Citizen AI: Intelligent Citizen Engagement Platform Using IBM Granite

Project Documentation

Introduction

Project title: Citizen AI: Intelligent Citizen Engagement Platform Using IBM Granite

Team members:

Team member 1: Lingeswaran K
Team member 2: Tinesh Kumar E
Team member 3: Manoj D
Team member 4: Aswathaman D

Project Overview

Purpose:

The purpose of Citizen AI is to provide users with an intelligent, AI-powered assistant capable of answering queries related to government services, civic issues, and public engagement. By leveraging IBM's Granite LLM and real-time AI processing, the platform aims to deliver accurate, accessible, and user-friendly guidance while supporting authorities in improving transparency and citizen satisfaction.

Features:

1.Conversational Interface:

Key Point: Natural language interaction
Functionality: Users can ask questions about government services, policies, or civic issues and receive AI-generated responses.

2.Service Guidance:

Key Point: Citizen-focused information
Functionality: Provides step-by-step guidance for services like Aadhaar application, Voter ID registration, RTI queries, and transportation-related information.

3.Quick Question Buttons:

Key Point: Easy access to common queries
Functionality: Pre-set buttons for frequently asked topics such as Aadhaar, RTI Act, Voter ID, and Railway ticket booking.

4.Chat Management

Key Point: Enhanced user control
Functionality: Options to send queries, clear chat history, and download conversations as a text file.

5.Gradio UI

Key Point: User-friendly interface
Functionality: Gradient-based chatbot design with text input, quick action buttons, and real-time responses.

Architecture:

Frontend (Gradio):

Chat-based web interface with input textbox and functional buttons (Send, Clear, Download, Quick Questions).

Inputs: User queries in natural language.

Outputs: AI-generated responses displayed in the chat, with option to download as text.

Backend (Python + Transformers):

1.Processes user queries and generates AI responses using IBM Granite LLM.

2.Handles model loading and GPU optimization if available.

LLM Integration (IBM Granite – Hugging Face Model):

Model: ibm-granite/granite-1b-code-instruct (lightweight and efficient for Colab).

1.Performs natural language understanding and response generation. Deployment (Google Colab):

2.Runs in Google Colab with T4 GPU for smooth AI processing.

3. Application can be shared publicly using launch(share=True).

Setup Instructions Prerequisites:

1.Python 3.9 or later

2.pip for package installation

3.Internet connection (for downloading model)

4.GPU recommended for faster response (T4 GPU in Colab)

Installation Process:

1. Clone the repository.

2. Install dependencies: pip install transformers torch gradio

3. Run the Gradio app: python citizen_ai.py

4. Open the provided local URL or use the public share link

Folder Structure:

citizen_ai.py – Main Gradio app and UI layout

requirements.txt – Python dependencies

Running the Application:

1. Launch the Python script citizen_ai.py or run the notebook in Colab.

2. Gradio interface opens in the browser.

3. User flow: - Enter a query → Click Send → View AI response.

4. Use Quick Question Buttons for instant responses.

5. Use Clear Chat to reset conversation.

API Documentation:

(Note: Current version runs as a Gradio interface; no REST API implemented. Optional future enhancement could add FastAPI backend.)

Inputs: User query (Textbox or Quick Question button)

Outputs: AI response (Text), Chat history (Downloadable file)

Authentication:

1.Current version runs in an open environment.

2.No authentication implemented.

3.Future enhancement: Add login system and role-based access.

User Interface:

1.Minimalist chatbot interface with Gradio.

2.Includes: Text input field Action buttons (Send, Clear)

3.Quick Question buttons (RTI, Aadhaar, Voter ID, Railway)

4.Gradient-based background for improved visual appeal.

Testing:

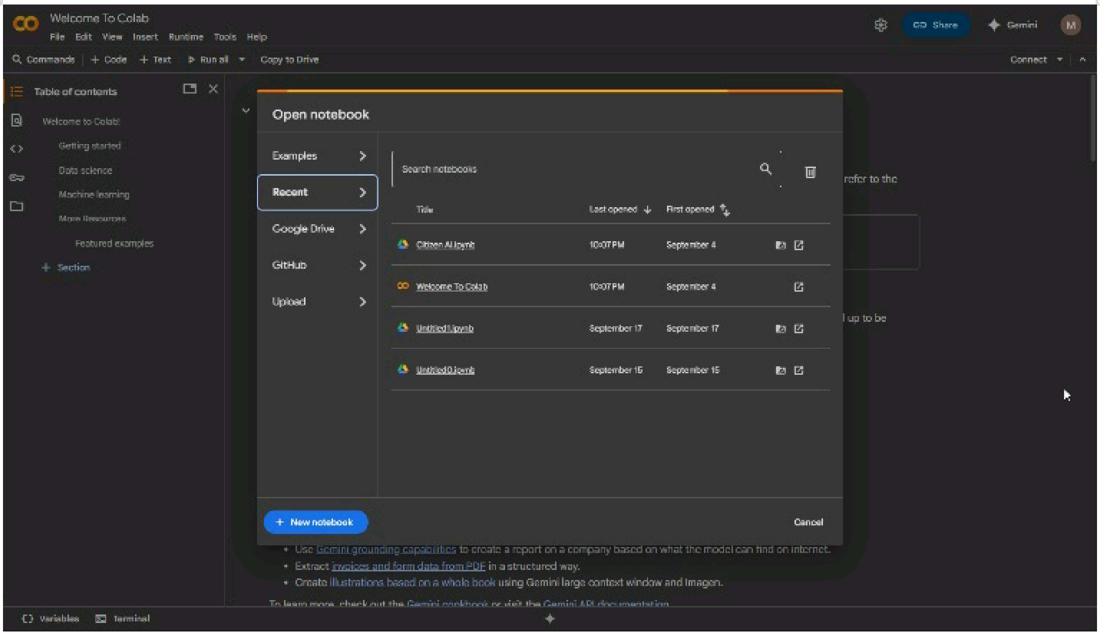
Unit Testing:
Validate input handling and model response.

Manual Testing: Ask queries on government services and verify responses.

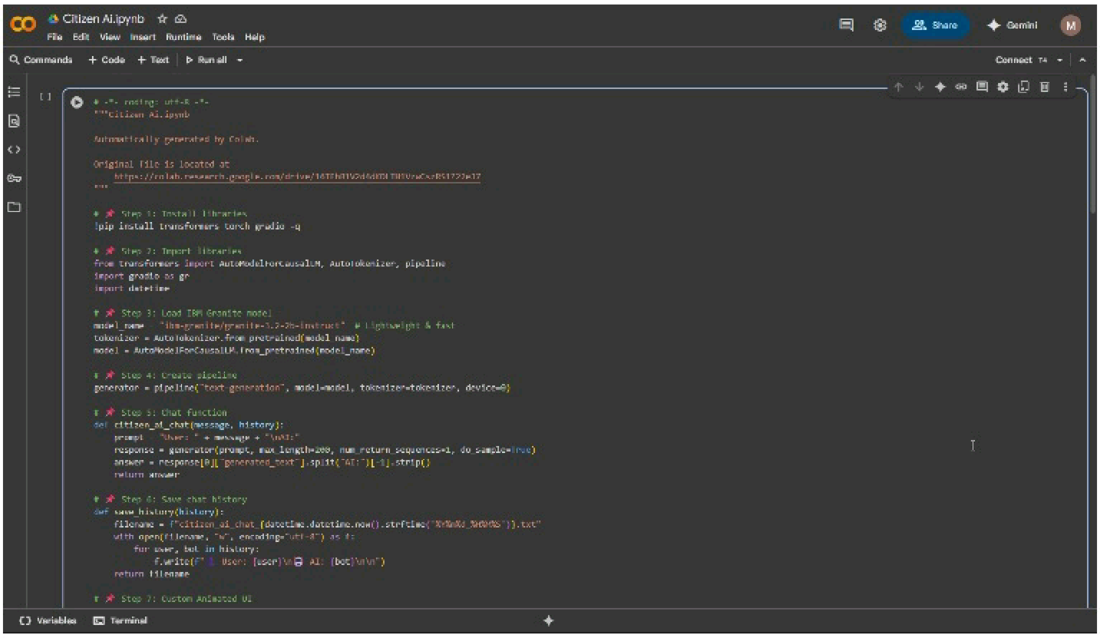
Edge Case Handling: Empty input, repeated queries, long text.

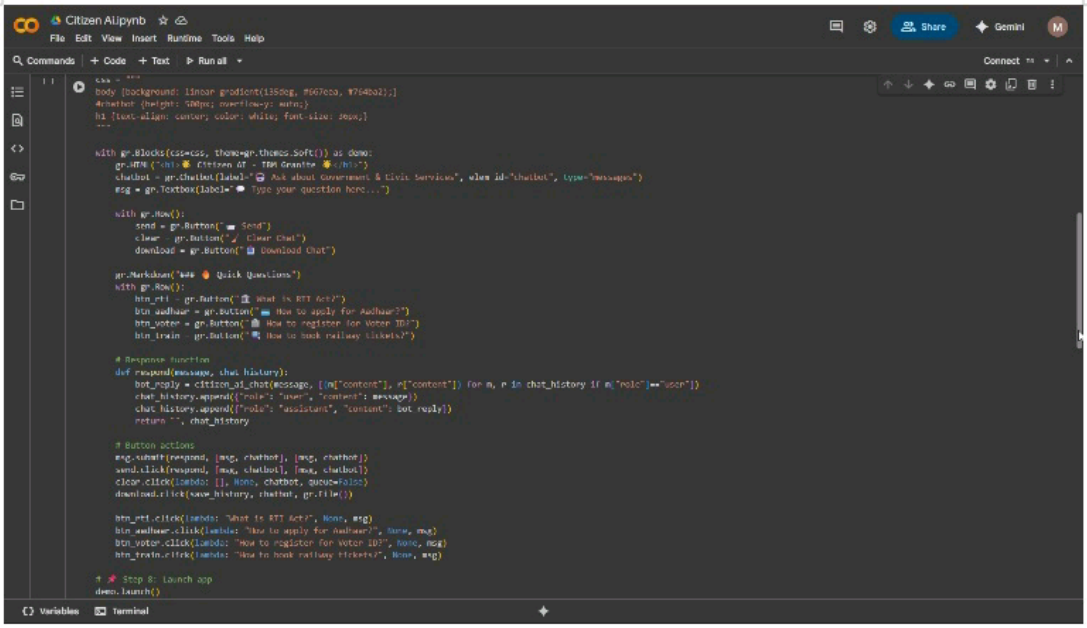
Screenshots:

TOOLS:

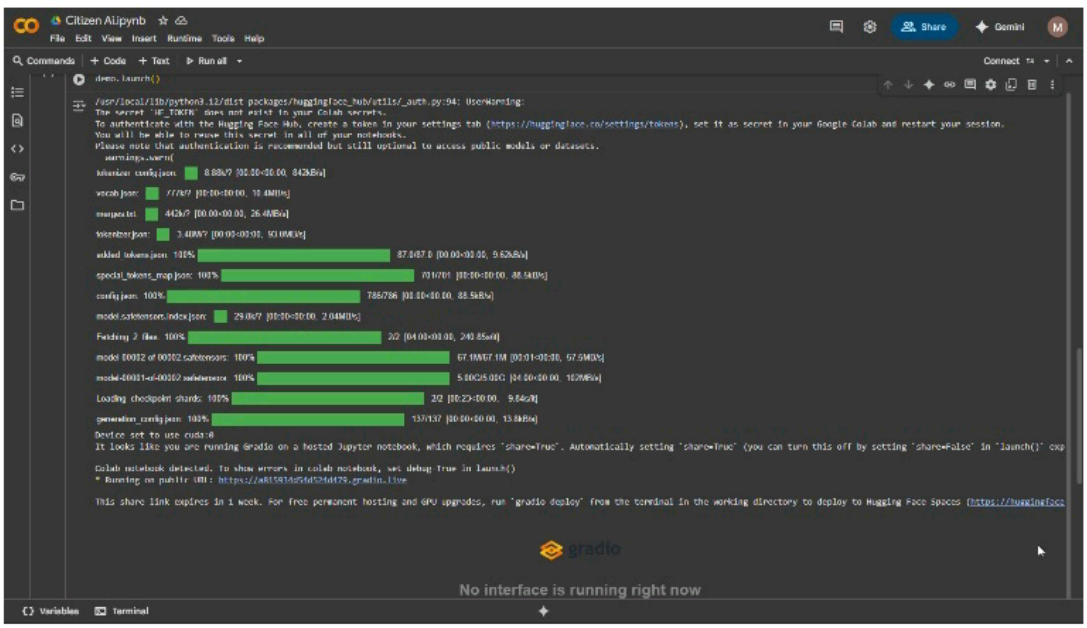

Welcome To Colab
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all Copy to Drive
Share
Gemini
Connect
Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More Resources
Featured examples
Section
Open notebook
Examples
Recent
Google Drive
GitHub
Upload
Search notebooks
Title
Last opened
First opened
Citizen AI.ipynb
10:07 PM
September 4
Welcome To Colab
10:07 PM
September 4
Untitled1.ipynb
September 17
September 17
Untitled0.ipynb
September 15
September 15
New notebook
Cancel
Variables
Terminal

CODING:


Citizen AI.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
Connect
"""Citizen AI.ipynb
Automatically generated by Colab.
Original file is located at
Step 1: Install libraries
!pip install transformers torch gradio -q
# Step 2: Import libraries
from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline
import gradio as gr
import datetime
# Step 3: Load IBM Granite model
model_name = "ibm-granite/granite-
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)
# Step 4: Create pipeline
generator = pipeline("text-generation", model=model, tokenizer=tokenizer, device=0)
# Step 5: Chat function
def citizen_ai_chat(message, history):
prompt = "User: " + message + "\nAI:"
response = generator(prompt, max_length=200, num_return_sequences=1, do_sample=True)
answer = response[0]["generated_text"].split("AI:")[-1].strip()
return answer
# Step 6: Save chat history
def save_history(history):
filename = f"citizen_ai_chat_{datetime.datetime.now().strftime(
with open(filename, "w", encoding="utf-8") as f:
for user, bot in history:
f.write(f"
return filename
Step 7: Custom Animated UI
Variables
Terminal

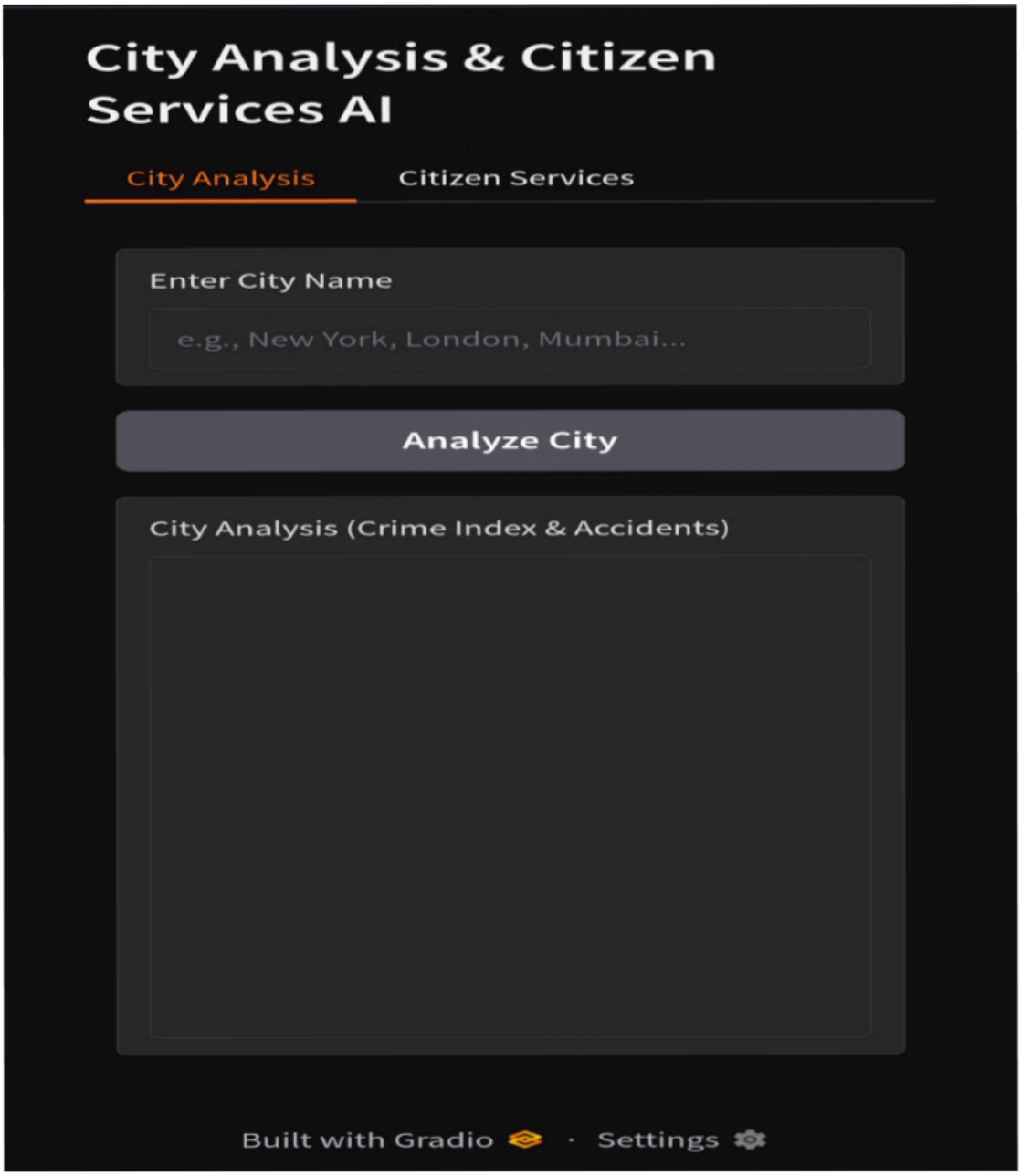

Citizen AI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect
Variables
Terminal

OUTPUT LINK:


Citizen AI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect
demo.launch()
No interface is running right now
Variables
Terminal

FINAL OUTPUT:

Screenshots of Coding, Output Link, and Final Output:
[Insert Screenshot 1: Coding Screenshot][Insert Screenshot 2: Output Link Screenshot][Insert Screenshot 3: Final Output (Gradio UI)]



Known Issues:

1.Large queries may exceed model's maximum token limit.

2.Some responses may be generic due to limited domain training.

3.Free Colab session may disconnect, stopping the app.

Future Enhancements:

1.Deploy permanently on Hugging Face Spaces or Streamlit Cloud.

2.Add voice input and voice output (speech-to-text and text-to-speech).

3.Add dashboard for sentiment analysis of citizen feedback. - Enable multilingual support (e.g., Tamil, Telugu, Hindi, etc.).