Project Report: City Analysis & Citizen Services AI

Team Leader: Indhu Prabha T
Team Members: Rajalaksmi R, Shalini J, Sri Lakshmi Sruthi A

1. Introduction

The project 'City Analysis & Citizen Services AI' is designed to empower citizens and officials with actionable insights about urban safety and civic services. By leveraging IBM Granite LLM and Gradio, the system provides city-specific crime and accident analysis, and answers queries related to public services and policies in real time.

2. Project Overview

Purpose: To build an AI-powered assistant that provides detailed city analysis, including crime and accident statistics, and assists citizens with government-related queries. This system enhances transparency, supports urban planning, and improves civic engagement.

Features:
• Conversational Interface – Citizens interact via natural language.
• City Analysis – Provides crime index, accident rates, and safety insights.
• Citizen Services – Answers queries about public services and civic issues.
• AI-Powered Responses – Uses IBM Granite 3.2-2B Instruct model for contextual understanding.
• User-Friendly Interface – Built with Gradio for accessible web-based interaction.

3. Architecture

```
Frontend (Gradio):
- Provides interactive UI with tabs for City Analysis and Citizen Services.

Backend (HuggingFace Transformers with PyTorch):
- Loads IBM Granite model for natural language generation.

Core Modules:
- City Analysis Function: Generates safety insights (crime, accidents, safety score).
- Citizen Interaction Function: Handles queries on policies, services, and civic issues.

Execution Flow:
- User enters input (city name or query).
- Input is processed via tokenizer and Granite model.
- Model output is decoded and displayed in Gradio interface.
```

4. Setup Instructions

```
Prerequisites:
- Python 3.9 or later
- pip package manager
- HuggingFace Transformers, Gradio, PyTorch

Installation Process:
1. Clone the repository.
2. Install dependencies: pip install -r requirements.txt
3. Run the Python script to launch the app.
4. The Gradio app provides a shareable link for access.
```

5. Core Implementation Code

```
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\n
    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 citiz
    return generate_response(prompt, max_length=1000)

# 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, Mu
                    analyze_btn = gr.Button("Analyze City")
                with gr.Column():
                    city_output = gr.Textbox(label="City Analysis", 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 policies or servic
                    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)
```

6. Running the Application

Run the Python script. Gradio will start a local server and generate a public shareable link. Navigate to the link to access City Analysis and Citizen Services tabs.

7. Future Enhancements

• Integration with live crime and traffic APIs for real-time statistics.
• Addition of more civic modules (waste management, water supply updates).
• Multilingual support for better accessibility.
• Role-based authentication for officials and citizens.
• Export analysis results directly as PDF reports from the interface.