Citizen AI Project Documentation

1. Introduction

Project Title: Citizen AI – Intelligent Citizen Engagement Platform
Team Id:NM2025TMID00806SS
Team Members:

- Subapriya C
- Safrin Banu M
- Thoufika Nilofer O A

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

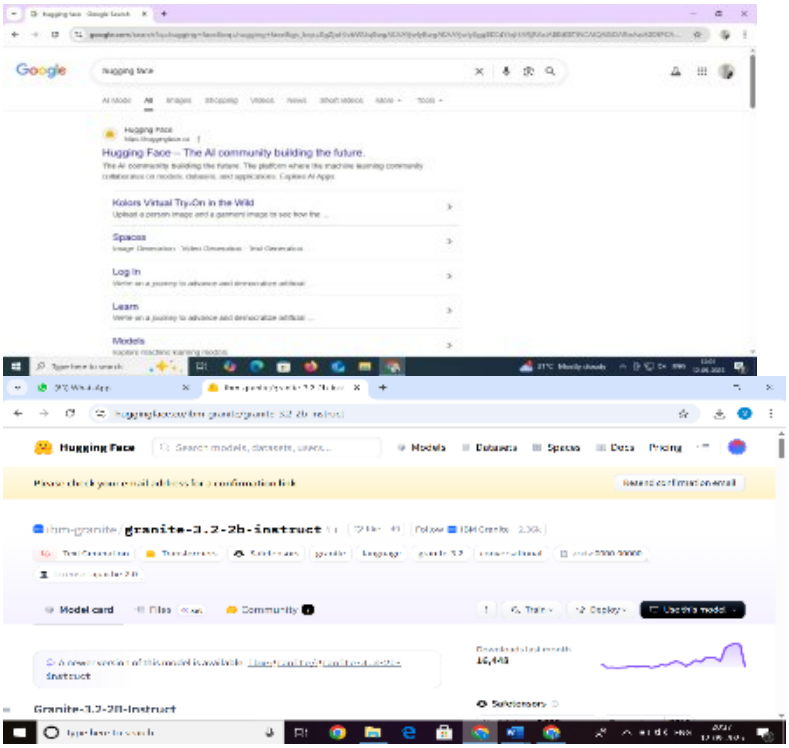
- Index Page: Introductory landing page with a welcome message and "Get Started" call to action.
- Login Page: Secure login form for user authentication.
- About Page: Describes the platform's mission, key features, and benefits to users.
- Chat Page: Interactive interface for AI-powered conversations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

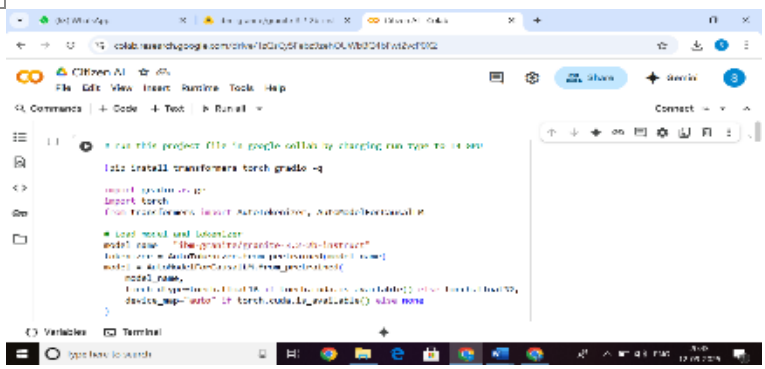
10. Testing Strategy

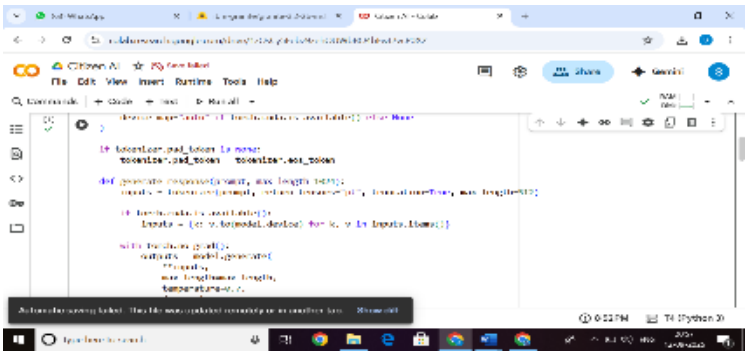
- Unit Testing: Covers Flask routes and core AI functionalities.

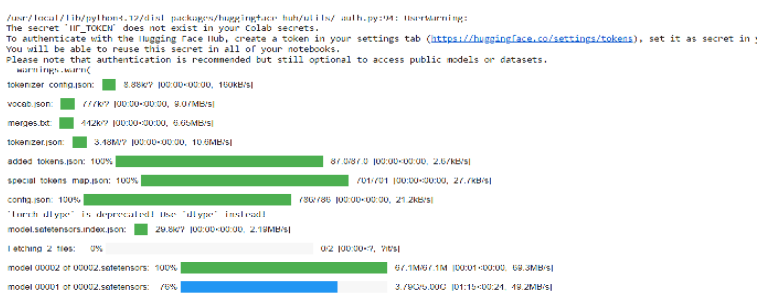
- Integration Testing: Validates complete workflows across chat, feedback submission, and dashboard updates.
- Manual Testing: Includes form submissions, sentiment classification accuracy, and issue reporting validation.
- Edge Case Handling: Tests for scenarios like empty input fields, invalid login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

11. Screenshots









---

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.