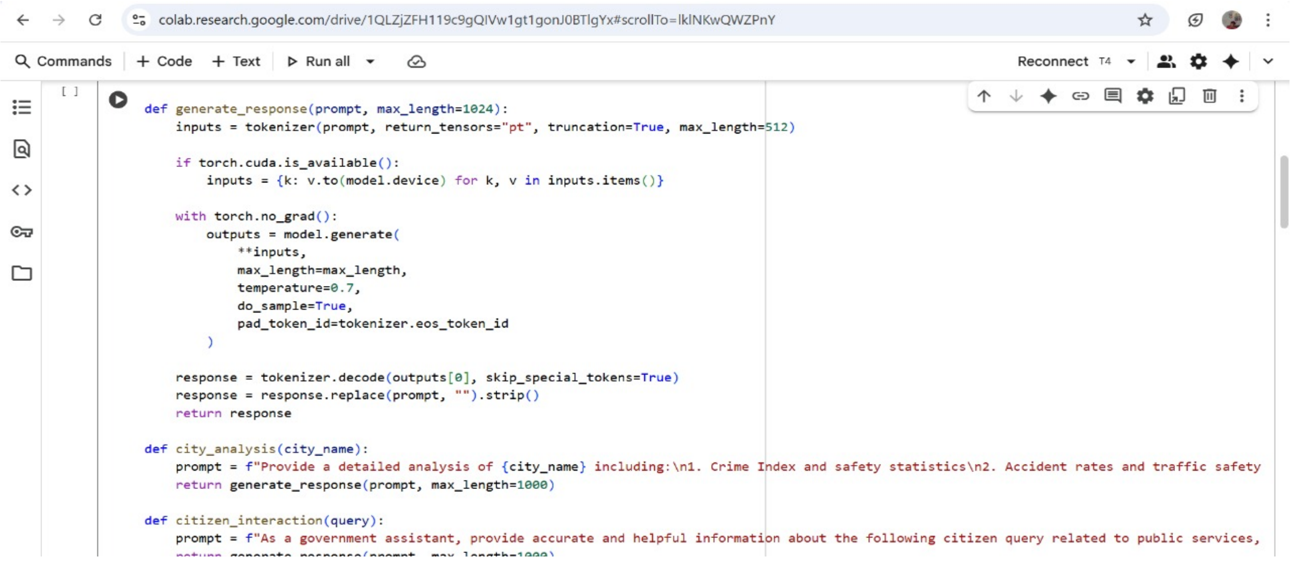
11.Source Code Screenshots:



colab.research.google.com/drive/1QLZjZFH119c9gQIVw1gt1gonJ0BTlgYx#scrollTo=lkINKwQWZPnY
Commands
+ Code
+ Text
Run all
Reconnect T4
[ ]
[ ]

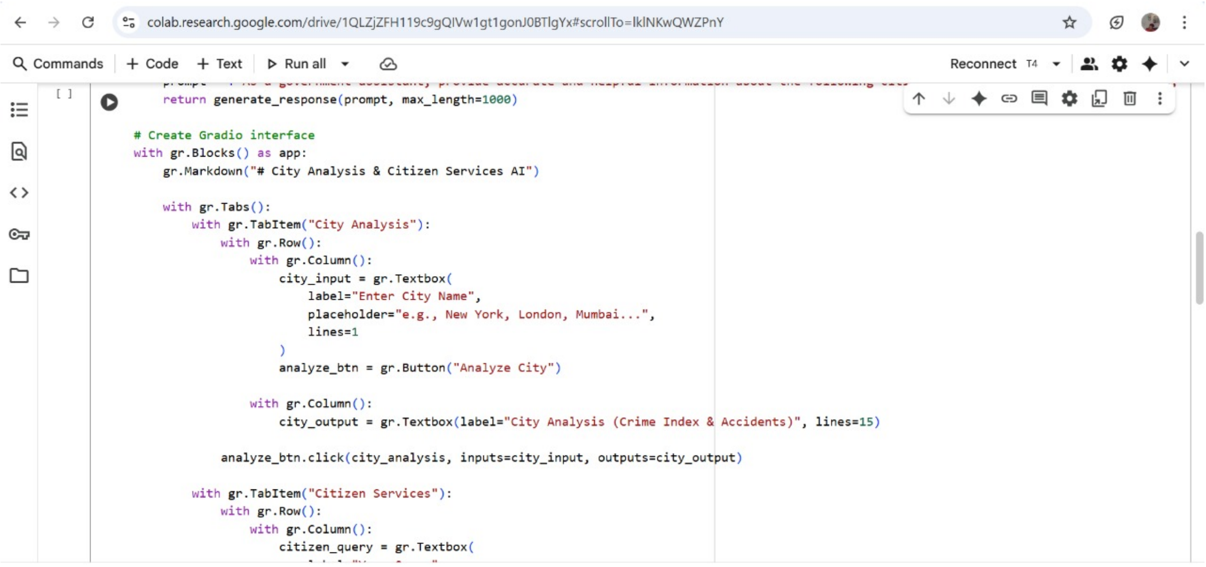
```
# run this project file in google collab by changing run type to T4 GPU

!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
```

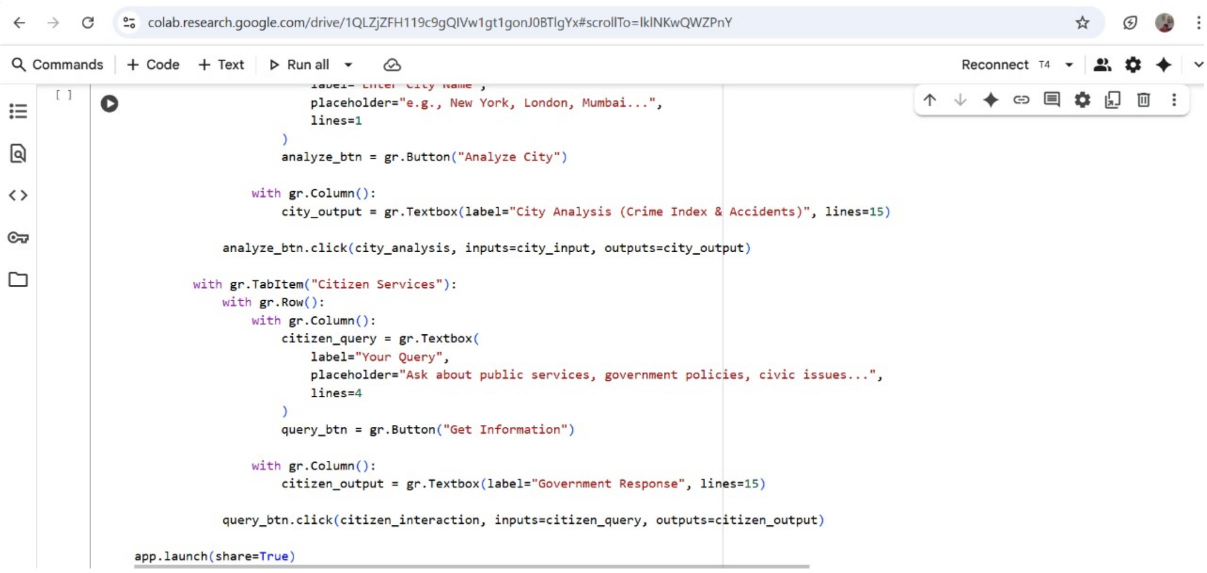


colab.research.google.com/drive/1QLZjZFH119c9gQIVw1gt1gonJ0BTlgYx#scrollTo=lkINKwQWZPnY
Commands
+ Code
+ Text
Run all
Reconnect T4
[ ]

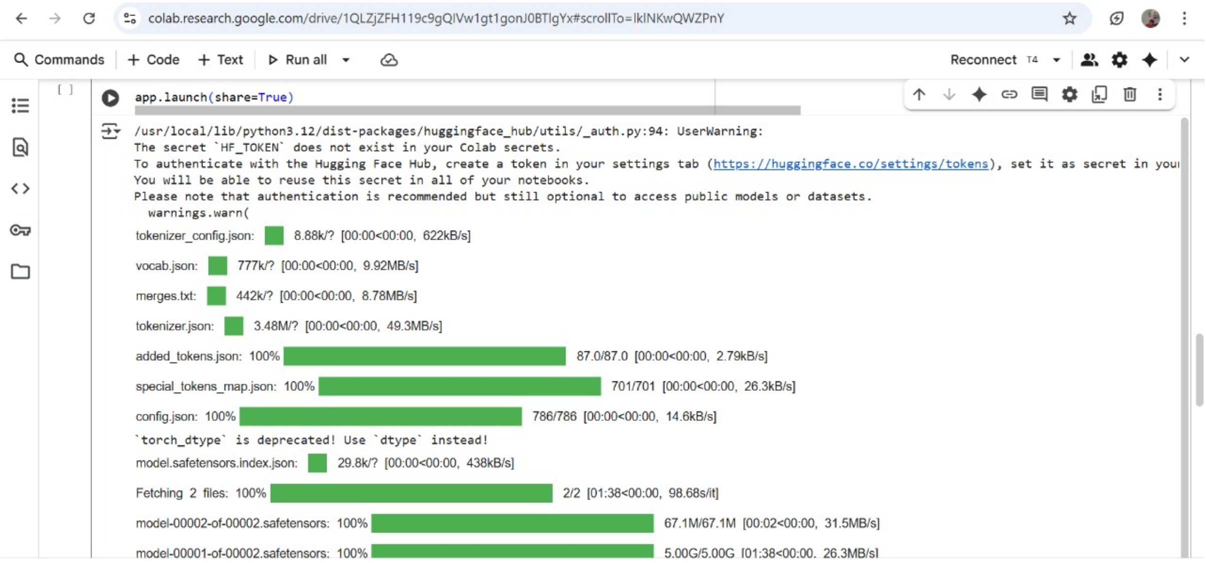
```
def generate_response(prompt, max_length=1024):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

    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 rates and traffic safety
    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 related to public services,
    return generate_response(prompt, max_length=1000)
```


colab.research.google.com/drive/1QLZjZFH119c9gQIVw1gt1gonJ0BTlgYx#scrollTo=lkINKwQWZPnY
Commands
+ Code
+ Text
▷ Run all
Reconnect T4
[ ]

```
        prompt = f"As a government assistant, provide accurate and helpful information about the following citi
        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",
```


colab.research.google.com/drive/1QLZjZFH119c9gQIVw1gt1gonJ0BTlgYx#scrollTo=lkINKwQWZPnY
Commands
+ Code
+ Text
▷ Run all
Reconnect T4
[ ]

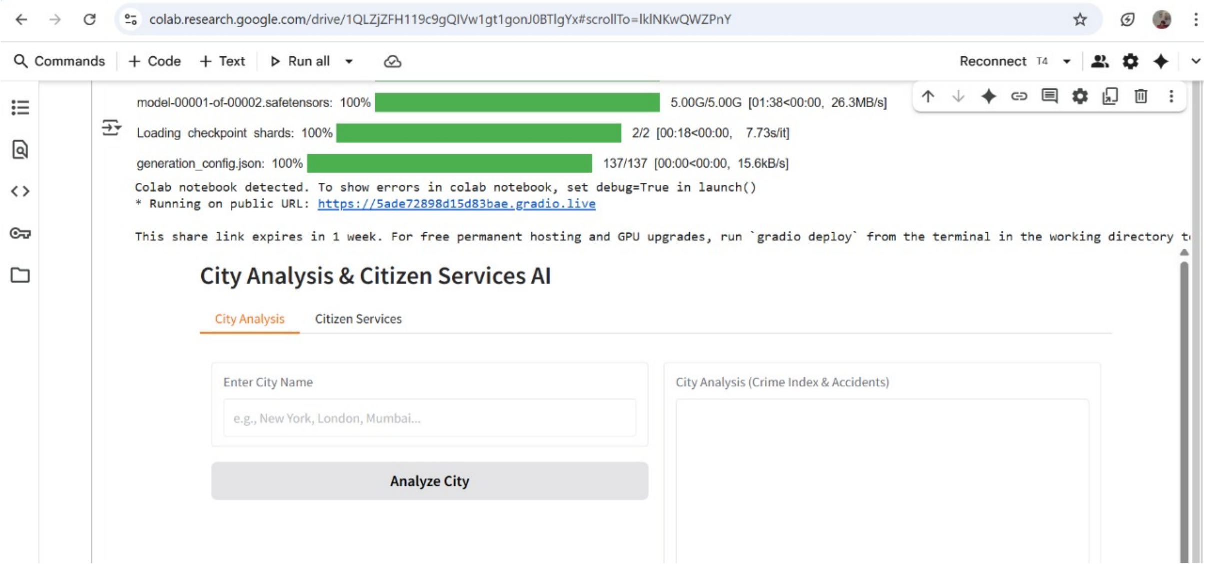
```
                        label="Enter City Name",
                        placeholder="e.g., New York, London, Mumbai...",
                        lines=1
                    )
                    analyze_btn = gr.Button("Analyze City")

                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)
```



colab.research.google.com/drive/1QLZjZFH119c9gQIVw1gt1gonJ0BTlgYx#scrollTo=lkINKwQWZPnY
Commands
+ Code
+ Text
Run all
Reconnect T4

```
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
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, 622kB/s]
vocab.json: 777k/? [00:00<00:00, 9.92MB/s]
merges.txt: 442k/? [00:00<00:00, 8.78MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 49.3MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 2.79kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 26.3kB/s]
config.json: 100% 786/786 [00:00<00:00, 14.6kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 438kB/s]
Fetching 2 files: 100% 2/2 [01:38<00:00, 98.68s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:02<00:00, 31.5MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:38<00:00, 26.3MB/s]
```



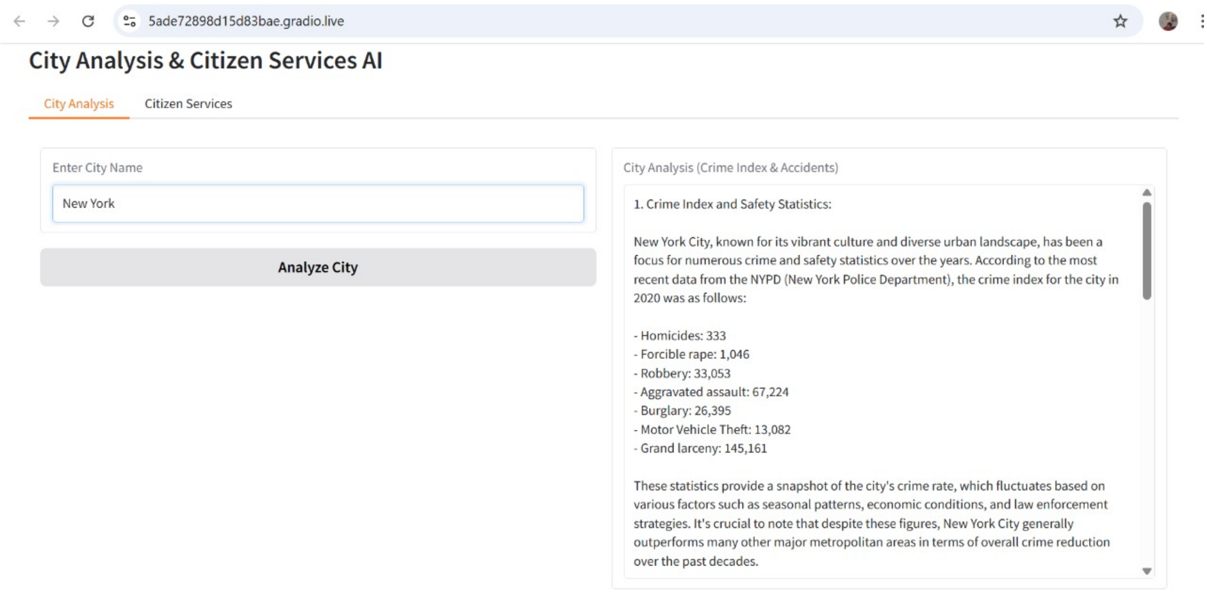
colab.research.google.com/drive/1QLZjZFH119c9gQIVw1gt1gonJ0BTlgYx#scrollTo=lkINKwQWZPnY
Commands
+ Code
+ Text
Run all
Reconnect T4

```
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:38<00:00, 26.3MB/s]
Loading checkpoint shards: 100% 2/2 [00:18<00:00,  7.73s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 15.6kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://5ade72898d15d83bae.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 t
```

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)

12.Source Output:


5ade72898d15d83bae.gradio.live
City Analysis & Citizen Services AI
City Analysis
Citizen Services
Analyze City

Enter City Name

New York

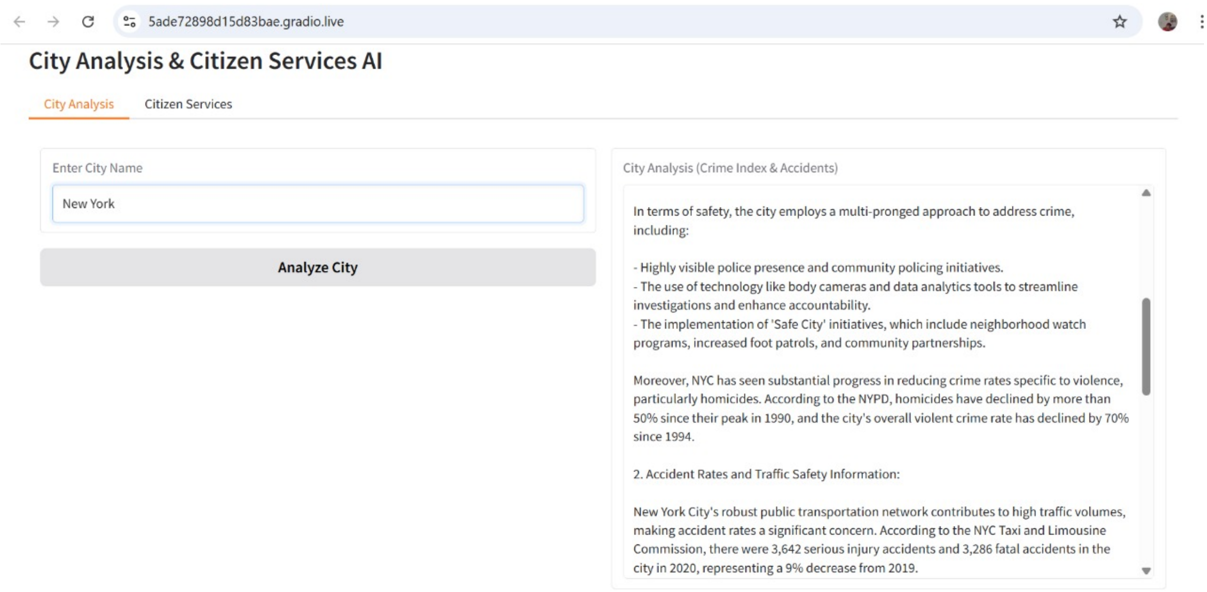
City Analysis (Crime Index & Accidents)

1. Crime Index and Safety Statistics:

New York City, known for its vibrant culture and diverse urban landscape, has been a focus for numerous crime and safety statistics over the years. According to the most recent data from the NYPD (New York Police Department), the crime index for the city in 2020 was as follows:

- Homicides: 333
- Forcible rape: 1,046
- Robbery: 33,053
- Aggravated assault: 67,224
- Burglary: 26,395
- Motor Vehicle Theft: 13,082
- Grand larceny: 145,161

These statistics provide a snapshot of the city's crime rate, which fluctuates based on various factors such as seasonal patterns, economic conditions, and law enforcement strategies. It's crucial to note that despite these figures, New York City generally outperforms many other major metropolitan areas in terms of overall crime reduction over the past decades.


5ade72898d15d83bae.gradio.live
City Analysis & Citizen Services AI
City Analysis
Citizen Services
Analyze City

Enter City Name

New York

City Analysis (Crime Index & Accidents)

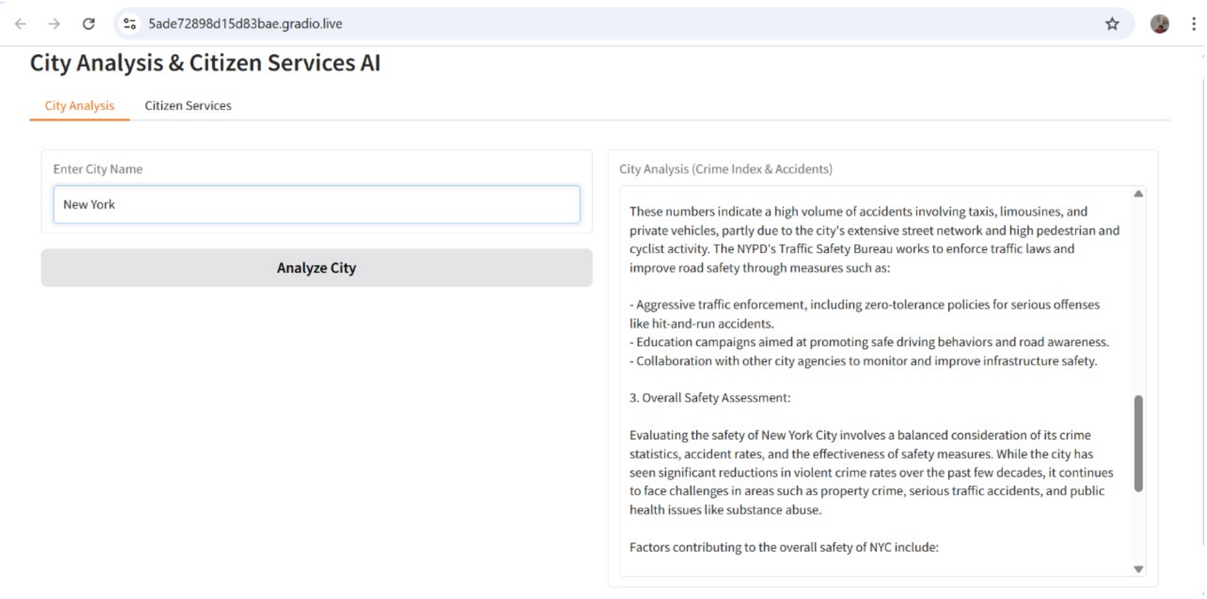
In terms of safety, the city employs a multi-pronged approach to address crime, including:

- Highly visible police presence and community policing initiatives.
- The use of technology like body cameras and data analytics tools to streamline investigations and enhance accountability.
- The implementation of 'Safe City' initiatives, which include neighborhood watch programs, increased foot patrols, and community partnerships.

Moreover, NYC has seen substantial progress in reducing crime rates specific to violence, particularly homicides. According to the NYPD, homicides have declined by more than 50% since their peak in 1990, and the city's overall violent crime rate has declined by 70% since 1994.

2. Accident Rates and Traffic Safety Information:

New York City's robust public transportation network contributes to high traffic volumes, making accident rates a significant concern. According to the NYC Taxi and Limousine Commission, there were 3,642 serious injury accidents and 3,286 fatal accidents in the city in 2020, representing a 9% decrease from 2019.


5ade72898d15d83bae.gradio.live
City Analysis & Citizen Services AI
City Analysis
Citizen Services
Analyze City

Enter City Name

New York

City Analysis (Crime Index & Accidents)

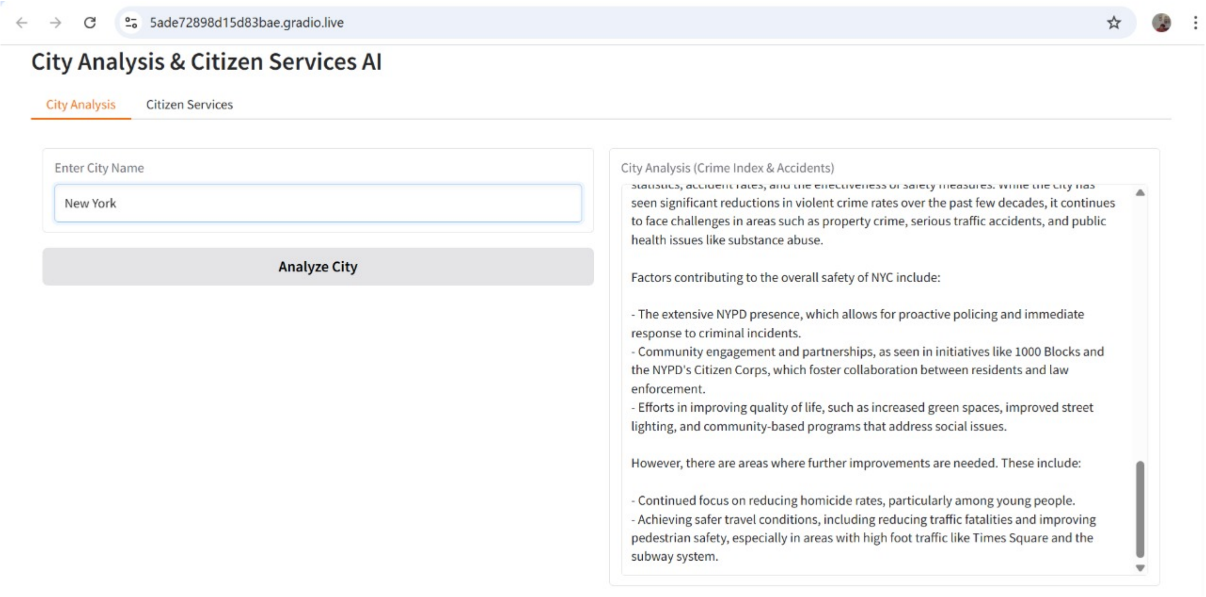
These numbers indicate a high volume of accidents involving taxis, limousines, and private vehicles, partly due to the city's extensive street network and high pedestrian and cyclist activity. The NYPD's Traffic Safety Bureau works to enforce traffic laws and improve road safety through measures such as:

- Aggressive traffic enforcement, including zero-tolerance policies for serious offenses like hit-and-run accidents.
- Education campaigns aimed at promoting safe driving behaviors and road awareness.
- Collaboration with other city agencies to monitor and improve infrastructure safety.

3. Overall Safety Assessment:

Evaluating the safety of New York City involves a balanced consideration of its crime statistics, accident rates, and the effectiveness of safety measures. While the city has seen significant reductions in violent crime rates over the past few decades, it continues to face challenges in areas such as property crime, serious traffic accidents, and public health issues like substance abuse.

Factors contributing to the overall safety of NYC include:


5ade72898d15d83bae.gradio.live
City Analysis & Citizen Services AI
City Analysis
Citizen Services
Analyze City

Enter City Name

New York

City Analysis (Crime Index & Accidents)

statistics, accident rates, and the effectiveness of safety measures. While the city has seen significant reductions in violent crime rates over the past few decades, it continues to face challenges in areas such as property crime, serious traffic accidents, and public health issues like substance abuse.

Factors contributing to the overall safety of NYC include:

- The extensive NYPD presence, which allows for proactive policing and immediate response to criminal incidents.
- Community engagement and partnerships, as seen in initiatives like 1000 Blocks and the NYPD's Citizen Corps, which foster collaboration between residents and law enforcement.
- Efforts in improving quality of life, such as increased green spaces, improved street lighting, and community-based programs that address social issues.

However, there are areas where further improvements are needed. These include:

- Continued focus on reducing homicide rates, particularly among young people.
- Achieving safer travel conditions, including reducing traffic fatalities and improving pedestrian safety, especially in areas with high foot traffic like Times Square and the subway system.

13. Future Enhancements:

- **User Sessions and History Tracking:** The project plans to add the ability to track user sessions and interaction history. This will allow for a more personalized experience.
- **Security:** For secure deployments, the project can integrate token-based authentication (JWT or API keys), OAuth2 with IBM Cloud credentials, and role-based access for different users (e.g., admin, citizen, researcher).