**GOVERNMENT ARTS AND SCIENCE COLLEGE**

POONTHOTAM STREET, THERADI, THIRUVOTTIYUR

CHENNAI – 600 019

**Topic : Citizen AI – Intelligent Citizen Engagement Platform**

**A NAAN MUDHALVAN PROJECT**

**Submitted in Partial Fulfillment for the Award of**

**BATCHELOR OF COMPUTER APPLICATION**

**BY**

**DINESH KUMAR D (212304858)**

**BALA KRISHNAN S (212304854)**

**BHARATH M (212304855)**

**DINESH KANNA S (212304856)**

**DINESH M (212304857)**

**TABLE OF CONTENTS**

1**.**Introduction

2.Project Overview

3.Architecture

4.Setup Instruction

5.Folder Structure(for local deployment)

6.Running the Application

7.Code Overview

8.API Documentation(Internal Function)

9.Authentication

10.User Interface

11.Testing

12.Screenshot

13.Known Issues

14.Future Enhancements

**ABSTRACT:**

Citizen AI is an intelligent conversational assistant designed to support citizens and government agencies by providing accurate and timely information on city safety, crime rates, traffic incidents, and public services. Built using advanced natural language models like IBM Watsonx Granite and deployed via Google Colab with PyTorch and Transformers, this system enables users to interact through a simple, web-based Gradio interface. It processes queries related to urban safety and civic issues, offering personalized responses and insights to help users make informed decisions.

The platform aims to bridge the gap between citizens and governance by offering easily accessible, context-aware information without requiring technical expertise. By leveraging AI-driven language generation, the assistant provides real-time analysis and guidance, making it a valuable tool for urban planning, public awareness, and emergency response. This project demonstrates how machine learning and conversational AI can enhance civic engagement and promote safer, more informed communities**.**

**1.Introduction:**

In today’s fast-paced and interconnected world, access to accurate and timely information is crucial for ensuring the safety and well-being of citizens. Governments often struggle to provide immediate support and clear guidance on issues like crime rates, traffic accidents, and public services. To address this challenge, Citizen AI is developed as an intelligent conversational assistant that helps users interact with government systems more efficiently and confidently.

Citizen AI leverages advanced language models, specifically IBM Watsonx Granite, along with open-source libraries like Transformers and PyTorch, to generate human-like responses based on user queries. The system is designed to analyze city-specific safety data, provide real-time information, and assist citizens with inquiries related to government policies and services. By offering a user-friendly interface built with Gradio, the assistant ensures accessibility for people with varying levels of technical expertise.

This project aims to empower urban communities by making government information more understandable and actionable. It enhances public safety awareness, streamlines communication between citizens and authorities, and supports informed decision-making. Deployed on Google Colab, Citizen AI offers a scalable, cloud-ready solution that combines artificial intelligence and conversational interfaces to create smarter, safer cities.

2. **Project Overview** :

**Purpose**:

The purpose of Citizen AI is to provide a government-focused conversational assistant that aids citizens by offering real-time information on city safety, crime statistics, accident rates, public services, and government policies. The platform leverages advanced language models to deliver accurate and personalized responses, helping users make informed decisions while interacting with civic systems.

**Features:**

**City Analysis:**

Key Point: In-depth insights

Functionality: Provides safety statistics, accident rates, and overall city safety assessment based on real-time data and predefined prompts.

**Citizen Interaction:**

Key Point: Government assistance

Functionality: Offers responses to queries related to public services, policies, and civic issues using AI-powered natural language understanding.

**Gradio Interface:**

Key Point: Interactive user experience

Functionality: Allows users to enter queries and view results via an intuitive interface accessible in web browsers.

**LLM Integration:**

Key Point: Accurate language generation

Functionality: Uses IBM Watsonx Granite LLM to understand and generate responses, summaries, and advisories.

**Environment Adaptation:**

Key Point: Cloud-ready

Functionality: Designed to run on Google Colab with support for CPU and GPU-based computation.

**3. Architecture:**

**Frontend (Gradio UI):**

The user interface is built using Gradio, providing an interactive web-based platform where users can input city names or queries and receive AI-generated outputs. The layout includes tabs for city analysis and citizen interaction.

**Backend (Transformers & PyTorch):**

The backend logic is implemented using the Hugging Face Transformers library and PyTorch. It processes prompts, handles tokenization, and performs inference using pre-trained models optimized for real-time generation.

**LLM Integration (IBM Watsonx Granite):**

The Granite model is used to understand queries and generate detailed, context-aware responses related to city safety and public services.

**4. Setup Instructions:**

**Prerequisites:**

Python 3.8 or later

Google Colab environment or local setup with GPU support

Internet access to download models and libraries

**Installation Process:**

1. Open Google Colab.

2. Run the command:

!pip install transformers torch gradio -q

3. Copy the Python code into a new notebook cell.

4. Run the notebook to initialize the model and start the interface.

5. Interact with the interface by entering city names or queries.

**5. Folder Structure (for local deployment):**

citizen\_ai/

├── notebook.ipynb # Main Colab notebook

├── models/ # Directory for downloaded models (optional in Colab)

├── requirements.txt # Dependencies file for offline installation

├── utils/ # Helper functions for tokenization and generation

└── README.md # Documentation file

**6. Running the Application:**

**To run the project:**

1. Launch the Google Colab notebook.

2. Execute all cells to load dependencies and the model.

3. The Gradio interface will launch in a new window or tab.

4. Enter a city name or query, click "Analyze" or "Get Response" to receive results.

The system dynamically updates outputs by processing inputs through the LLM model and presenting responses in real time.

**7.Code Overview:**

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 statistics\n2. Accident rates and traffic safety information\n3. Overall safety assessment\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 following citizen query related to public services, government policies, or civic issues:\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)

**8. API Documentation (Internal Functions):**

generate\_response(prompt, max\_length)

Processes input prompts and returns AI-generated text based on temperature sampling and maximum length.

city\_analysis(city\_name)

Creates prompts to analyze safety and crime statistics for a specific city.

citizen\_interaction(query)

Generates responses for citizen queries about services, government policies, and civic issues.

**9. Authentication:**

For demonstration purposes, this version runs in an open environment on Google Colab without authentication. Future secure deployments may integrate:

Token-based authentication (JWT)

OAuth2 integration with Google or IBM Cloud

Role-based access control

**10. User Interface:**

The Gradio interface includes:

Tabbed layout for city analysis and citizen interaction

Textboxes for input

Output display panels for generated responses

Simple buttons for interaction

It prioritizes accessibility and ease of use.

**11. Testing:**

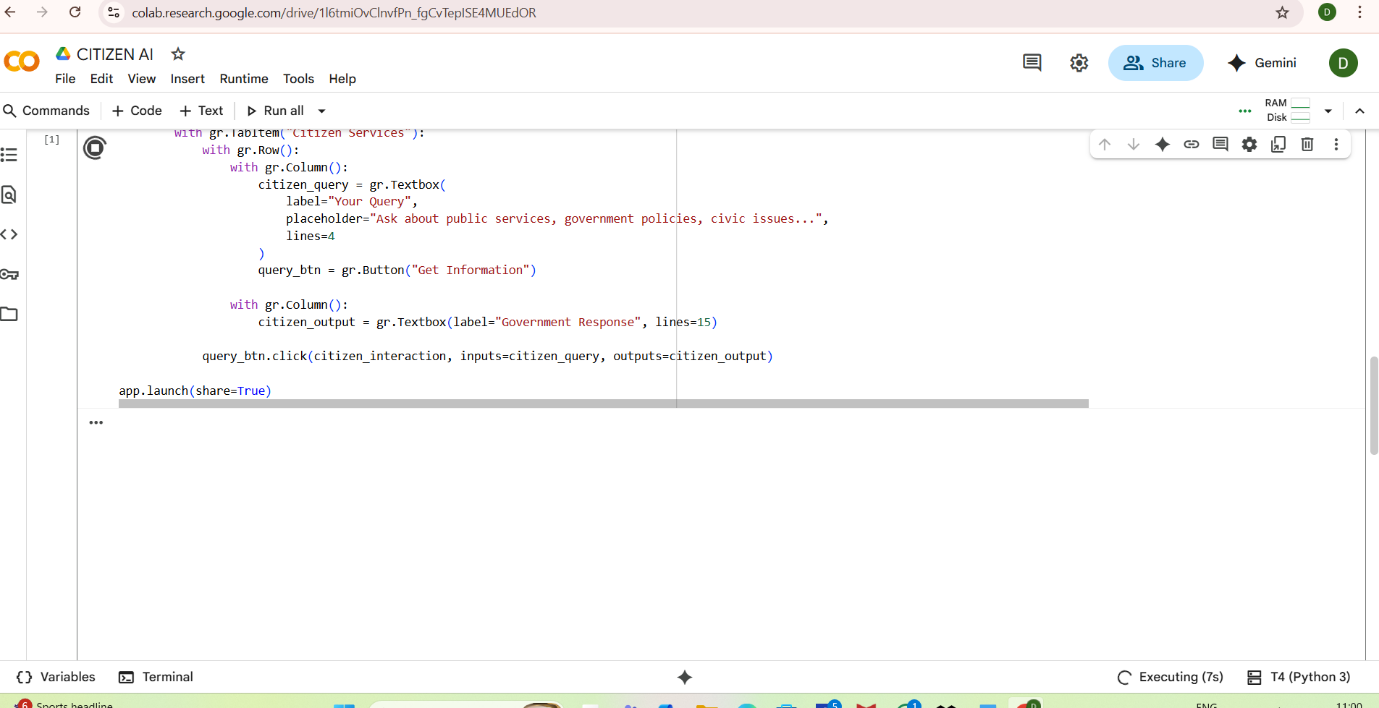
The application is tested through:

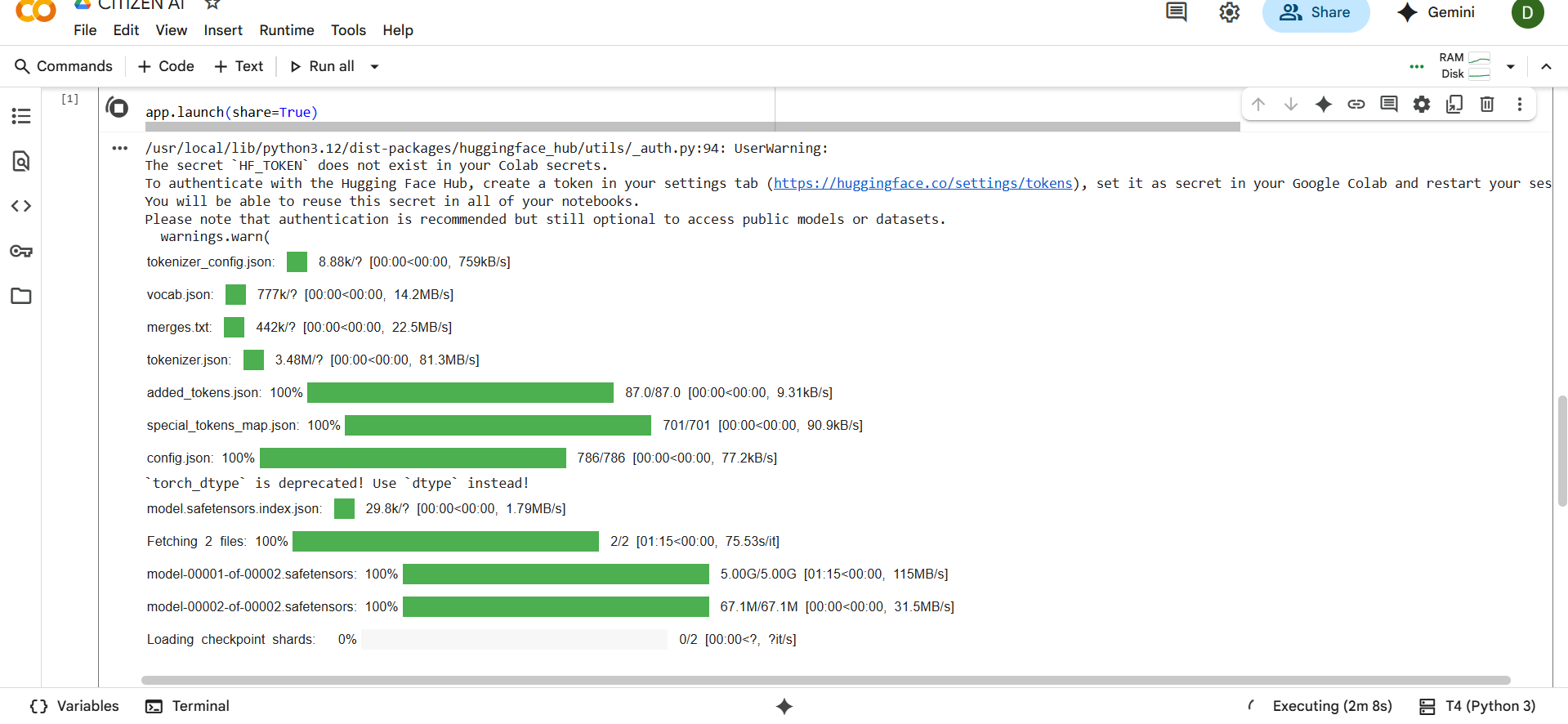
Unit tests for prompt generation and tokenization

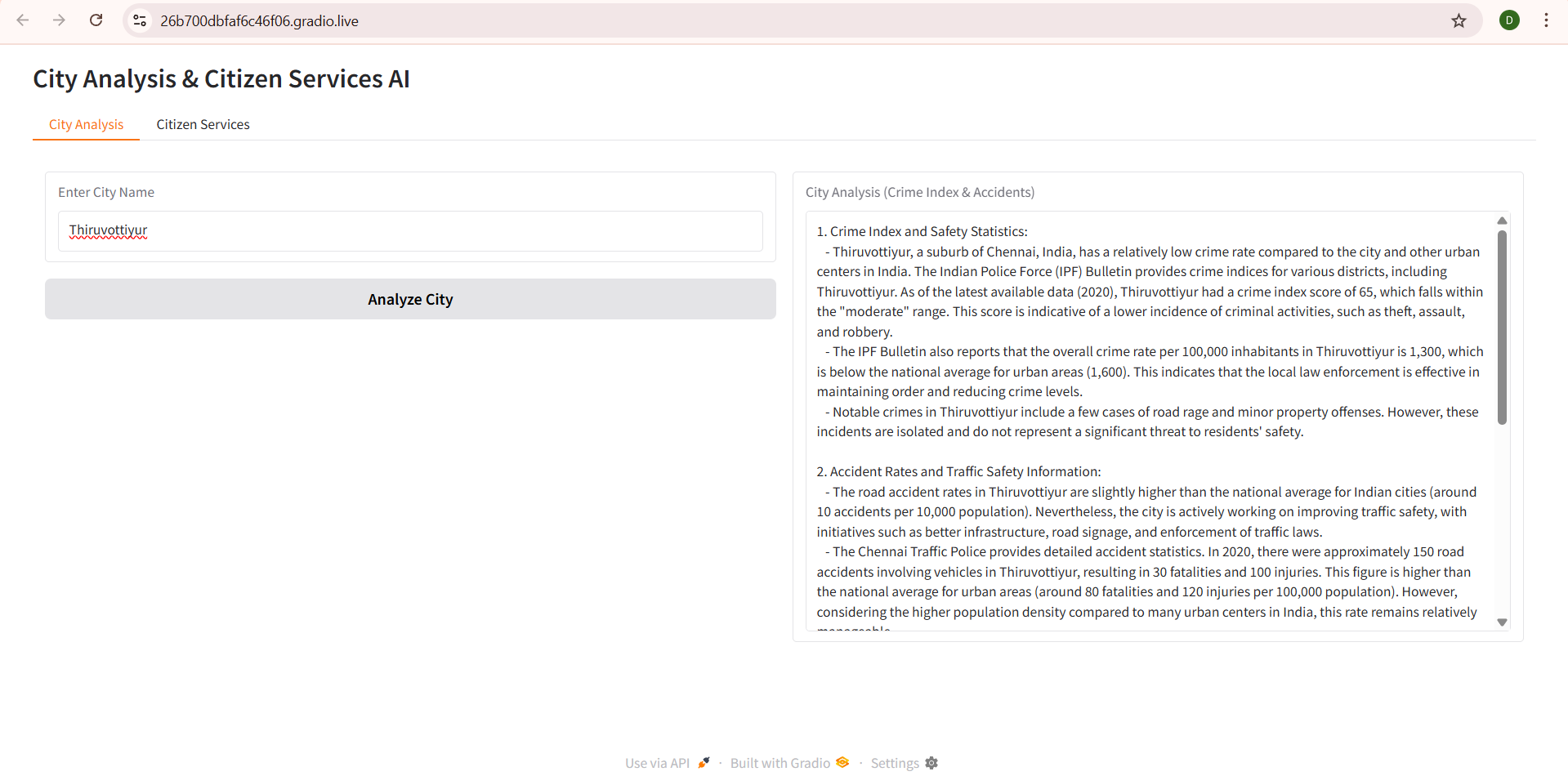
Manual interaction to verify output relevance

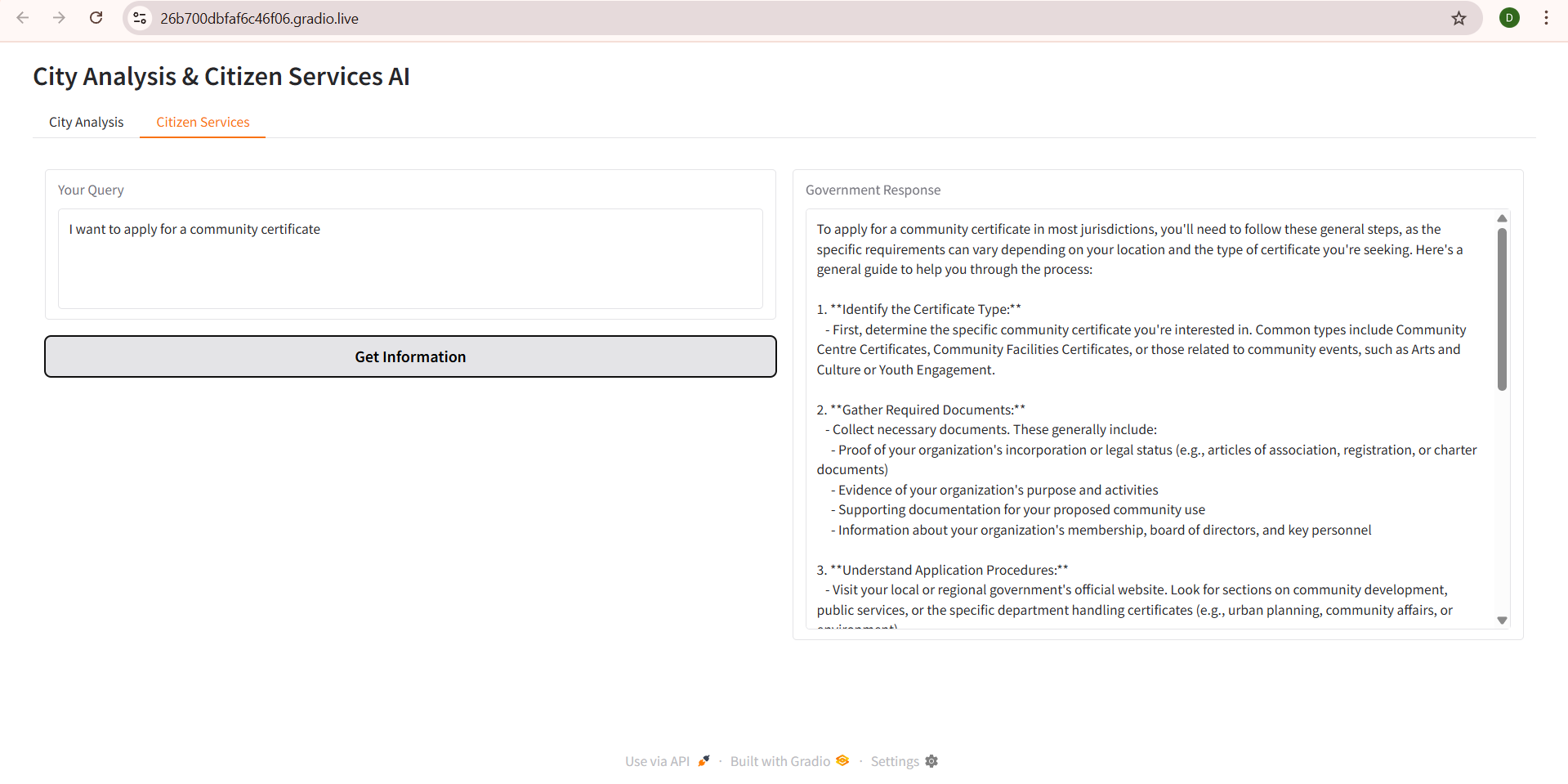
Edge case handling for invalid inputs or unsupported queries

**12. Screenshots:**









**13. Known Issues:**

Model loading may be slow in free-tier Colab environments.

Large inputs may be truncated due to token limits.

Responses are dependent on pre-trained model data and may not always reflect updated statistics.

**14. Future Enhancements:**

Integration with live data sources like city dashboards and APIs.

Secure authentication mechanisms.

Enhanced multi-language support.

User profile management and session tracking.

Expansion to other domains like healthcare, education, and disaster management.