# Citizen AI Project Documentation

1. Introduction

Project Title: Citizen AI – Intelligent Citizen Engagement Platform
Team Id:NM2025TMID00797
Team Members:

- Priya Dharshini P
- SelvaPriya S
- Pooja V

2. Project Overview

Purpose
Citizen AI is an advanced citizen engagement platform designed to transform the way governments connect with the public. Built using Flask, IBM Granite LLM, and IBM Watson, it delivers real-time, AI-powered responses to citizen questions about government services, policies, and civic matters.
By incorporating natural language processing (NLP) and sentiment analysis, the platform evaluates public sentiment, identifies emerging concerns, and provides actionable insights for government agencies.
Citizen AI enhances citizen satisfaction, streamlines government operations, and fosters greater public trust in digital governance by automating routine communications and supporting data-driven policy making.
Features

- Conversational Chat Assistant
  - Highlight: Real-time interaction with citizens
  - Description: Delivers immediate, human-like AI responses to inquiries about government services, policies, and civic concerns.
- Citizen Sentiment Analysis
  - Highlight: Monitoring public sentiment
  - Description: Analyzes citizen feedback and categorizes it as Positive, Neutral, or Negative to identify satisfaction levels and areas of concern.
- Dynamic Analytics Dashboard
  - Highlight: Insight-driven decision-making
  - Description: Presents visual data on trends, public sentiment, and reported issues to support informed policymaking.

- Concern Reporting
    - Highlight: Transparent issue resolution
    - Description: Enables citizens to submit complaints or concerns, which are tracked for follow-up and resolution.

Use Case Scenarios

1. Interactive AI Chat Assistant
   Citizens interact through a chat interface, receiving prompt and accurate responses from the AI regarding public services, government policies, and civic matters.
2. Sentiment Analysis of Citizen Feedback
   The system processes citizen input, evaluates sentiment (Positive, Neutral, or Negative), and compiles the data to support informed decision-making.
3. Real-Time Analytics Dashboard
   Government officials access a live dashboard that displays trends in public sentiment, service satisfaction, and reported issues, allowing for timely and data-driven responses.

3. Architecture

- Frontend:
  Built with HTML and CSS, the interface includes templates for the homepage, about section, services, chat interface, analytics dashboard, and user login.
- Backend (Flask):
  Responsible for handling application routes, user authentication, and backend data logic.
- Large Language Model (IBM Granite):
  Integrates advanced AI capabilities for natural conversation, text generation, and sentiment detection.
- Data Management:
  Currently uses in-memory storage to manage chat logs, sentiment data, and reported concerns, with future plans to implement a database solution.
- Data Visualization:
  Includes a dynamic dashboard that presents real-time charts and analytics for tracking sentiment patterns and issue reports.

4. Setup Instructions

Prerequisites

- Python 3.7+
- Flask
- PyTorch (with CUDA for GPU acceleration)
- Hugging Face libraries: transformers, accelerate, bitsandbytes
- Hardware:
  - 16GB+ RAM
  - NVIDIA GPU with 8GB+ VRAM recommended
- Internet connection (for first-time model download)

Installation Process
1. Clone repository and set up project structure (app.py, templates/, static/).
2. Create and activate a virtual environment:
3. python -m venv env
4. source env/bin/activate # Linux/Mac
5. env\Scripts\activate # Windows
6. pip install -r requirements.txt
7. Install Flask, PyTorch, and Hugging Face dependencies.
8. Configure IBM Granite model path (ibm-granite/granite-3.3-8b-instruct).
9. Run the Flask backend with:
10. python app.py
5. Folder Structure

app.py – Main Flask application
templates/ – HTML templates (index, about, services, chat, dashboard, login)
static/ – CSS, Images, Favicon
requirements.txt – Python dependencies
6. Running the Application

- Launch Flask backend (python app.py).
- Open browser and navigate to http://localhost:5000.
- Use navigation menu for:

  Chat: Interact with the AI assistant.

  Feedback: Submit text for sentiment analysis.

  Dashboard: View real-time citizen insights.

  Login: Authenticate to access protected content.

7. API Endpoints

- POST /ask – Accepts citizen inquiries and returns AI-generated responses.
- POST /feedback – Submits user feedback for sentiment analysis.
- POST /concern – Allows users to report issues or concerns.
- GET /dashboard – Retrieves aggregated sentiment data and reported issues.
- POST /login – Handles user login and authentication.
- POST /logout – Ends the current user session.

8. Authentication

- Supports user login with session-based authentication.
- Upcoming Feature: Role-based access control for different user types (citizens, administrators, government officials).

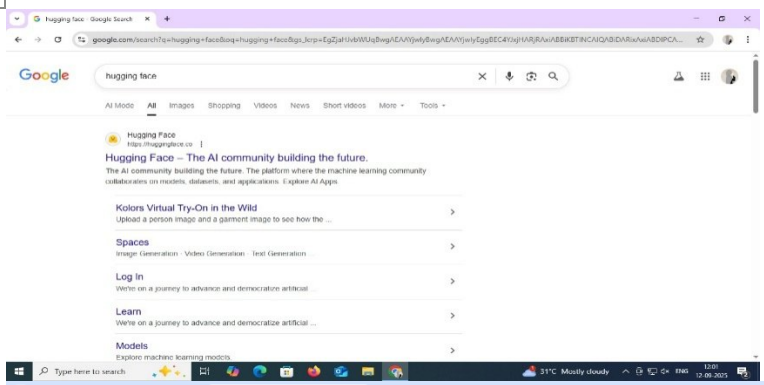
9. User Interface

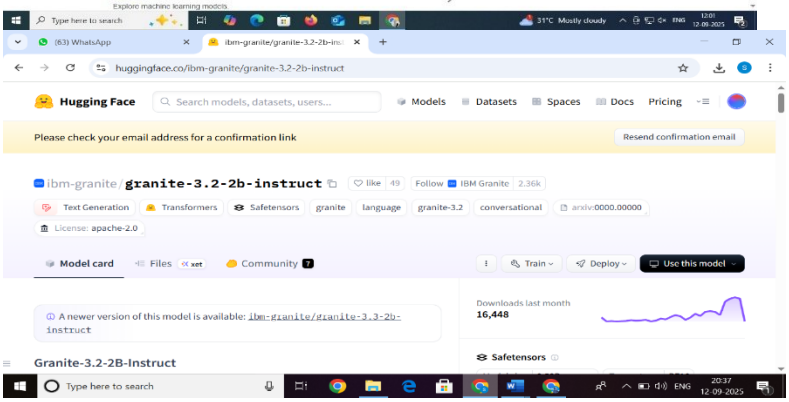
- Index Page: Introductory landing page with a welcome message and "Get Started" call to action.
- Login Page: Secure login form for user authentication.
- About Page: Describes the platform's mission, key features, and benefits to users.
- Chat Page: Interactive interface for AI-powered conversations with citizens.
- Dashboard: Displays sentiment breakdown (positive, neutral, negative) and recent citizen-reported issues in real time.

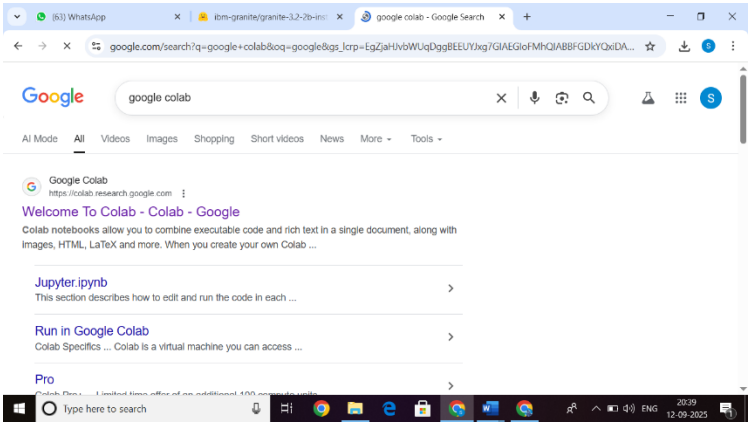
10. Testing Strategy

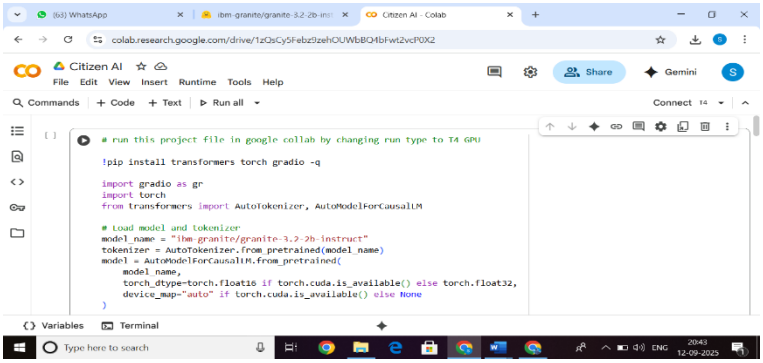
- Unit Testing: Covers Flask routes and core AI functionalities.
- Integration Testing: Validates complete workflows across chat, feedback submission, and dashboard updates.
- Manual Testing: Includes form submissions, sentiment classification accuracy, and issue reporting validation.
- Edge Case Handling: Tests for scenarios like empty input fields, invalid login attempts, and improperly formatted data.

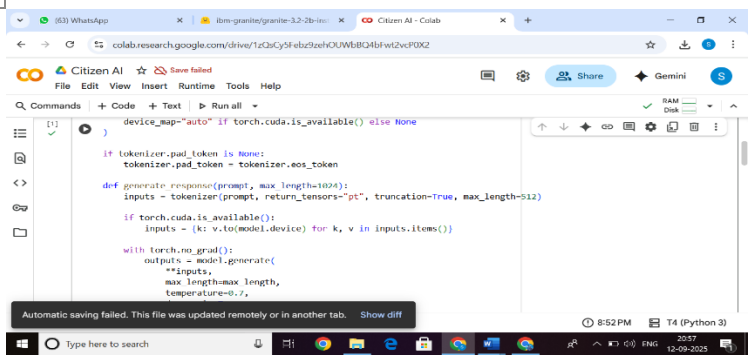
11. Screenshots


hugging face
Hugging Face – The AI community building the future.
The AI community building the future. The platform where the machine learning community collaborates on models, datasets, and applications. Explore AI Apps
Kolors Virtual Try-On in the Wild
Upload a person image and a garment image to see how the ...
Spaces
Image Generation · Video Generation · Text Generation
Log in
We're on a journey to advance and democratize artificial ...
Learn
We're on a journey to advance and democratize artificial ...
Models
Explore machine learning models.


huggingface.co/ibm-granite/granite-3.2-2b-instruct
Hugging Face
Search models, datasets, users...
Models
Datasets
Spaces
Docs
Pricing
Please check your email address for a confirmation link
Resend confirmation email
ibm-granite/granite-3.2-2b-instruct
Text Generation
Transformers
Safetensors
granite
language
granite-3.2
conversational
License: apache-2.0
Model card
Files
Community
Train
Deploy
Use this model
A newer version of this model is available: ibm-granite/granite-3.3-2b-instruct
Downloads last month
16,448
Safetensors
Granite-3.2-2B-Instruct


google.com/search?q=google+colab&oq=google&gs_lcrp=EgZjaHJvbWUqDggBEEUYJxg7GIAEGIoFMhQIABBFGDkYQxiDA...
google colab
AI Mode
All
Videos
Images
Shopping
Short videos
News
More
Tools
Google Colab
https://colab.research.google.com
Welcome To Colab - Colab - Google
Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab ...
Jupyter.ipynb
This section describes how to edit and run the code in each ...
Run in Google Colab
Colab Specifics ... Colab is a virtual machine you can access ...
Pro


colab.research.google.com/drive/1zQsCy5Febz9zehOUWbBQ4bFwt2vcP0X2
Citizen AI
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
Code
Text
Run all
Connect
# run this project file in google collab by changing run type to T4 GPU
!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
)
Variables
Terminal




```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 160kB/s]
vocab.json: 777k/? [00:00<00:00, 9.07MB/s]
merges.txt: 442k/? [00:00<00:00, 6.65MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 10.6MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 2.67kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 27.7kB/s]
config.json: 100% 786/786 [00:00<00:00, 21.2kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.19MB/s]
Fetching 2 files: 0% 0/2 [00:00<?, ?it/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 69.3MB/s]
model-00001-of-00002.safetensors: 76% 3.79G/5.00G [01:15<00:24, 49.2MB/s]
```

## 12. Current Limitations

- Data is temporarily stored in memory (no long-term persistence).
- Performance is slow due to CPU-only execution.
- Scalability is restricted until database support is added.

## 13. Planned Enhancements

- Integration with a database for reliable data storage.
- Support for multiple languages.
- Fully responsive, mobile-friendly design.
- Adoption of advanced NLP models for improved policy summarization.
- Connectivity with social media and public forums for sentiment analysis.