CitizenAI – Intelligent Citizen Engagement Platform

# **1. Introduction**

**PROJECT TITLE :**CitizenAI – Intelligent Citizen Engagement Platform

**TEAM LEADER :**

**ATCHAYA B**

**TEAM MEMBERS :**

**DHANUJA S**

**DHAVAMUGI V**

**DHIVYA DHARSHINI C D**

# 2. Project Overview

• Purpose :  
The purpose of CitizenAI is to empower cities and their residents with an AI-driven platform that simplifies access to civic services and information. By leveraging the IBM Granite model from Hugging Face along with real-time interactions, the platform helps users get quick answers to government-related queries, understand policies, and receive guidance in multiple languages. It aims to enhance citizen engagement, improve access to public services, and support informed decision-making by city officials.  
  
• Features:  
  
Conversational Interface  
Key Point: Natural language interaction  
Functionality: Allows citizens and officials to ask questions and get responses in plain language powered by AI.  
  
Policy Summarization  
Key Point: Simplified understanding  
Functionality: Converts complex government queries into actionable, understandable responses.  
  
Multilingual Support  
Key Point: Communication in different languages  
Functionality: Provides responses in English, Spanish, or French based on user selection.  
  
Downloadable Reports  
Key Point: Documentation of insights  
Functionality: Enables users to download analysis reports and query responses as text files.  
  
Custom Analysis Options  
Key Point: Flexible interaction  
Functionality: Offers city-specific reports based on user-selected aspects like crime index, air quality, and healthcare access.  
  
Adaptive Detail Levels  
Key Point: Tailored responses  
Functionality: Provides options to get detailed reports or brief summaries depending on user preference.  
  
Model Integration  
Key Point: AI-driven insights  
Functionality: Uses IBM Granite’s LLM for understanding and generating accurate and contextual responses.

# 3. Architecture

Frontend (Gradio UI):  
The interface is built using Gradio, providing an intuitive dashboard that supports input forms, checkboxes, dropdowns, and buttons. It features tabs for city analysis and citizen services, with output areas and download options for users to easily view and retrieve generated reports.  
  
Backend (Transformers & Torch):  
The backend is powered by Hugging Face’s Granite model, running with PyTorch for efficient computation. It processes user prompts, generates responses, and manages model interactions while leveraging GPU support if available.  
  
Model Integration (IBM Granite LLM):  
Granite LLM models are utilized for natural language understanding and response generation. Carefully crafted prompts help generate city analysis reports and government query responses.  
  
Data Handling:  
Inputs are tokenized using Hugging Face’s AutoTokenizer, and outputs are decoded to produce user-friendly text.

# 4. Setup Instructions

Prerequisites:  
o Python 3.9 or later  
o pip for installing dependencies  
o Internet access for downloading models  
o Google Colab environment or local Python setup  
  
Installation Process:  
o Install required libraries using `!pip install transformers torch gradio`  
o Load the Granite model from Hugging Face using AutoTokenizer and AutoModelForCausalLM  
o Run the interface using Gradio’s Blocks method  
o Launch and share the interface using `app.launch(share=True)`

# 5. Folder Structure

app/ # Contains the backend logic, including model interaction  
models/ # Storage for model files or cached weights  
notebooks/ # Contains Jupyter/Colab notebooks  
reports/ # Stores downloadable reports like CitizenAI\_Report.txt  
  
Main script:  
CitizenAI.py – Handles both frontend and backend functionalities using Gradio and Transformers.

# 6. Running the Application

To start the project:  
➢ Install the dependencies using pip in a Colab or local environment  
➢ Load the IBM Granite model  
➢ Run the Gradio app and open the provided link  
➢ Use the City Analysis or Citizen Services tab to input queries  
➢ View generated responses and download reports  
  
All interactions are processed using AI models and dynamically updated in real-time.

# 7. API Documentation

This project does not expose REST APIs but uses Python functions connected through Gradio interface components:  
  
Functions:  
- generate\_response(prompt, max\_length) – Generates text based on the input prompt  
- city\_analysis(city\_name, aspects, detail\_level, language) – Provides city-specific insights  
- citizen\_interaction(query, detail\_level, language) – Answers user queries with government-like assistance  
- download\_report(text) – Saves and downloads reports  
  
These functions are linked to buttons for user interaction within the Gradio interface.

# 8. Authentication

The application runs in an open environment suitable for demonstration purposes. Future enhancements could include:  
• User authentication using API keys or OAuth  
• Role-based access controls  
• Session management and history tracking

# 9. User Interface

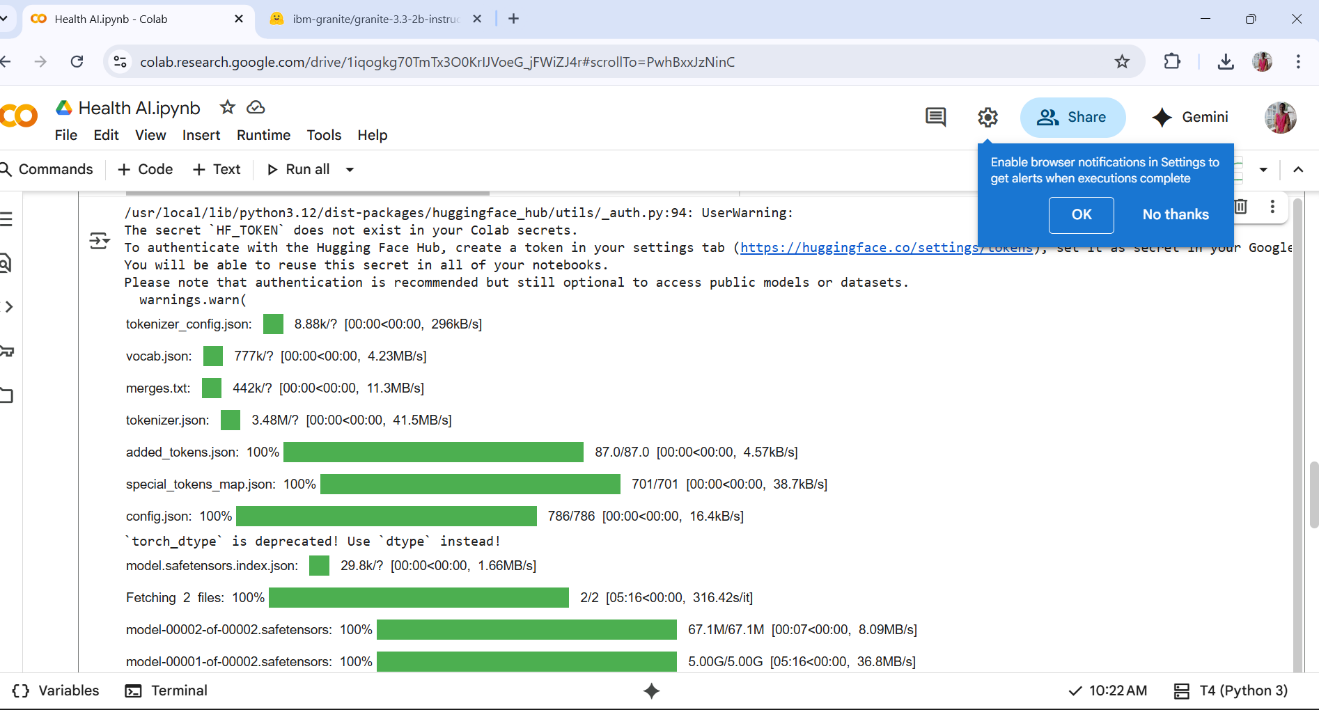
The interface is designed for simplicity and accessibility:  
- Sidebar navigation is replaced with tabs for city and citizen services  
- Inputs include textboxes, checkboxes, radio buttons, and dropdowns  
- Outputs are displayed in multiline textboxes  
- Reports can be downloaded easily as text files  
  
The focus is on providing users with immediate and relevant assistance.

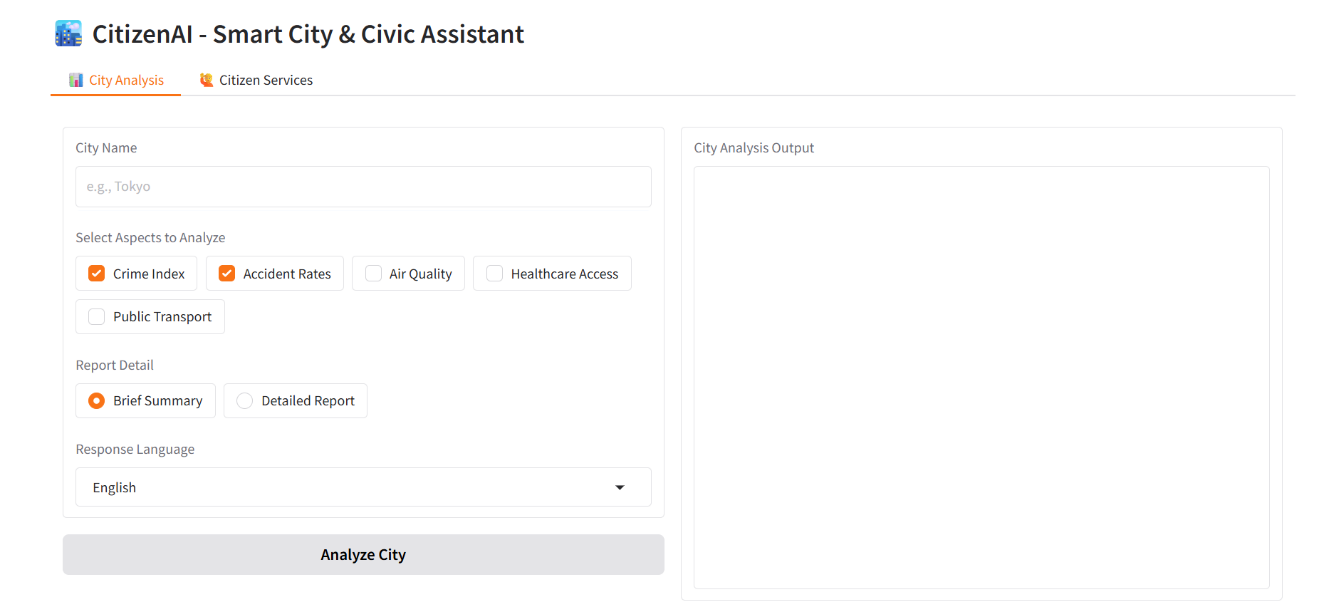
# 10. Testing

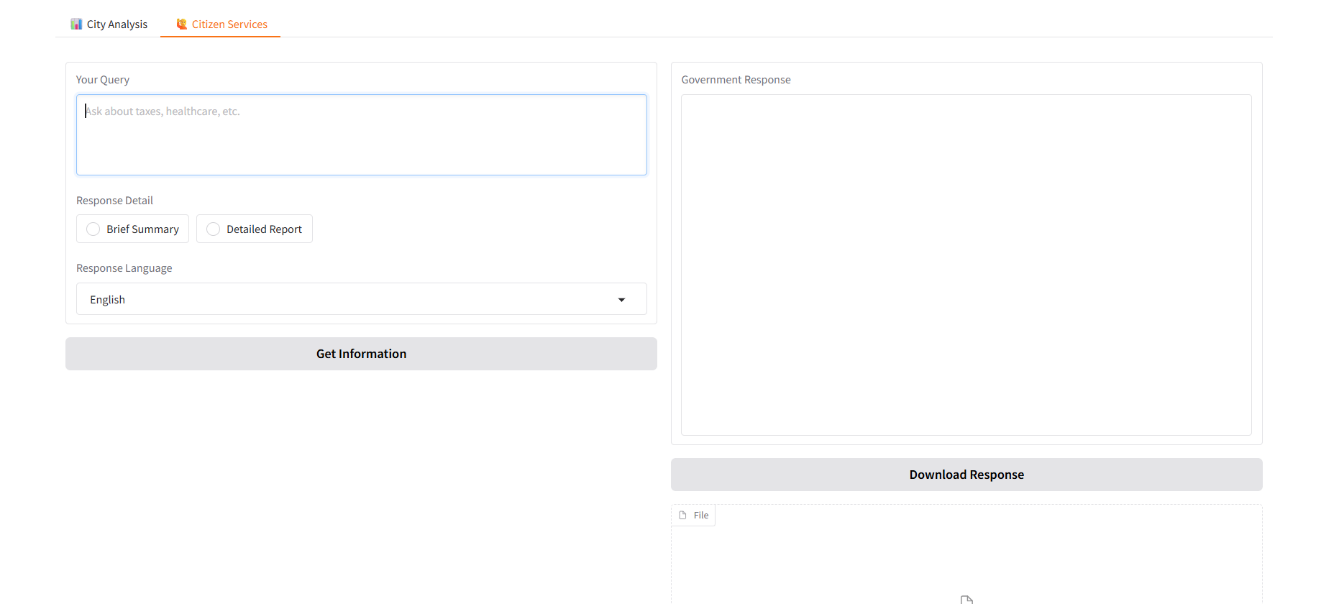
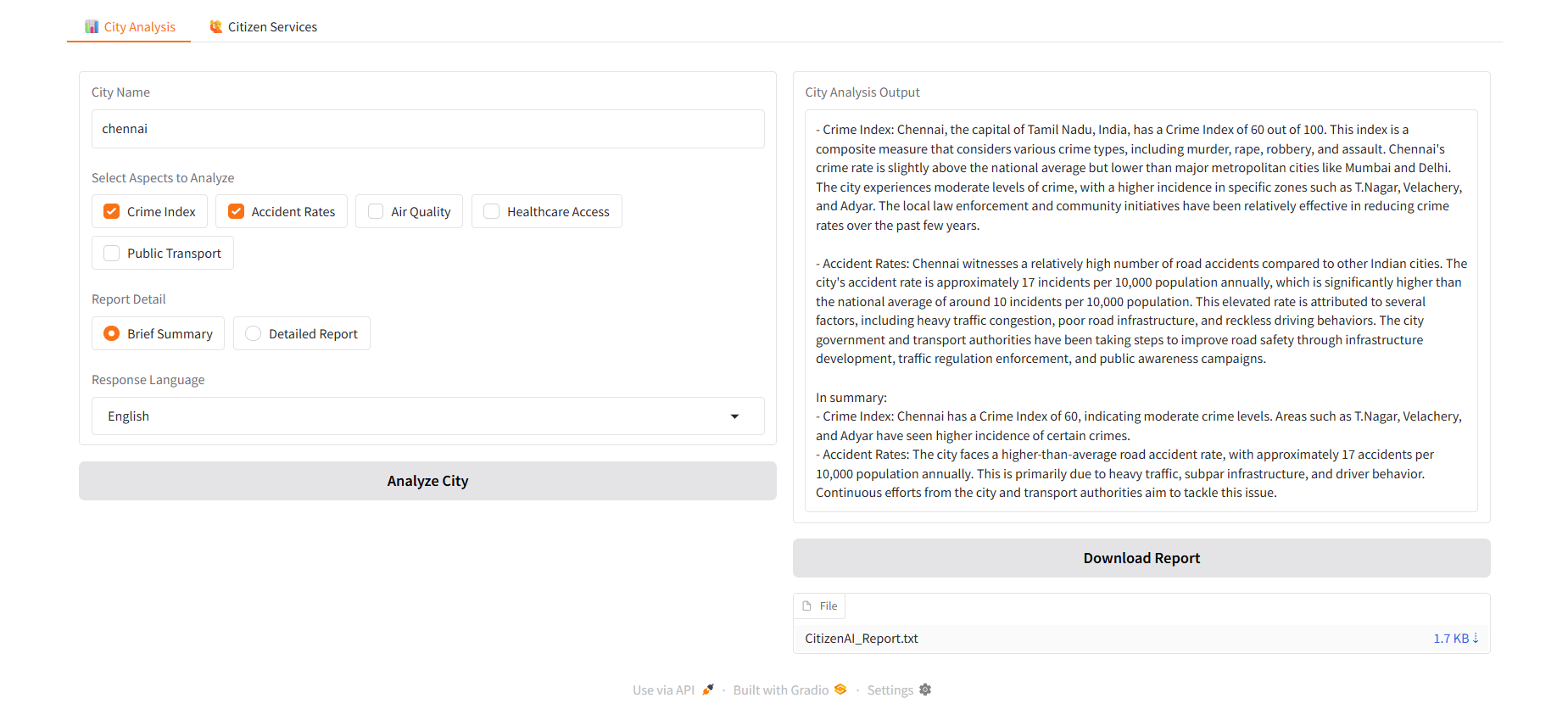
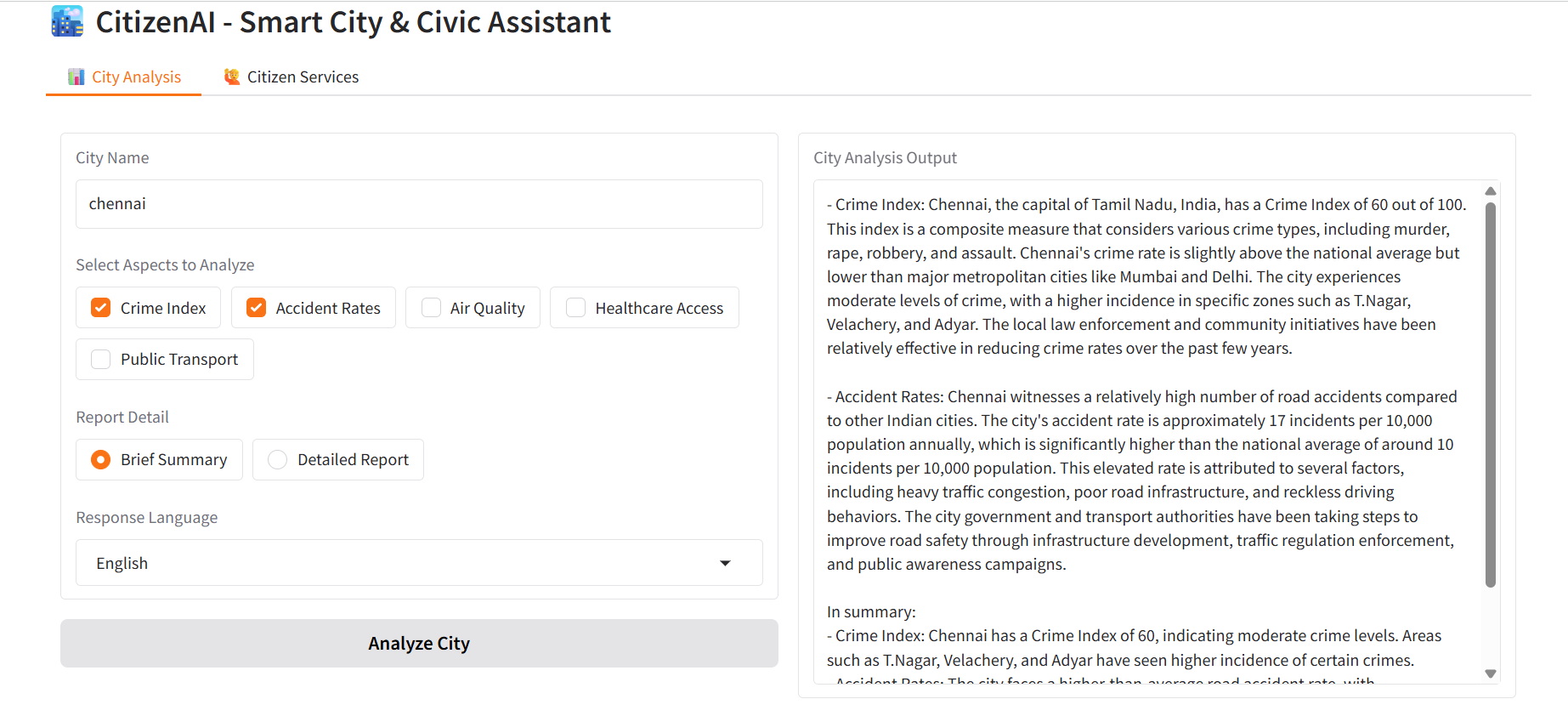
Testing includes:  
Unit Testing: For prompt construction and response generation  
Interface Testing: Ensuring proper handling of user inputs in Gradio  
Manual Testing: Uploading queries, selecting options, and verifying outputs  
Performance Testing: Checking GPU availability and fallback to CPU processing  
Edge Cases: Handling empty inputs, unsupported languages, and long queries

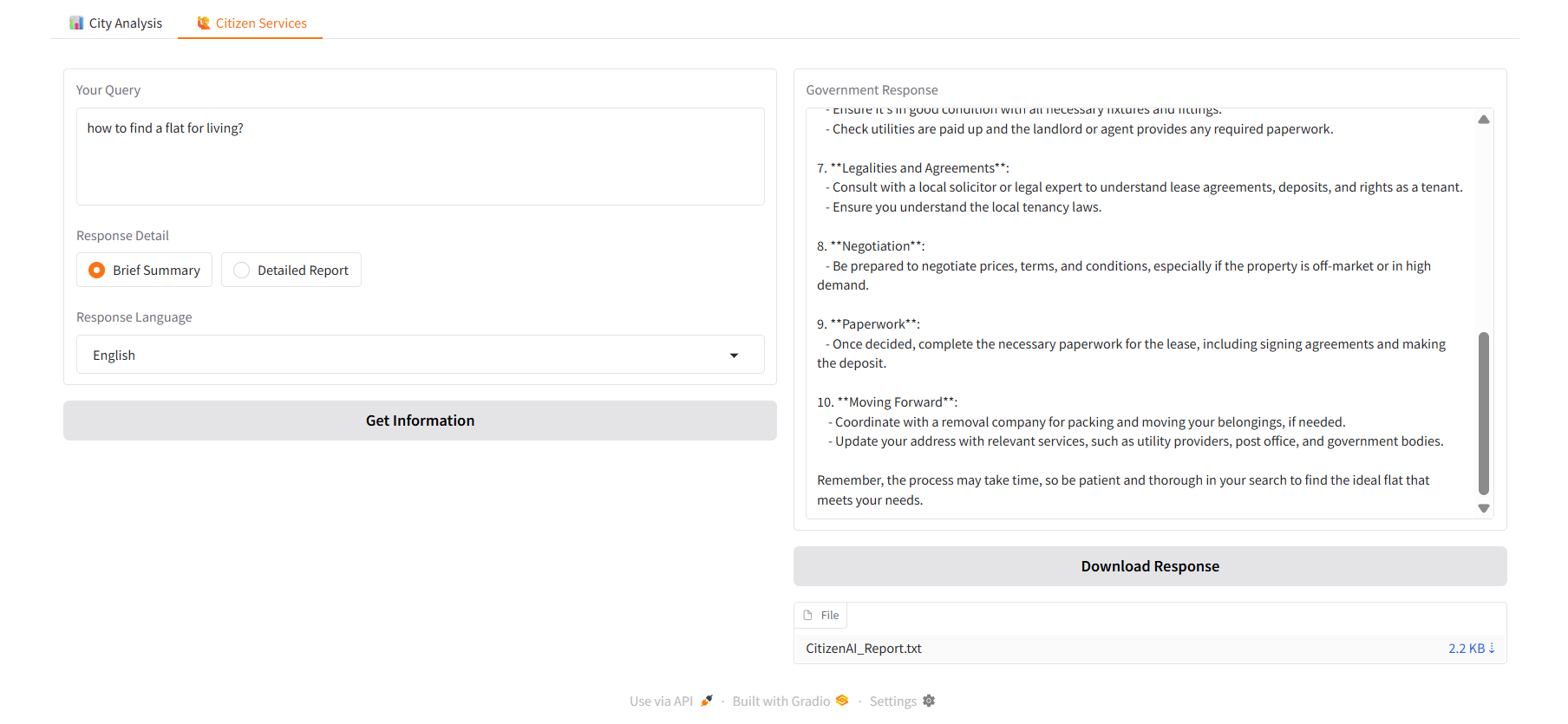
# 11. Screenshots

(Screenshots to be inserted here from the running Gradio interface showing the city analysis and citizen services tabs.)









# 12. Known Issues

• Model loading may be slow depending on hardware and network speed  
• Limited language support (English, Spanish, French)  
• Responses may vary due to sampling during generation  
• No authentication, so it is not suitable for production environments without security measures

# 13. Future Enhancements

• Add support for more languages and local dialects  
• Integrate with government APIs for real-time data retrieval  
• Implement user profiles with history and preferences  
• Improve error handling for invalid inputs and longer conversations  
• Add visualization charts for city analysis metrics  
• Deploy on cloud platforms with authentication and scalable architecture