REPORT

ON

“Citizen AI – Intelligent Citizen Engagement Platform”

Submitted in partial fulfillment of the requirements of the

Virtual Internship Program

Organized by

SMART BRIDGE

Submitted by

Partha Sarathi R

Karthick S

Gokul s

Mushraf S K

TEAM ID:

NM2025TMID00557

Index

BRAINSTORMING &IDEATION

Problem Context

Governments often struggle with delayed and inefficient communication with the public. Traditional systems such as phone calls, static websites, or manual helpdesks make it difficult for citizens to get timely and relevant information regarding public services, policies, and civic issues.

Additionally, the government lacks effective tools to understand the overall mood or sentiment of citizens based on their feedback. As a result, citizen trust and engagement with digital governance platforms remain limited.

Purpose of the Project

Citizen AI was developed to solve these challenges by introducing a real-time AI assistant capable of:

- Answering citizen questions with human-like responses using IBM Granite models.
- Analyzing citizen feedback to detect sentiments such as satisfaction or dissatisfaction.
- Presenting actionable insights on a dynamic dashboard.
- Enhancing government responsiveness, transparency, and public trust through AI-powered interactions.

Problem Statement

- Citizens face delays in receiving information or reporting issues.
- Feedback collected by government departments is rarely analyzed in real time.
- There is no central platform to combine public service interaction, sentiment analysis, and policy responsiveness.

Proposed Solution

- **Citizens:** To ask questions, provide feedback, and get quick answers.
- **Government Officials:** To monitor public sentiment and improve services.
- **Policy Makers:** To use data insights for better governance decisions.
- **Developers/Admins:** To manage system operations and improve AI performance.

Target Users

- **City Residents** – To report civic issues easily and get eco-advice.
- **City Administrators –** To monitor feedback, analyze KPIs, and respond to anomalies.
- **Urban Planners –** To summarize long documents and make informed policy decisions.
- **Teachers & Students –** To explore sustainability practices via the Eco Tips Generator.
- **Government Departments –** For utility monitoring and forecasting resource demands

.

Expected Outcomes

- A responsive, intelligent chatbot available 24/7 for public queries.
- Real-time analysis of citizen feedback to detect trends and issues.
- Dashboards that visualize public mood and interaction frequency.
- Enhanced digital governance through AI, leading to improved public satisfaction and trust.

REQUIREMENT ANALYSIS

Technical Requirements:

- Backend developed using **Flask (Python)** with RESTful routing.
- Frontend created using **HTML, CSS, and JavaScript** for interactive UI.
- AI integration handled via **IBM Granite API** for NLP and chat responses.
- Database setup using **SQLite or PostgreSQL** for structured data storage.
- Sensitive data and configuration managed through .env and config.py.
- Chat latency kept under **3 seconds** per response for smooth UX.
- Scalable and modular design to support multiple users and future features.
- Input sanitization and secure API key handling for data protection.
- Unit and integration tests included to ensure quality and reliability.

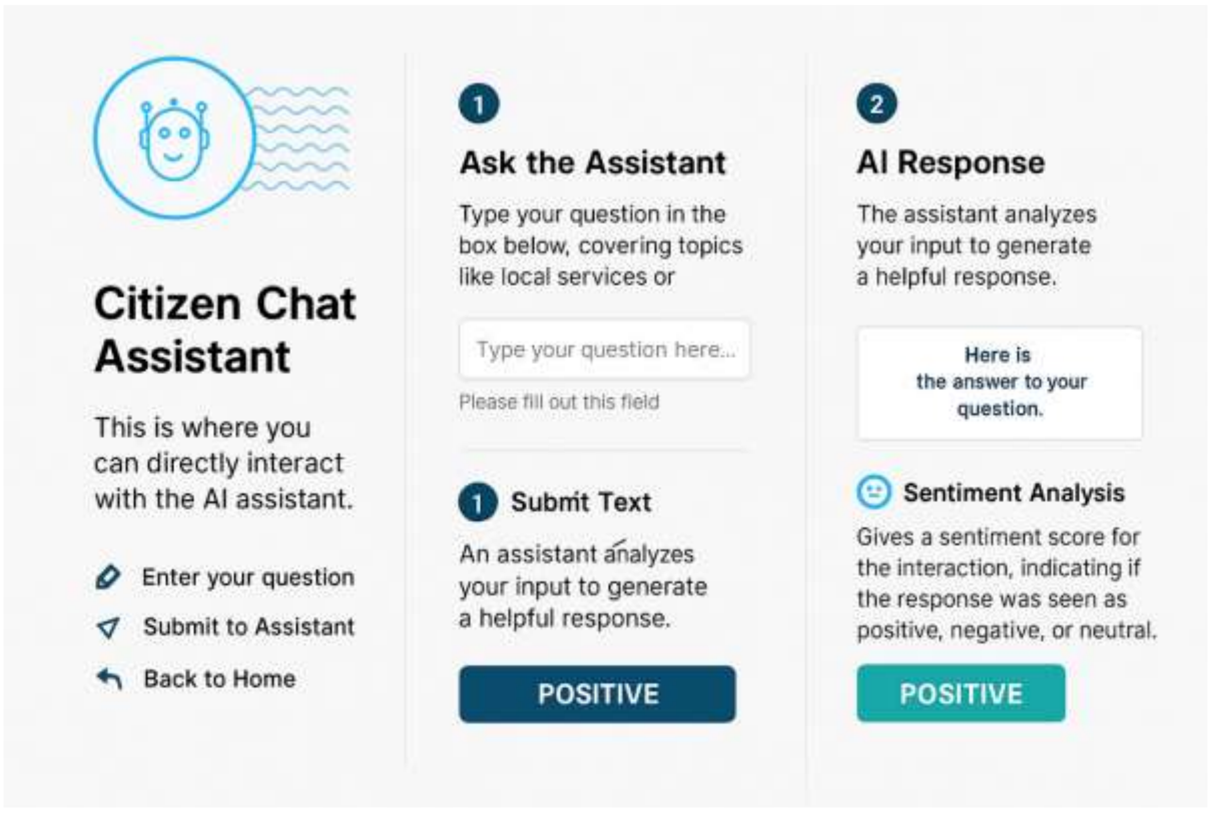
Functional Requirements:

- Enable real-time chatbot interaction using IBM Granite.
- Accept citizen queries and provide human-like responses.
- Allow feedback submission through a web form interface.
- Automatically analyze sentiment (Positive, Neutral, Negative) from feedback.
- Display sentiment insights and user trends on a dynamic dashboard.
- Provide personalized and context-aware replies based on session history.
- Store all user queries, feedback, and sentiment results in a secure database.

PROJECT DESIGN

The Citizen AI platform follows a modular Flask architecture with distinct components for routes, models, templates, and static files, ensuring clean structure and maintainability. User queries are handled via chat_routes.py, processed by granite_interface.py using IBM Granite, and returned in real-time through chat.js. Feedback is collected via feedback_routes.py, analyzed by sentiment_analysis.py, and stored in the database. Configuration is securely managed through .env and config.py, supporting smooth AI-driven workflows.

Block Diagram of the project



Citizen Chat Assistant

This is where you can directly interact with the AI assistant.

Enter your question

Submit to Assistant

Back to Home

1 Ask the Assistant

Type your question in the box below, covering topics like local services or

Type your question here...

Please fill out this field

1 Submit Text

An assistant analyzes your input to generate a helpful response.

POSITIVE

2 AI Response

The assistant analyzes your input to generate a helpful response.

Here is the answer to your question.

Sentiment Analysis

Gives a sentiment score for the interaction, indicating if the response was seen as positive, negative, or neutral.

POSITIVE

flow:

- A **citizen** opens the platform and interacts with the chatbot by typing a query.
- The message is sent to the backend, where IBM Granite processes it and returns a response.
- The citizen may also submit feedback using a form, which is analyzed for sentiment and stored in the database.
- An **admin** logs into the dashboard to monitor overall sentiment trends and feedback analytics.
- The system tracks session history, enabling context-aware, more personalized responses for returning or active users.

PROJECT PLANNING

The Project Planning phase is to structure the development of the Citizen AI platform into manageable tasks and timeframes using an Agile methodology.

By organizing work into focused sprints and distributing responsibilities across the team, the project ensures steady progress, timely feedback, and successful completion of each module. This structured plan includes sprint goals, task allocation, and milestone tracking.

Sprint Planning:

- **Sprint 1: Project Setup**
    - Initialize Flask project with app.py, config.py, .env, and install dependencies.
- **Sprint 2: Chatbot Development**
    - Build real-time chatbot with chat_routes.py, granite_interface.py, and frontend (chat.html, chat.js).
- **Sprint 3: Feedback & Sentiment**
    - Create feedback form, classify sentiment using sentiment_analysis.py, and store data in DB.
- **Sprint 4: Dashboard Integration**
    - Develop admin dashboard to visualize sentiment and engagement trends.
- **Sprint 5: Contextual Chat**
    - Implement session-based contextual replies using helpers.py.
- **Sprint 6: Database Setup**
    - Define schemas in models.py, initialize DB with init_db.py.
- **Sprint 7: Testing & QA**
    - Perform unit/integration testing and optimize performance.
- **Sprint 8: Deployment**
    - Prepare deployment configs, add demo data, finalize UI and documentation.

Tasks Allocation :

Members	Tasks

Manikanta Surya sai Sunkara	• Granite integration • chatbot logic • UI components, testing • final review
Murali mohan Venkata Durga Anil Karra	• Feedback module • database setup • .env configuration • deployment support
Venkata Suresh Swamireddy	• Sentiment analysis • Flask routing • frontend UI • unit testing
Harsith Kumar Vegi	• Dashboard implementation • admin analytics • database model definitions

TimeLine of the project

Date	**Milestone**
Week-1	• Flask setup • Granite API integration • chatbot module
Week-3	• Feedback form • sentiment analysis • database connectivity

Week 4	• Dashboard implementation • context tracking logic
Week 5	• Final testing, bug fixes, UI polish, documentation, and deployment

PROJECT DEVELOPMENT

- **Backend Framework:** Python with Flask (for modular API routing)
- **Frontend Technologies:** HTML, CSS, JavaScript (with Bootstrap)
- **AI Integration:** IBM Granite LLM (for chatbot and contextual responses)
- **Sentiment Analysis:** Custom Python module using NLP techniques
- **Database:** SQLite (development) / PostgreSQL (production-ready)
- **Configuration & Secrets:** .env file and config.py for environment-based setup
- **Version Control:** Git and GitHub
- **Deployment (Optional):** Docker, cloud hosting options

➢ Development Process:

The development of Citizen AI followed a structured and modular approach. The project was divided into independent yet connected components, ensuring flexibility and easier debugging. Each major feature was implemented and tested step by step, starting from backend setup to AI integration and frontend user experience. Below are the key stages of the development process:

Flask Project Initialization:

- Created the basic Flask structure with app.py, config.py, and .env for secure settings.

- Installed necessary dependencies including Flask, IBM Granite SDK, and SQLAlchemy.

Chatbot Module Implementation:

- Developed chat_routes.py to handle incoming chat messages.
- Integrated IBM Granite API through granite_interface.py to fetch AI responses.
- Designed chat.html and chat.js for real-time user interaction on the frontend.

Feedback & Sentiment Analysis:

- Created a feedback form and handled submissions via feedback_routes.py.
- Analyzed sentiment using the analyse_sentiment() function in sentiment_analysis.py.
- Stored feedback and sentiment data into the database using SQLAlchemy models.

Dashboard Development:

- Built dashboard_routes.py to serve sentiment and feedback metrics.
- Designed dashboard.html with embedded charts for real-time analytics using JavaScript.
- Displayed sentiment counts and citizen engagement trends for admin users.

Contextual Response Enhancement:

- Implemented session tracking to maintain chat context.
- Used helpers.py to store previous interactions and send enriched prompts to Granite for more relevant responses.

Database Integration:

- Defined all database schemas (User, Feedback, Sentiment) in models.py.
- Created init_db.py for easy initialization and reset during testing.

- Connected the application to SQLite/PostgreSQL for persistent data handling.

Testing and Final Integration:

- Performed module-wise testing followed by full integration.
- Addressed sync issues between frontend and backend.
- Ensured end-to-end functionality from chat input to dashboard

FUNCTIONAL & PERFOMANCE TESTING

- **Chatbot:**
  Verified real-time responses from IBM Granite for various citizen queries.
- **Feedback Submission:**
  Checked form handling, data storage, and confirmation messages.
- **Sentiment Analysis:**
  Tested classification as Positive, Neutral, or Negative for diverse feedback inputs.
- **Dashboard:**
  Validated data visualization, sentiment graphs, and trend accuracy.
- **Contextual Chat:**
  Ensured conversation history is maintained and used for relevant replies.
- **Database:**
  Confirmed data integrity, proper schema usage, and reliable storage.
- **Performance:**
  Measured response times (<3s), smooth UI experience, and multi-user handling.

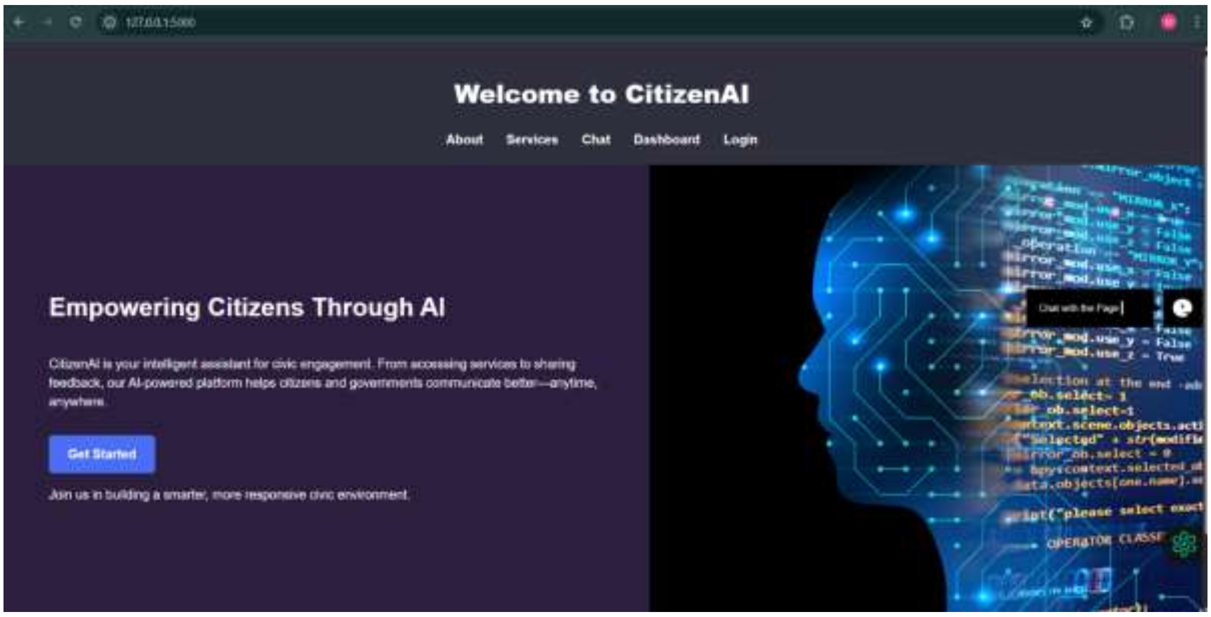
Final OUTPUT Validation:

After thorough testing, the Citizen AI platform was confirmed to:

- Handle real-time citizen interaction effectively.

- Process and store feedback accurately.
- Display sentiment trends and analytics correctly.
- Respond quickly and scale to multiple users.
- Meet the functional requirements defined in the planning phase.

Result of the project:-

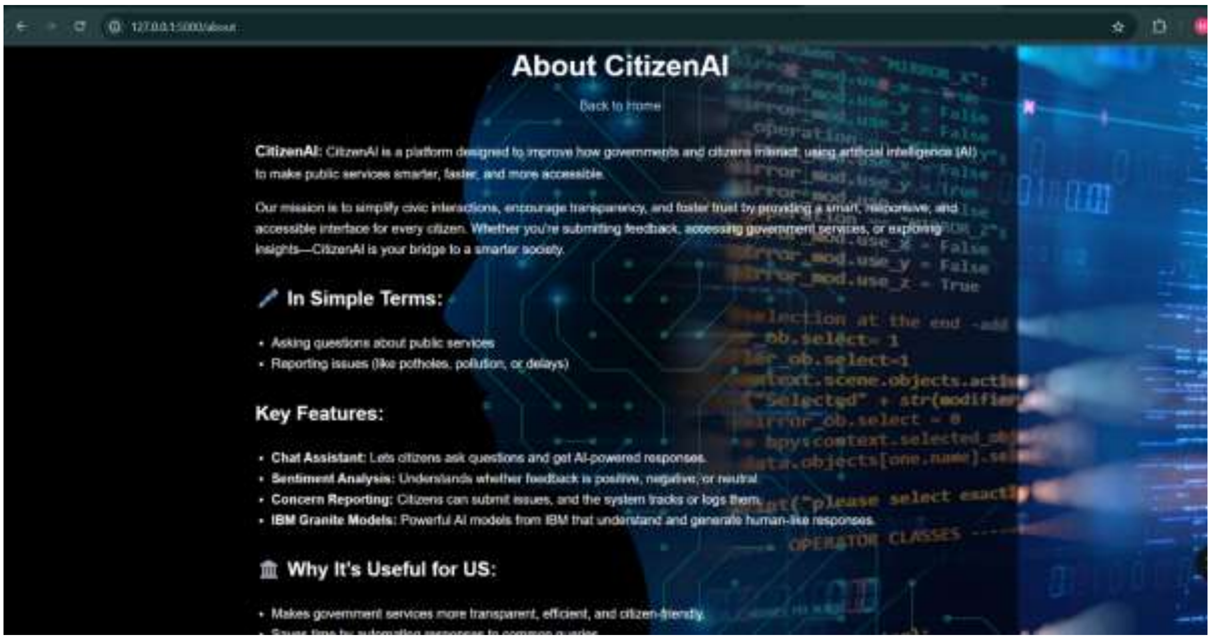


Welcome to CitizenAI

Empowering Citizens Through AI

CitizenAI is your intelligent assistant for civic engagement. From accessing services to sharing feedback, our AI-powered platform helps citizens and governments communicate better—anytime, anywhere.

Join us in building a smarter, more responsive civic environment.



About CitizenAI

CitizenAI: CitizenAI is a platform designed to improve how governments and citizens interact, using artificial intelligence (AI) to make public services smarter, faster, and more accessible.

Our mission is to simplify civic interactions, encourage transparency, and foster trust by providing a smart, responsive, and accessible interface for every citizen. Whether you're submitting feedback, accessing government services, or exploring insights—CitizenAI is your bridge to a smarter society.

In Simple Terms:

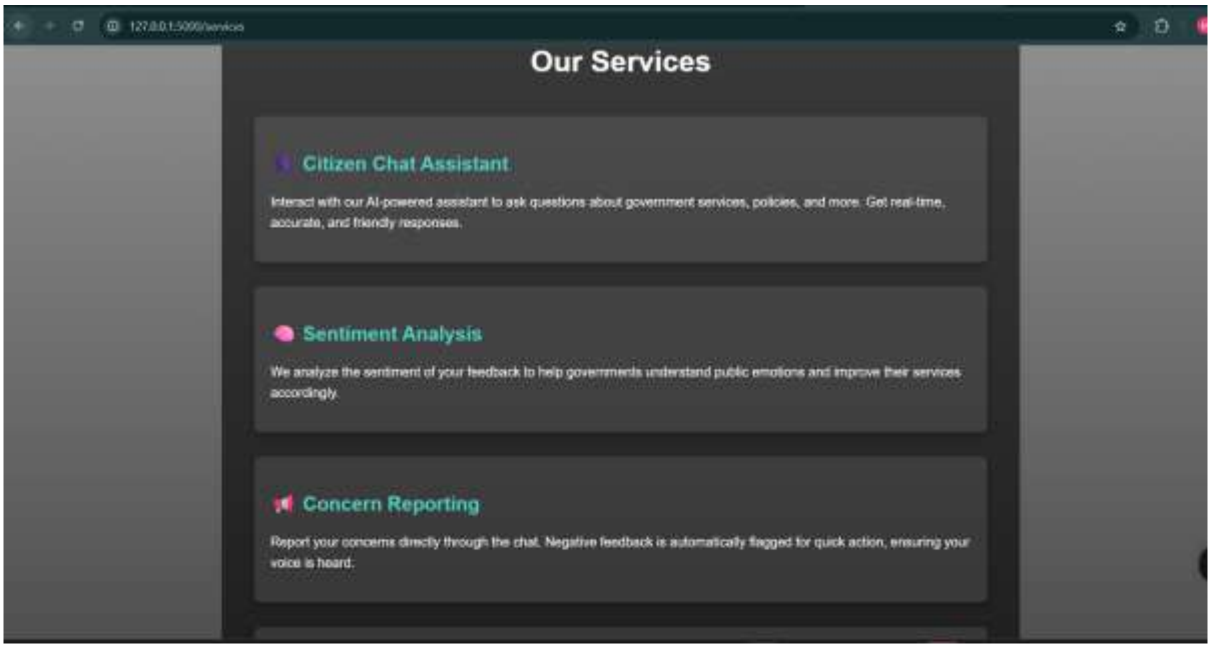
- Asking questions about public services
- Reporting issues (like potholes, pollution, or delays)

Key Features:

- **Chat Assistant:** Lets citizens ask questions and get AI-powered responses.
- **Sentiment Analysis:** Understands whether feedback is positive, negative, or neutral
- **Concern Reporting:** Citizens can submit issues, and the system tracks or logs them.
- **IBM Granite Models:** Powerful AI models from IBM that understand and generate human-like responses.

Why It's Useful for US:

- Makes government services more transparent, efficient, and citizen-friendly.



Our Services

Citizen Chat Assistant

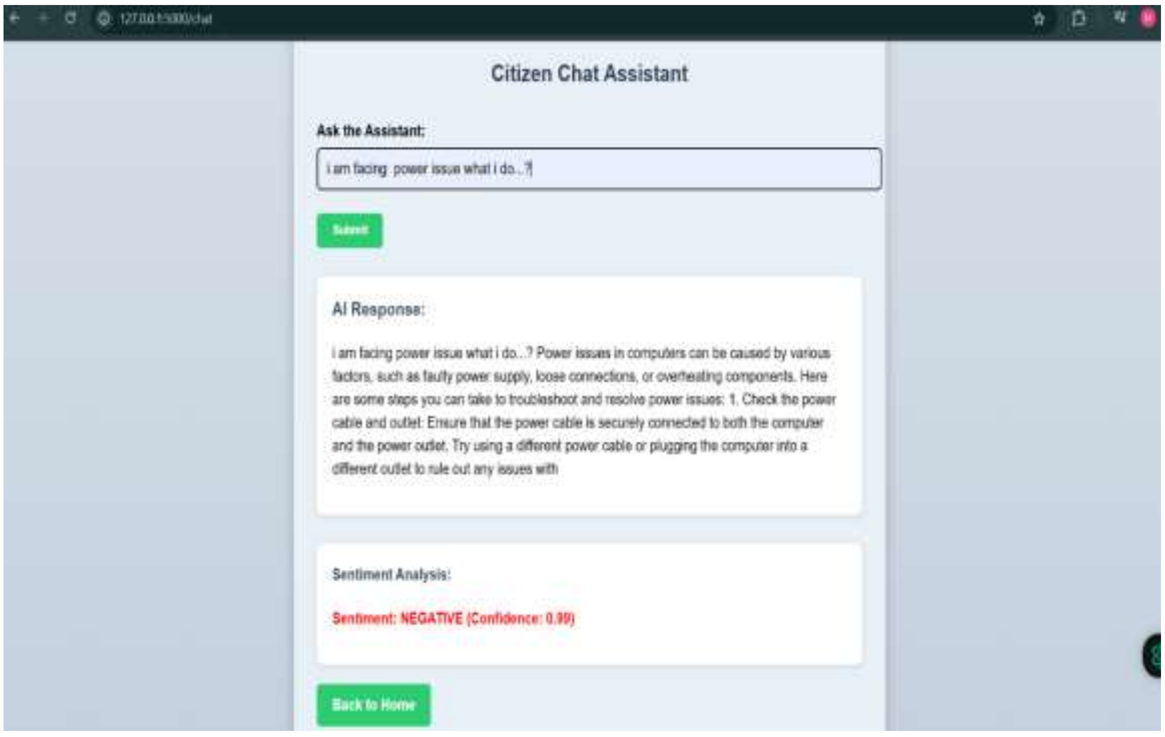
Interact with our AI-powered assistant to ask questions about government services, policies, and more. Get real-time, accurate, and friendly responses.

Sentiment Analysis

We analyze the sentiment of your feedback to help governments understand public emotions and improve their services accordingly.

Concern Reporting

Report your concerns directly through the chat. Negative feedback is automatically flagged for quick action, ensuring your voice is heard.



Citizen Chat Assistant

Ask the Assistant:

i am facing power issue what i do...?

AI Response:

i am facing power issue what i do...? Power issues in computers can be caused by various factors, such as faulty power supply, loose connections, or overheating components. Here are some steps you can take to troubleshoot and resolve power issues: 1. Check the power cable and outlet: Ensure that the power cable is securely connected to both the computer and the power outlet. Try using a different power cable or plugging the computer into a different outlet to rule out any issues with

Sentiment Analysis:

Sentiment: NEGATIVE (Confidence: 0.99)

ADVANTAGES & DISADVANTAGES:-

Advantages:-

AI-Powered Support: Automates citizen queries using advanced language models like IBM Granite.

24/7 Availability: The chatbot assistant is always available for citizens, unlike manual support.

Sentiment Analysis: Helps government bodies understand public opinion in real time.

Simple UI: Web interface is intuitive, making it easy even for non-technical users.

Scalable: The backend and AI components can be scaled using cloud infrastructure.

Rapid Deployment: Built using Python Flask and Transformers, making it easy to deploy and test.

✕ Disadvantages

Limited Training Context: Model lacks domain-specific fine-tuning for government datasets.

No Real-Time Database: Currently uses in-memory storage, making data non-persistent unless upgraded.

Single User Authentication: Only a hardcoded dummy user exists; lacks multi-user support.

Dependency on Internet: Model loading and sentiment analysis require internet access or large local resources.

CONCLUSION

CitizenAI bridges the gap between the government and the public by using AI-driven solutions to automate support, gather feedback, and report citizen concerns. This system increases transparency, trust, and responsiveness in governance. By integrating sentiment analysis and chatbot functionality into one platform, CitizenAI empowers both users and administrators with actionable insights. The project proves that AI can play a meaningful role in civic participation and e-governance.

FUTURE SCOPE

✅ Persistent Database: Integrate a cloud-based DB (e.g., Firebase, MongoDB) to store data across sessions.

✅ Multi-User Authentication: Add secure login with JWT/Flask-Login and support for different roles (Admin, Citizen).

✅ Chat History Dashboard: Visual dashboards for historical analytics and issue tracking.

✅ Mobile Application: Extend platform via a mobile frontend using Flutter or React Native.

✅ Voice Assistant Integration: Add speech-to-text and text-to-speech with IBM Watson STT or Google APIs.

✅ Multilingual Support: Allow citizens to interact in regional languages using translation models.

✅ Fine-tuning Models: Use custom datasets to improve relevance and contextual understanding.