Citizen AI — Project Documentation

1. Introduction

- **Team ID**: NM2025TMID04760
- **Team Size**: 4
- **Team Leader**: **PACHAMMAL V**
- **Team Members**: SUBASRI A, VISHNUPRIYA M, RAMYA K

Citizen AI is a generative AI–powered platform designed to act as a bridge between citizens and government institutions. It supports open communication, policy understanding, and feedback collection in a way that is accessible to every citizen regardless of technical expertise.

2. Project Overview

Purpose
Citizen AI aims to create a transparent and intelligent citizen engagement platform. By leveraging IBM Granite LLMs, the system can respond to natural language queries, summarize lengthy policy documents, and analyze citizen sentiment. This empowers governments to make informed decisions while improving trust and communication with the public.

Key Features (Detailed)

1. **Conversational Interface**
   Citizens can ask natural language questions about government services. The system responds instantly with context-aware answers, removing the need to browse complex websites.

2. **Policy Summarization**
   Long government policies and circulars are converted into short, plain-language summaries. This allows citizens to quickly understand rules, schemes, and updates.

3. **Sentiment Analysis**
   Every citizen query is analyzed to determine whether feedback is positive, negative, or neutral. This provides officials with early insight into growing issues or dissatisfaction.

4. **Citizen Feedback Loop**
   Feedback collected from queries is logged and analyzed. Officials can then identify what services are working well and where improvements are needed.

5. **Dashboard & Logging**
   All queries, responses, and sentiment scores are logged into CSV files. These logs can later be visualized in a dashboard to show patterns, frequent issues, and public mood.

6. **Gradio Web UI**
   A simple web-based interface (built with Gradio) makes the platform easy to use for both citizens and officials. No technical background is required.

3. Architecture

Frontend (Gradio)
Provides a lightweight, browser-accessible UI. Citizens can type queries, view answers, and officials can check logs.

Backend (Python + Transformers)
Handles model execution, manages citizen inputs, performs sentiment analysis, and updates conversation logs.

LLM Integration (IBM Granite)
IBM Granite-3.2-2b-instruct powers conversation and summarization. It is optimized for civic engagement use cases.

Data Logging
All interactions are stored in a structured CSV log file. This ensures queries and responses are available for later review.

Deployment
Optimized for Google Colab with GPU runtime. Can also be deployed locally or on cloud servers.

4. Setup Instructions

Prerequisites

- Python 3.9 or above
- pip (Python package manager)
- Hugging Face account and API token (if model requires gated access)
- Internet access (Google Colab with GPU recommended)

Steps

1. Upload project files (`app.py`, `requirements.txt`) into Colab or your system.
2. Install dependencies:

   Copy code

   ```
   pip install -r requirements.txt
   ```

3. If required, set Hugging Face API token:

   Python Copy code

   ```python
   import os
   os.environ['HUGGINGFACEHUB_API_TOKEN
   '] = "hf_xxxxx"
   ```

4. Run the application:

   Copy code

   ```
   python app.py
   ```

5. Open the Gradio link and interact with the chatbot.

5. Folder Structure

- **app.py** – Main application script (chatbot + sentiment analysis + logging)
- **requirements.txt** – List of required dependencies
- **conversations.csv** – Stores citizen queries, responses, and sentiment analysis results
- **README.md** – Documentation and setup guide

6. Running the Application

- Run the app (`python app.py`) in Colab or terminal.
- Open the provided Gradio link in your browser.
- Citizens can ask questions in the chat box.
- Responses will appear instantly.
- Logs can be refreshed to show the latest citizen queries, responses, and sentiment scores.

7. API Documentation

(Planned for future expansion)

- **POST /chat** ′n Accepts a user query and returns AI-generated response.
- **GET /logs** ′n Retrieves the most recent conversation logs.
- **POST /feedback** ′n Collects and stores citizen feedback for later analytics.

8. Authentication

Currently, the demo runs in open access mode for testing.

For production:

- Token-based authentication (JWT / API keys).
- Role-based access (citizens, officials, administrators).
- Session tracking to maintain conversation history.

9. User Interface

The UI is simple and intuitive:

- Chat input box for citizen queries.
- Response box showing AI answers.
- Logs table with queries, responses, and sentiments.
- Future versions will include dashboards and visualizations for administrators.

10. Testing

Testing process included:

- **Unit Testing**: Verified input-output consistency of the chatbot.
- **API Testing**: Validated backend endpoints using sample queries.
- **Manual Testing**: Interacted with the chatbot in Colab to check responses.
- **Edge Case Testing**: Checked long queries, empty inputs, and invalid tokens.

11. Known Issues

- Large models may exceed free Colab GPU memory.
- Responses may occasionally include hallucinations.
- Logs are stored only in CSV format (no full database).
- Backend API endpoints are basic in current version.

12. Future Enhancements

- Full-featured analytics dashboard with charts and trend analysis.
- Database integration (PostgreSQL / Firebase) for scalable storage.
- Multilingual support for citizens across regions.
- Integration with IBM WatsonX for enterprise-scale deployment.
- Retrieval-Augmented Generation (RAG) for document-based question answering.