CITIZEN AI:INTELLIGENT CITIZEN ENGAGEMENT PLATFORM

1.Introduction

- Project title **:** Citizen AI:Intelligent Citizen Engagement Platform
- Team member : Sangeetha S
- Team member : Reethika K
- Team member : Reshma S
- Team member : Shalini M

2.project overview

• Purpose :

Intelligent Citizen Engagement Platform is developed to revolutionize the interaction between governments and citizens by leveraging artificial intelligence technologies. It serves as an intelligent communication hub where citizens can easily access public information, engage in discussions, provide feedback, and participate in decision-making processes through user-friendly digital channels. By incorporating AI-powered features like chatbots, natural language processing, and sentiment analysis, the platform enables timely and personalized responses to citizen inquiries, improving overall accessibility and ensuring that diverse community voices are heard. This increased engagement fosters transparency and builds trust between citizens and government bodies, helping to create a more open and accountable governance environment.

Moreover, the platform uses data-driven insights to help public agencies understand community needs and opinions more effectively. It automates routine administrative tasks such as application processing, complaint management, and service requests, which significantly enhances the efficiency of public service delivery.

With multilingual support and accessibility options, the platform ensures inclusivity for all citizens, regardless of language or ability. Ultimately, the Citizen AI platform aims to empower citizens, promote active participation, and support governments in delivering responsive, transparent, and efficient services that align with the evolving expectations of a digitally connected society.

• Features:

Conversational Interface

Key Point: Enables citizens to interact using natural language (text or voice).

Functionality: Citizens can ask questions, lodge complaints, or request services.Chatbots/voice assistants provide instant responses.Reduces the need for in-person visits or paperwork.

Policy Summarization

Key Point: Simplifies complex government policies.

Functionality: Uses AI to summarize long documents into short insights.Helps citizens quickly understand rules and schemes.Increases policy awareness and compliance.

Sentiment & Feedback Analysis

Key Point: Measures public opinion using AI.

Functionality: Analyzes feedback from surveys, portals, and social media.Detects citizen satisfaction, frustration, or concern.Helps governments identify areas for improvement.

Service Request Tracking

Key Point: Real-time updates on service requests.

Functionality: Citizens can check the status of complaints, applications, etc.Provides tracking IDs and notifications.Increases transparency and accountability.

Decision Support for Officials

Key Point: Data-driven tools for administrators.

Functionality: Dashboards with trends, issues, and KPIs.Helps monitor service efficiency and citizen engagement.Supports better planning and faster decisions.

Multi-Language Support

Key Point: Makes the platform inclusive and accessible.

Functionality: Citizens can interact in their preferred regional language.Supports translation and voice recognition.Bridges the language barrier for rural users.

Secure Data Handling

Key Point: Ensures data privacy and compliance.

Functionality: Role-based access and strong encryption.Follows data protection laws and policies.Protects sensitive citizen information.

Mobile & Web User Interface

Key Point: Access anytime, anywhere.

Functionality: Available on both mobile and desktop platforms.Simple, responsive, and user-friendly design.Makes digital governance accessible to all.

3. Architecture

Frontend (Streamlit/Gradio):

Provides dashboards, complaint submission forms, chat interface, and service tracking.

Backend (FastAPI):

Handles API requests for feedback collection, complaint resolution, and AI-driven services.

LLM Integration (IBM Watsonx / Open-source models):

Used for summarization, sentiment analysis, and query handling.

Vector Search (Pinecone/FAISS):

Enables semantic search through past complaints, policies, and FAQs.

ML Modules:

Predicts emerging issues (e.g., traffic, sanitation complaints) and detects anomalies in service delivery.

4. Setup Instructions

Prerequisites:

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).

Installation Process:

o Clone the repository

o Install dependencies from requirements.txt

o Create a .env file and configure credentials

o Run the backend server using Fast API

o Launch the frontend via Stream lit

o Upload data and interact with the modules

5.Folder Structure

app/ → Backend logic for API routes (chat, complaints, feedback, reports).

ui/ → Frontend with dashboards, forms, and chat interface.

citizen_ai_llm.py → Handles interaction with AI models.

feedback_analyzer.py → Performs sentiment analysis.

policy_summarizer.py → Converts government documents to summaries.

service_tracker.py → Tracks complaint and service requests.

report_generator.py → Generates citizen engagement and satisfaction reports.

6. Running the Application

- Start FastAPI server.
- Run the Streamlit/Gradio dashboard.
- Citizens can file complaints, ask questions, and view service status.
- Officials can access reports, analytics, and citizen sentiment insights.

7. API Documentation

Backend APIs available include:

POST /chat/ask – Citizen queries and AI-generated responses.

POST /submit-feedback – Collects citizen complaints or feedback.

GET /analyze-feedback – Returns sentiment analysis results.

GET /policy-summary – Provides summarized version of uploaded policies.

GET /service-status/{id} – Tracks citizen service request status.

8. Authentication

each endpoint is tested and documented in Swagger UI for quick inspection

and trial during development.

This version of the project runs in an open environment for demonstration.

However, secure deployments can integrate:

- JWT or API Key-based login.
- OAuth2 for government officials.
- Role-based access (Citizen, Officer, Admin).

9. User Interface

The interface is minimalist and functional, focusing on accessibility for non-

technical users. It includes:

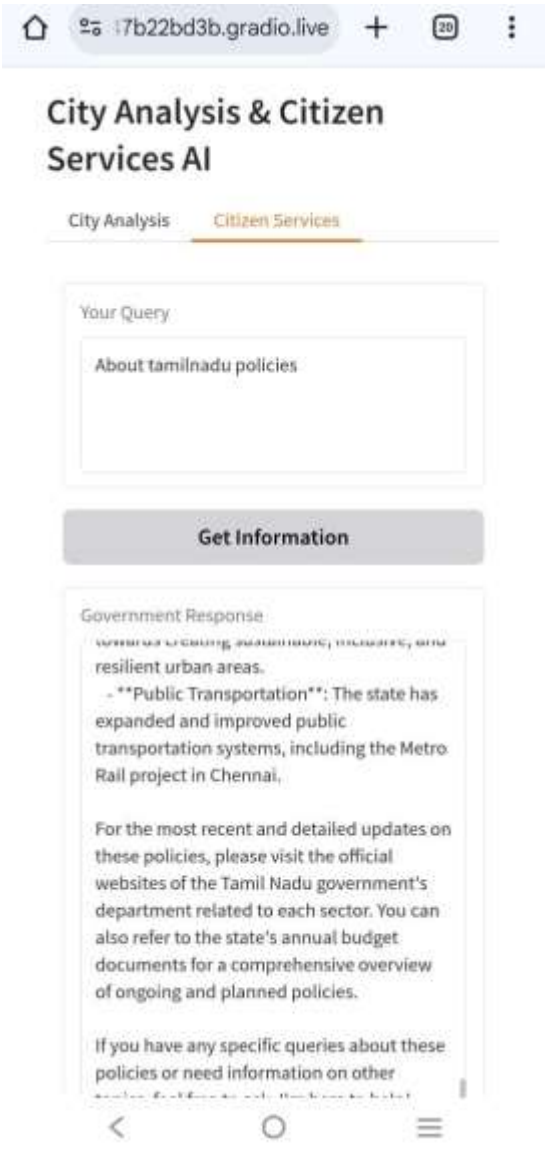
- Citizen portal (complaints, FAQs, service tracking).
- Government dashboard (analytics, feedback analysis, engagement trends).
- Multilingual chat support.
- Real-time notifications and downloadable reports.

10. Testing

Testing was done in multiple phases:

- Unit Testing: For AI functions (summarization, sentiment analysis).
- API Testing: Swagger/Postman.
- Manual Testing: Citizen complaints, policy uploads, and dashboards.
- Edge Cases: Wrong inputs, unsupported file types, invalid service requests.

11.screen shots



12. Known Issues

- Limited offline mode.
- Dependency on stable internet.
- Requires continuous retraining of AI models for accuracy.

13. Future enhancement

- Voice-enabled citizen support.
- Integration with WhatsApp/Telegram bots.
- Predictive governance (forecasting citizen issues).
- Blockchain-based complaint tracking for transparency.