

# **Citizen AI: Intelligent Citizen Engagement Platform**

## **Project Documentation**

### **Introduction**

**Project title :** Citizen AI: Intelligent Citizen Engagement Platform

Team Member: Nirnajan.M

Team Member: Balamurugan.M

Team Member: surendiran.T

Team Member: Ragu.p

### **Project Overview**

#### **Purpose :**

Project Description:

Citizen AI is an intelligent citizen engagement platform designed to revolutionize how governments interact with the public. Leveraging Flask, IBM Granite models, and IBM Watson, Citizen AI provides real-time, AI-driven responses to citizen inquiries regarding government services, policies, and civic issues. The platform integrates natural language processing (NLP) and sentiment analysis to assess public sentiment, track emerging issues, and generate actionable insights for government agencies. A dynamic analytics dashboard offers real-time visualizations of citizen feedback, helping policymakers enhance service delivery and transparency. By automating routine interactions and enabling data-driven governance, Citizen AI improves citizen satisfaction, government efficiency, and public trust in digital governance.

#### **Features:**

Citizen AI is an intelligent citizen engagement platform that offers several key

features to enhance citizen-government interactions. Some of the notable features include <sup>1 2</sup>:

**AI-Powered Chat Assistant\***: Provides instant, accurate responses to citizen inquiries, understanding complex queries in everyday language and maintaining conversation context for better responses.

**Real-Time Analytics Dashboard\***: Offers insights into citizen engagement patterns, sentiment trends, and service performance through interactive dashboards, including metrics such as total conversations, average session duration, and user return rate.

**Sentiment Analysis**: Analyzes citizen feedback and classifies sentiment as Positive, Neutral, or Negative, enabling governments to track sentiment trends and make data-driven decisions.

**Concern Reporting**: Allows citizens to report issues and track their resolution, with features like automated categorization, priority assignment, and status tracking.

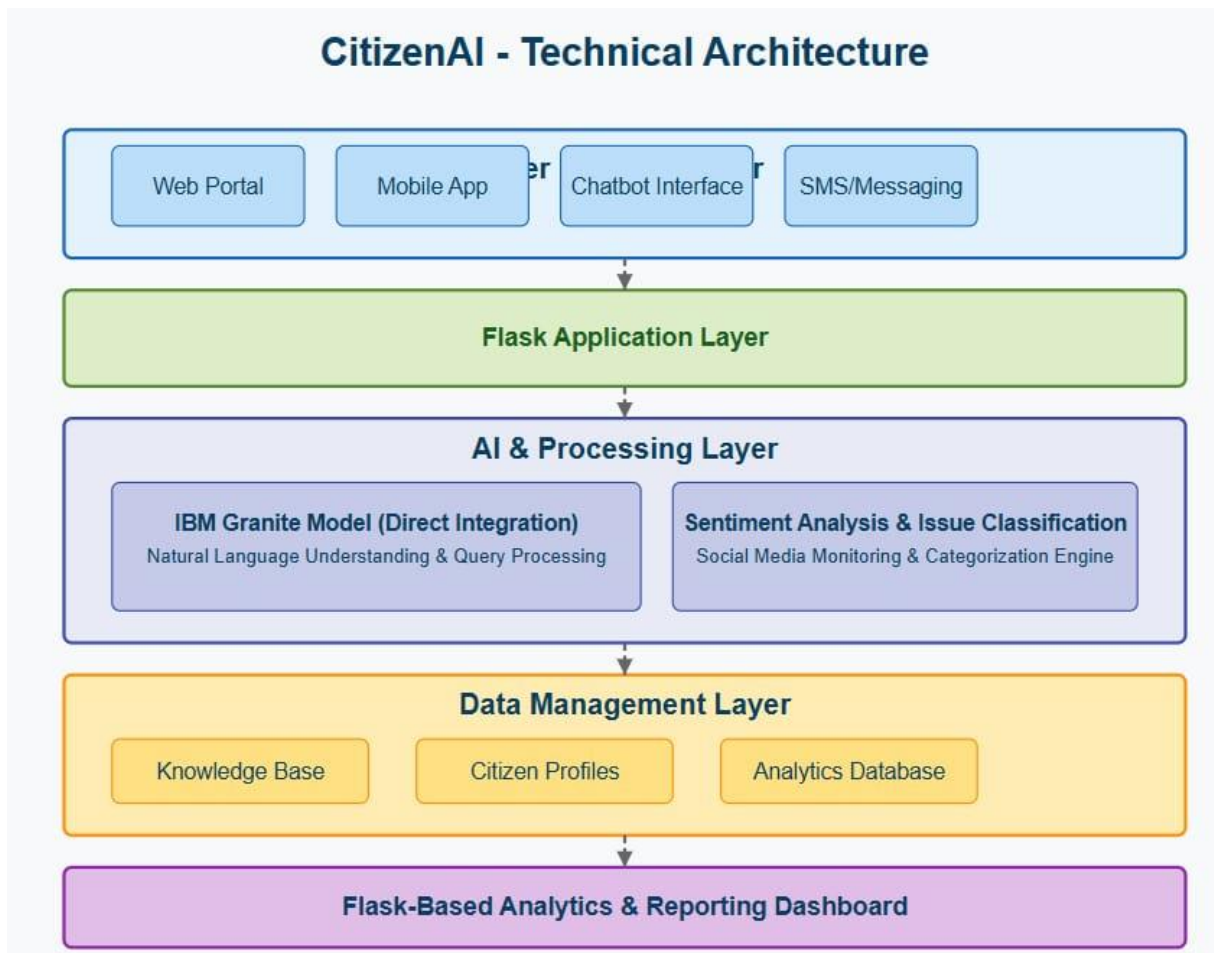
**Accessibility Features**: Designed with accessibility in mind, Citizen AI ensures compliance with WCAG 2.1 AA standards, screen reader compatibility, keyboard navigation, and high contrast display support.

**Security Measures**: Includes enterprise-grade security features like session-based authentication, role-based access control, data encryption, and audit logging to ensure data protection and user privacy.

**Integration Capabilities**: Provides RESTful APIs for integration with existing systems, including authentication, chat, analytics, and concern management APIs.

**Customization Options**: Offers theme configuration, logo integration, layout options, and font customization to enable governments to tailor the platform to their brand and needs.

## **Architecture**



## Pre-requisites

**Python:** You need a working Python 3.7+ environment installed on your system.

**Flask:** The Flask web framework is required to run the web application.

**PyTorch:** As you are using a deep learning model, you need PyTorch installed. If you plan to use your GPU for faster inference, ensure you install the version of PyTorch with CUDA support that matches your GPU and CUDA toolkit version.

**Hugging Face Libraries:** The transformers, accelerate, and bitsandbytes libraries are essential for loading and utilizing the IBM Granite model, especially with quantization.

Sufficient Hardware: Running a large language model like IBM Granite

3.3B requires significant resources. You will need:

RAM: A substantial amount of RAM (typically 16GB or more is recommended, even with quantization).

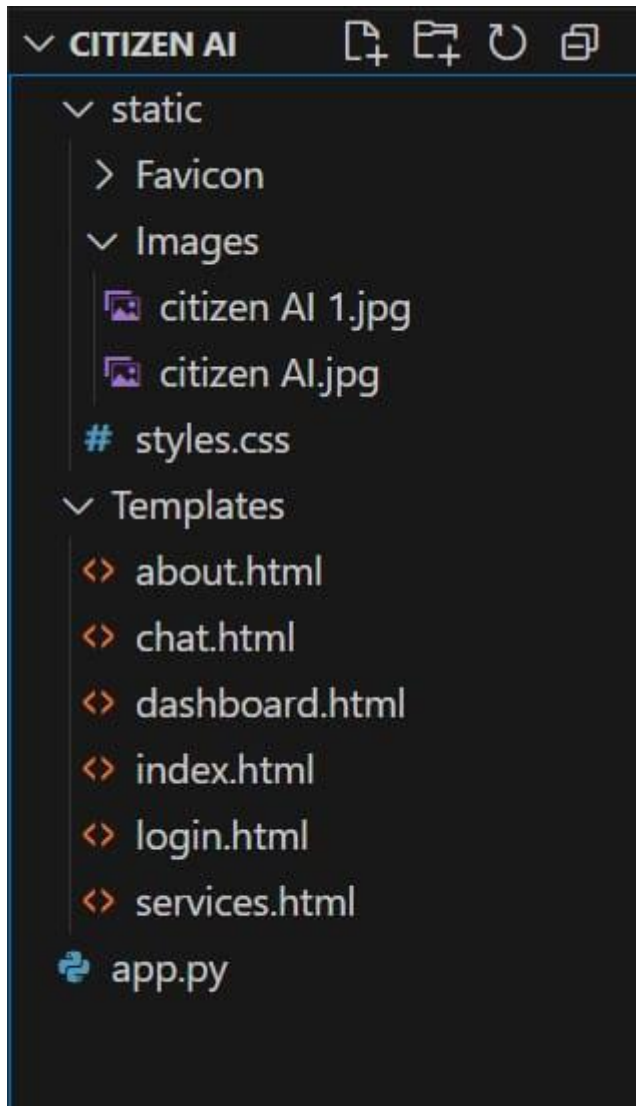
GPU (Recommended): A compatible NVIDIA GPU with sufficient VRAM (8GB or more is highly recommended, especially for the 8B model, even with 4-bit quantization) and correctly installed CUDA drivers for reasonable inference speed. Running solely on a CPU will be very slow.

Internet Connection: The first time you run the application, the IBM Granite model files will be downloaded from the Hugging Face Hub. You need an active internet connection for this.

Project Structure: The project files should be organized correctly with `app.py`, a `templates` folder containing your HTML files (`index.html`, `about.html`, `services.html`, `chat.html`, `dashboard.html`, `login.html`), and a `static` folder containing your CSS (`styles.css`) and `image/favicon` subfolders (e.g., `static/Images`, `static/Favicon`).

Set Up Application Structure:

Create the basic project directory structure: `app.py`, `templates/` (for HTML files), and `static/` (with `css/`, `Images/`, `Favicon/` subfolders).



## Functional Testing and Verify

Index page:

This is the landing page of the CitizenAI web application, designed for civic engagement through AI. Here's a breakdown:

Header Section:

Shows the main title: "Welcome to CitizenAI".

Contains navigation links: About, Services, Chat, Dashboard, and Login.

Left Panel (Intro Section):

Headline: "Empowering Citizens Through AI".

Describes CitizenAI as an intelligent assistant helping citizens engage with government services, provide feedback, and communicate more effectively.

Includes a prominent "Get Started" button, likely redirecting to user interaction or signup.

Right Panel (Visual):

Features a digital, futuristic background with a human head silhouette and glowing circuit lines.

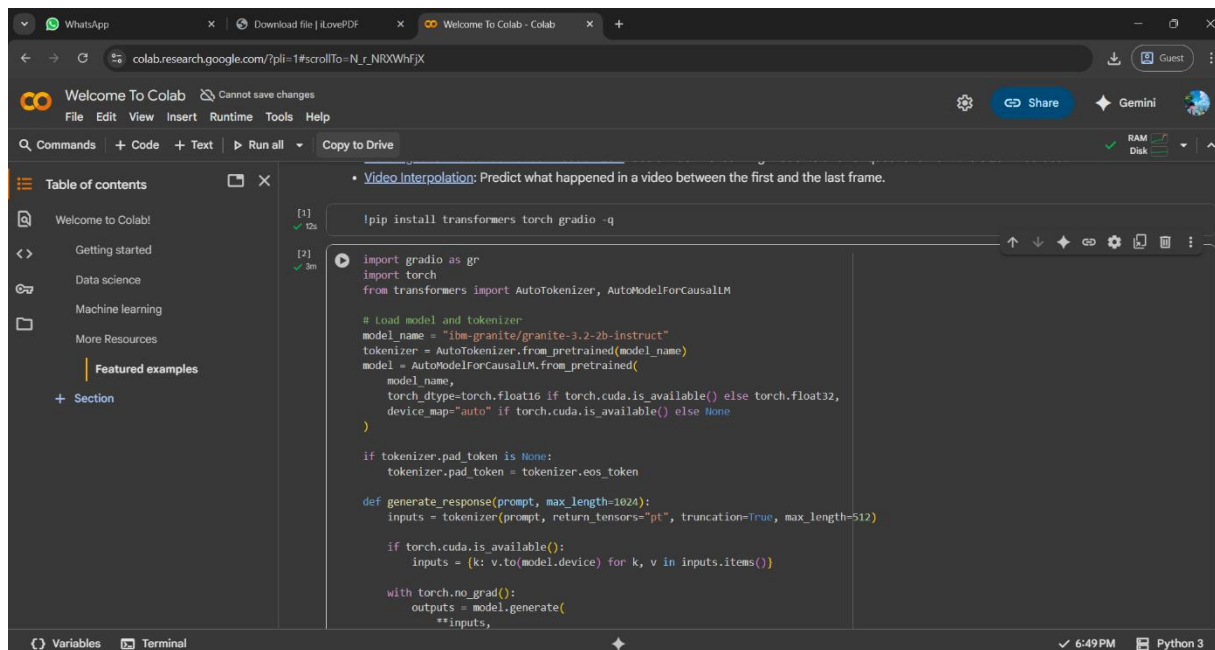
Overlaid with code and AI-related text to emphasize the tech-driven nature of the platform.

Purpose:

The page promotes an AI-powered civic platform aimed at building a smarter, responsive government-citizen relationship.

It encourages users to begin interacting with the system by clicking Get Started.

## Screenshoot



The screenshot shows a Google Colab notebook with the following code in the main cell:

```
[1] ✓ 12s
[2] ✓ 3m

!pip install transformers torch gradio -q

import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

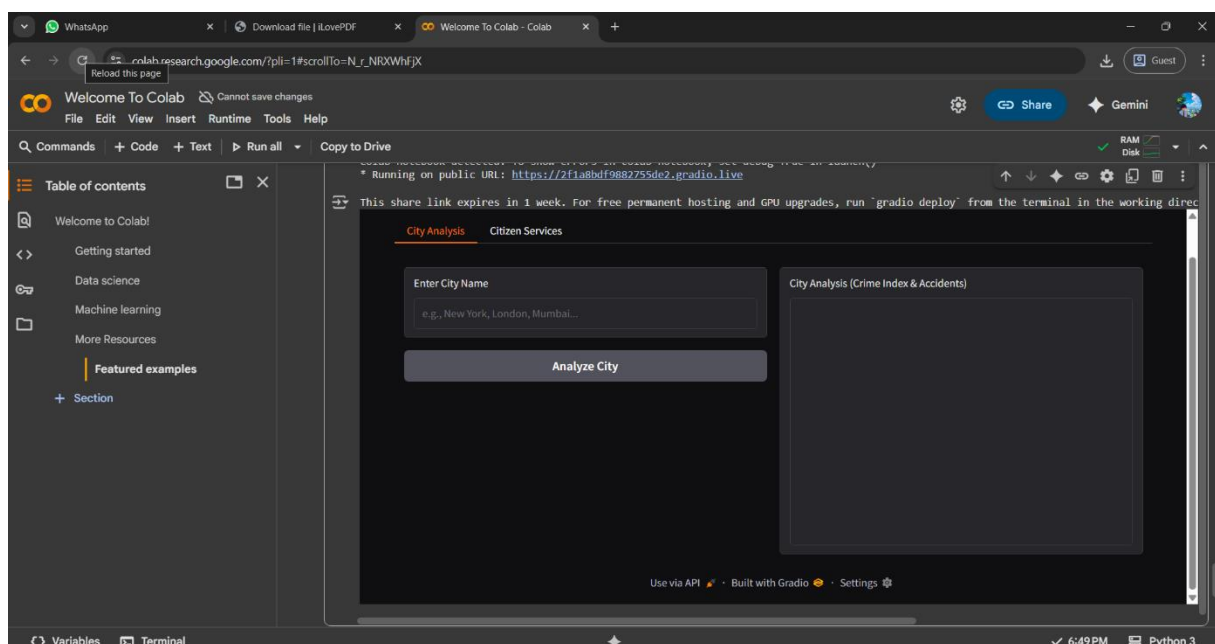
# load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

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,
```



## Conclusion

Your AI-powered CitizenAI platform is designed to enhance interaction, accessibility, and transparency between citizens and government services. By integrating an AI chat assistant, sentiment analysis, concern reporting, and dashboard insights, the platform empowers users to easily access information, provide feedback, and report issues. With a Flask backend and an interactive HTML/CSS frontend, powered by the IBM Granite AI model, your project ensures a user-friendly experience while providing smart and responsive civic engagement tools. This innovative solution simplifies communication and fosters trust, making civic participation more convenient and efficient for all.