# Project Documentation

**INTELLIGENT CITIZEN ENGAGEMENT PLATFORM**

## 1. Introduction

• Project Title Intelligent Citizen Engagement Platform

**Team Leader : Saranya J**

## •Team Member: RajaVarshini V B

## • Team Member: Reshmi D

## • Team Member: Sakthi Sevani B

## • Team Member: Sandhiya B

## 2. Project Overview

• Purpose:  
Citizen AI is an AI-powered citizen engagement platform that leverages Flask, IBM Granite models, and IBM Watson. It provides real-time conversational responses, analyzes citizen feedback through sentiment analysis, enables concern reporting, and offers a dynamic dashboard for policymakers. The platform improves accessibility, transparency, and trust in government services.

### Features:

* Real-Time Conversational AI Assistant – Natural language responses to citizen queries.
* Citizen Sentiment Analysis – Classifies feedback as Positive, Neutral, or Negative.
* Dynamic Dashboard – Provides real-time insights and visualizations.
* Concern Reporting – Citizens can log issues or complaints.
* Data-Driven Governance – Aggregates and analyzes citizen input for decision-making.

## 3. Architecture

The Citizen AI system is designed with a layered architecture:  
- User Interaction Layer: Web portal, chatbot, mobile interface, SMS support.  
- Application Layer: Flask backend routes handling chat, feedback, concerns, and dashboard requests.  
- AI & Processing Layer: IBM Granite model for NLU and generation; sentiment analysis logic.  
- Data Management Layer: In-memory storage for history, feedback, and concerns; extendable to databases.  
- Dashboard Layer: Real-time analytics visualized through Flask templates and charts.

## 4. Setup Instructions

**Prerequisites:**

* Python 3.7+ environment installed.
* Flask web framework.
* PyTorch (with CUDA if GPU available).
* Hugging Face libraries: transformers, accelerate, bitsandbytes.
* 16GB RAM recommended; NVIDIA GPU with 8GB+ VRAM for larger Granite models.
* Internet connection to download Granite models from Hugging Face.
* Organized project files with app.py, templates/, and static/ directories.

**Installation Process:**  
1.Create virtual environment: python -m venv env  
2. Activate environment: source env/bin/activate (Linux/Mac) or env\Scripts\activate (Windows)  
3. Install dependencies: pip install -r requirements.txt  
4. Run the Flask app: python app.py

## 5. Folder Structure

citizen/  
│── app.py – Main Flask application  
│── templates/ – HTML templates (index.html, about.html, services.html, chat.html, dashboard.html, login.html)  
│── static/ – CSS, Images, Favicon  
│── sentiment.py – Sentiment analysis logic  
│── granite\_llm.py – IBM Granite model integration  
│── dashboard.py – Visualization and analytics logic

## 6. Running the Application

* Start the Flask server: python app.py
* Open browser and go to <http://localhost:5000>
* Use the Chat Assistant to ask questions
* Submit feedback and concerns via forms
* Access Dashboard for sentiment distribution and reported issues

## 7. API Documentation

* POST /ask – Processes citizen queries and returns AI-generated response.
* POST /feedback – Submits feedback for sentiment analysis.
* POST /concern – Submits citizen concerns/issues.
* GET /dashboard-data – Returns aggregated sentiment and issue data.

## 8. Authentication

Authentication is implemented for login/logout and session management. In future deployments, it can be extended to include JWT tokens, OAuth2, and role-based access for citizens and officials.

## 9. User Interface

## Index Page

Landing page with navigation, welcome message, introduction text, and a Get Started button leading to login.

## Login Page

Secure login page with username/password fields and login validation.

## About Page

Explains CitizenAI’s mission, key features, and impact. Includes navigation and project description.

## Chat Page

Citizen Chat Assistant with input form for queries, sentiment analysis form, and concern reporting form. Displays AI responses, sentiment results, and confirmation messages.

## Dashboard Page

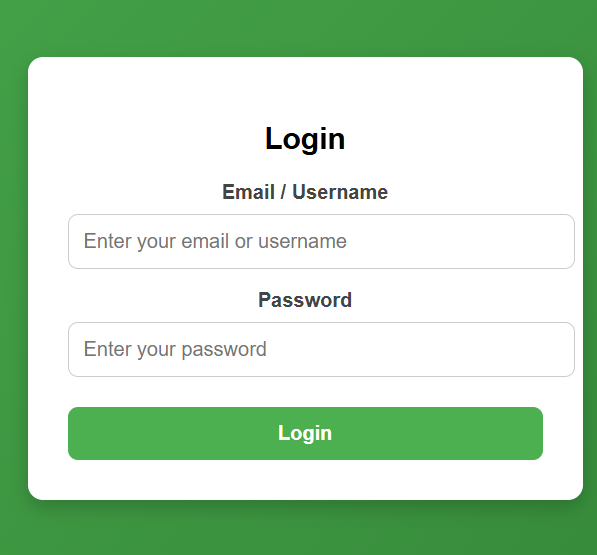
Real-time insights page showing sentiment counts (Positive, Neutral, Negative) and recent reported concerns.

## 10. Testing

- Functional testing was performed on all key pages:  
- Index Page: Verified navigation and Get Started button.  
- Login Page: Tested authentication, error messages, and session management.  
- About Page: Confirmed mission, features, and navigation links.  
- Chat Page: Verified AI responses, sentiment analysis, and concern reporting functionality.  
- Dashboard Page: Checked sentiment visualization and concern list updates.

11. Project Screenshots:





## 

## 

## 

## 12. Issues Faced in the Project

* Granite model inference is slow without GPU acceleration.
* Sentiment analysis may misclassify sarcasm or complex feedback.
* In-memory storage limits persistence until database integration is added.

## 13. Future Enhancements

* Add multilingual support for wider accessibility.
* Enable predictive analytics for trend forecasting.
* Develop a mobile app version for increased adoption.
* Deploy on cloud platforms (IBM Cloud, AWS, Azure) for scalability.
* Integrate role-based authentication with OAuth2.

14.conlusion

The Intelligent Citizen Engagement Platform (**CitizenAI**) showcases how AI can improve civic engagement by providing real-time responses, sentiment analysis, concern reporting, and data insights. The combination of Flask backend, HTML/CSS frontend, and IBM Granite AI creates a user-friendly, responsive platform that strengthens trust and transparency between citizens and government services.