GENERATIVE AI WITH IBM CLOUD

1) INTRODUCTION

- **PROJECT TITLE - Citizen AI – Intelligent Citizen Engagement Platform**
- **TEAM MEMBERS - Team Leader : Nelli Jegadeeswari**

  Team member : Gunupudi Surya Swarnitha

  Team member : Erothu Naga Uday Bhaskar

  Team member : Guthi Naga Karthikeya

- **DISCRIPTION -** Citizen AI is a responsible and ethical AI framework designed to serve the needs of all citizens fairly.It ensures transparency, accountability, and inclusivity in all its operations.
  The system is built to reduce bias and promote equal access to information and services.
  Citizen AI supports decision-making in public sectors like healthcare, education, and governance.
  It empowers communities by involving them in the development and oversight of AI tools.
  Security and data privacy are core principles, ensuring trust and safety for users.
  Overall, Citizen AI bridges the gap between technology and society for the greater public good.

2)PROJECT OVERVIEW

Citizen AI is an AI-powered civic assistant designed to foster responsible, fair, and transparent public engagement. It leverages generative AI (such as OpenAI or open-source models) to assist users with government services, citizen rights awareness, community resources, and digital participation — all through an intuitive and ethical interface.
Built using modern AI frameworks like FastAPI, Gradio, and HuggingFace Transformers, Citizen AI bridges the gap between people and civic systems through inclusive, accountable, and user-centric design.

Purpose:
The primary goal of Citizen AI is to:

- Empower citizens with accessible and unbiased civic information
- Promote transparency and trust in digital governance
- Enable participation in democratic processes through education and engagement
- Provide quick assistance for public services and grievance redressal
- Support inclusivity and fairness in AI-driven civic platforms

Key Features:

- **AI Chat Assistant for Civic Queries:** Provides real-time answers about government services, rights, and policies.
- **Sentiment Analysis for Citizen Feedback:** Analyzes the emotional tone of user messages to better understand public sentiment and prioritize critical issues.
- **Feedback Collection & Analysis System:** Enables users to submit suggestions, complaints, or concerns; uses AI to categorize and summarize feedback for continuous improvement.
- **Multilingual Support:** Offers services in multiple languages to ensure accessibility for diverse communities.
- **Accessibility-First Design:** Optimized for mobile devices and low-bandwidth environments to reach underserved areas.

- **Data Privacy & Ethical AI Compliance:** Follows strict data protection protocols and ethical AI practices for safe, trustworthy usage.

3)<u>SYSTEM ARCHITECTURE</u>

FRONTEND:

- **Tool Used:** Gradio
- **Purpose:** Provides a clean, user-friendly web interface for citizens to interact with the Citizen AI system**.**
- **Functionality:**
  - Users can input natural language queries related to public services, rights, or community issues.
  - The interface displays AI-generated, verified, and contextual responses in real-time.
  - Organized into intuitive tabs: Civic Chat, Government Schemes, Sentiment Feedback, and Service Finder.
  - No installation required — accessible through any browser via a shareable link**.**

BACKEND:

- **Platform:** Google Colab (Python runtime environment)
- **Languages & Libraries Used:**
  - Transformers – for loading IBM Granite model from Hugging Face.
  - Torch – for efficient model inference using CPU or GPU.
  - Gradio – for integrating backend logic with the frontend interface.
- **Functionality:**
  - Accepts user input from the frontend and formats it into a structured, context-aware prompt.
  - Sends the prompt to the AI model for semantic understanding and generation.
  - Post-processes the model output to ensure clarity, tone, and factual accuracy.
  - Returns formatted responses to the frontend in real time.
  - Powers additional features like Sentiment Analysis, Feedback Summarization, and Civic Data Lookup using Python’s NLP and data processing libraries.

AI MODEL:

- **Model Used:** IBM Granite-3.3-2B-Instruct
- **Hosted via:** Hugging Face Hub
- **Role in the System:**
  - Core intelligence engine of the application, driving real-time civic interactions.
- **Capable of:**
  - Understanding queries related to public services, government schemes, and legal rights.
  - Responding empathetically to citizen concerns and feedback.
  - Summarizing and interpreting policies, FAQs, and procedural guides.
  - Classifying feedback sentiment and urgency.

DATABASE HANDLING:

- **Current Setup:**
    - Used SQLLite for Storing Data.
- **For Production Use (Recommended):**
    - Integrate cloud-based databases such as Firebase SQLlite, PostgreSQL, or MongoDB to handle:
        - Citizen interaction history and session logs
        - Common/repeated civic queries
        - Feedback and sentiment reports
        - Role-based access (admin, citizen, volunteer)
    - Include basic logging, audit trails, and analytics dashboards for monitoring usage and improving service delivery.

4)<u>SETUP INSTRUCTIONS -CITIZEN AI</u>

✅ **1. Prerequisites:**
Before starting the Citizen AI setup, make sure the following resources are available:

- ✅ **Google Account** – Required to access and run the project in Google Colab.
- ✅ **Python (3.x)** – Pre-installed in Colab; used as the primary programming language.
- ✅ **Reliable Internet Connection** – Necessary for loading AI models and serving the interface.
- ✅ **GitHub Account (optional)** – Useful for cloning project files if hosted in a repository.

✅ **Hugging Face Account** – To access the IBM Granite model (if authentication is required).

📦 **2. Required Libraries:**
Install the following Python packages inside your Colab notebook:

■ !pip install transformers gradio langchain.

Usage of Libraries:

- transformers – For loading and running language models.
- gradio – For building a web-based interactive UI.
- langchain – (Optional) For multi-turn query handling and response chaining.

🔧 **3. Model Setup (IBM Granite 3B Instruct)**

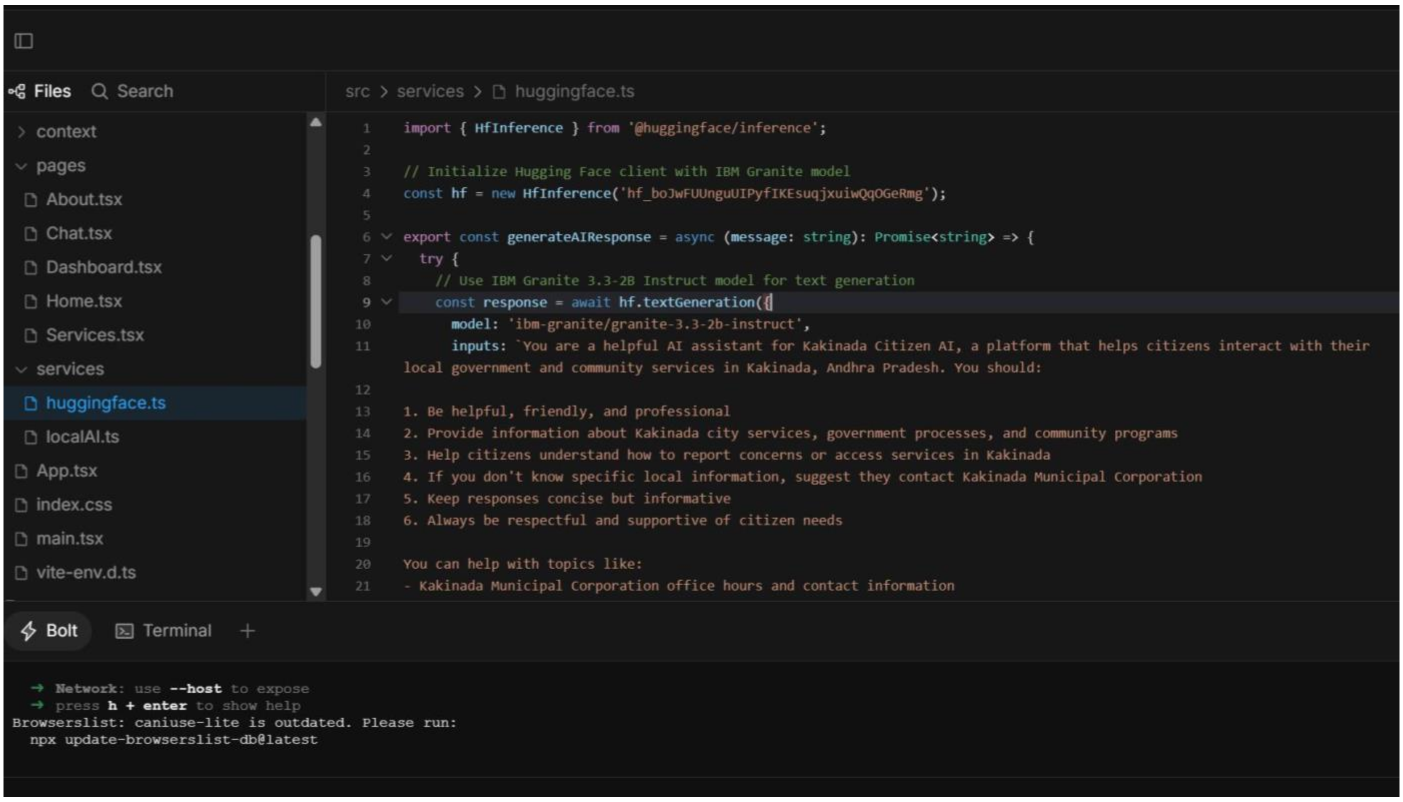
- **Load the model from Hugging Face:**

```
from transformers import AutoModelForCausalLM, AutoTokenizer

model_name = "ibm-granite/granite-3b-instruct"

tokenizer = AutoTokenizer.from_pretrained(model_name)

model = AutoModelForCausalLM.from_pretrained(model_name)
```



```
import { HfInference } from '@huggingface/inference';

// Initialize Hugging Face client with IBM Granite model
const hf = new HfInference('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');

export const generateAIResponse = async (message: string): Promise<string> => {
  try {
    // Use IBM Granite 3.3-2B Instruct model for text generation
    const response = await hf.textGeneration({
      model: 'ibm-granite/granite-3.3-2b-instruct',
      inputs: `You are a helpful AI assistant for Kakinada Citizen AI, a platform that helps citizens interact with their local government and community services in Kakinada, Andhra Pradesh. You should:

1. Be helpful, friendly, and professional
2. Provide information about Kakinada city services, government processes, and community programs
3. Help citizens understand how to report concerns or access services in Kakinada
4. If you don't know specific local information, suggest they contact Kakinada Municipal Corporation
5. Keep responses concise but informative
6. Always be respectful and supportive of citizen needs

You can help with topics like:
- Kakinada Municipal Corporation office hours and contact information
```

```
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
Browserslist: caniuse-lite is outdated. Please run:
  npx update-browserslist-db@latest
```

💡 **Note:**

- You may need to be logged into Hugging Face and use an access token if the model is gated or requires authentication.

# Optional: Using Hugging Face token (if needed)

from huggingface_hub import login

login("your_huggingface_token")

🔗 **You can find the model here:** https://huggingface.co/ibm-granite/granite-3b-instruct

🌐 **4. Gradio UI Setup:**

- **Create a simple interface:**

```
import gradio as gr

def chat_function(user_input):

    # Pass input to model and return response (example placeholder)

    return "Citizen AI Response to: " + user_input

interface = gr.Interface(

    fn=chat_function,

    inputs="text",

    outputs="text",

    title="Citizen AI Assistant",
    description="Ask about government schemes, services, or civic rights."
)
interface.launch()
```

5)FOLDER STRUCTURE

```
CitizenAI/
├── client/
│   ├── ui.py               # Frontend (Gradio interface)
│   └── assets/             # (Optional) Images, logos, icons for branding
│
├── server/                 # Backend (model + logic)
│   ├── model_handler.py      # Load & run the IBM Granite model
│   ├── civic_response.py     # Civic question understanding and response generation logic
│   ├── sentiment_analysis.py  # Sentiment detection and feedback classification
│   └── utils.py            # Helper functions (e.g., formatting, validation)
│
├── config/
│   └── settings.py          # (Optional) API keys, model names, environment flags
│
├── requirements.txt         # Python dependencies
├── README.md               # Project overview and setup guide
└── main.ipynb              # Google Colab notebook (entry point to run everything)
```

📁 **Explanation:**

client/ – Frontend

- Handles user interaction using **Gradio**.
- ui.py sets up the chatbot interface and links to backend response logic.
- assets/ may contain logos, icons, or other visuals for the UI.

server/ – Backend

- Core AI processing happens here:
  - model_handler.py loads and manages the **IBM Granite 3B Instruct** model.
  - civic_response.py processes queries about rights, government services, or public resources.
  - sentiment_analysis.py analyzes user feedback to detect tone and urgency.

- utils.py includes helper methods like text cleaning or message formatting.

config/ – *(Optional)*

- Store config files such as Hugging Face tokens, model names, or environment-specific options.

main.ipynb –

- A **Google Colab notebook** that integrates all modules.
- Run this notebook to launch the complete system — UI, model, and all logic.

6)<u>RUNNIG THE APPLICATION</u>

Step 1: Set Up the Environment

You'll be using **Google Colab** as your main development and runtime platform.

✅ **Open a new notebook in Google Colab**
✅ **Install required libraries**:

```
!pip install gradio transformers langchain
```

⚙️ **Step 2: Backend (AI Model Logic)**

The backend is responsible for processing user input and generating meaningful civic responses.

✅ **Load the IBM Granite Model**:

```
from transformers import AutoTokenizer, AutoModelForCausalLM

model_name = "ibm-granite/granite-3b-instruct"

tokenizer = AutoTokenizer.from_pretrained(model_name)

model = AutoModelForCausalLM.from_pretrained(model_name)
```

✅ **Define the response generation function**:

```
def generate_response(prompt):

   inputs = tokenizer(prompt, return_tensors="pt")

   outputs = model.generate(**inputs, max_new_tokens=100)

   return tokenizer.decode(outputs[0], skip_special_tokens=True)
```

🌐 **Step 3: Frontend (Gradio Interface)**

This is the interface where users interact with the Citizen AI assistant.

✅ **Set up Gradio UI**

```
import gradio as gr

def chat_interface(user_input):

   response = generate_response(user_input)

   return response
```

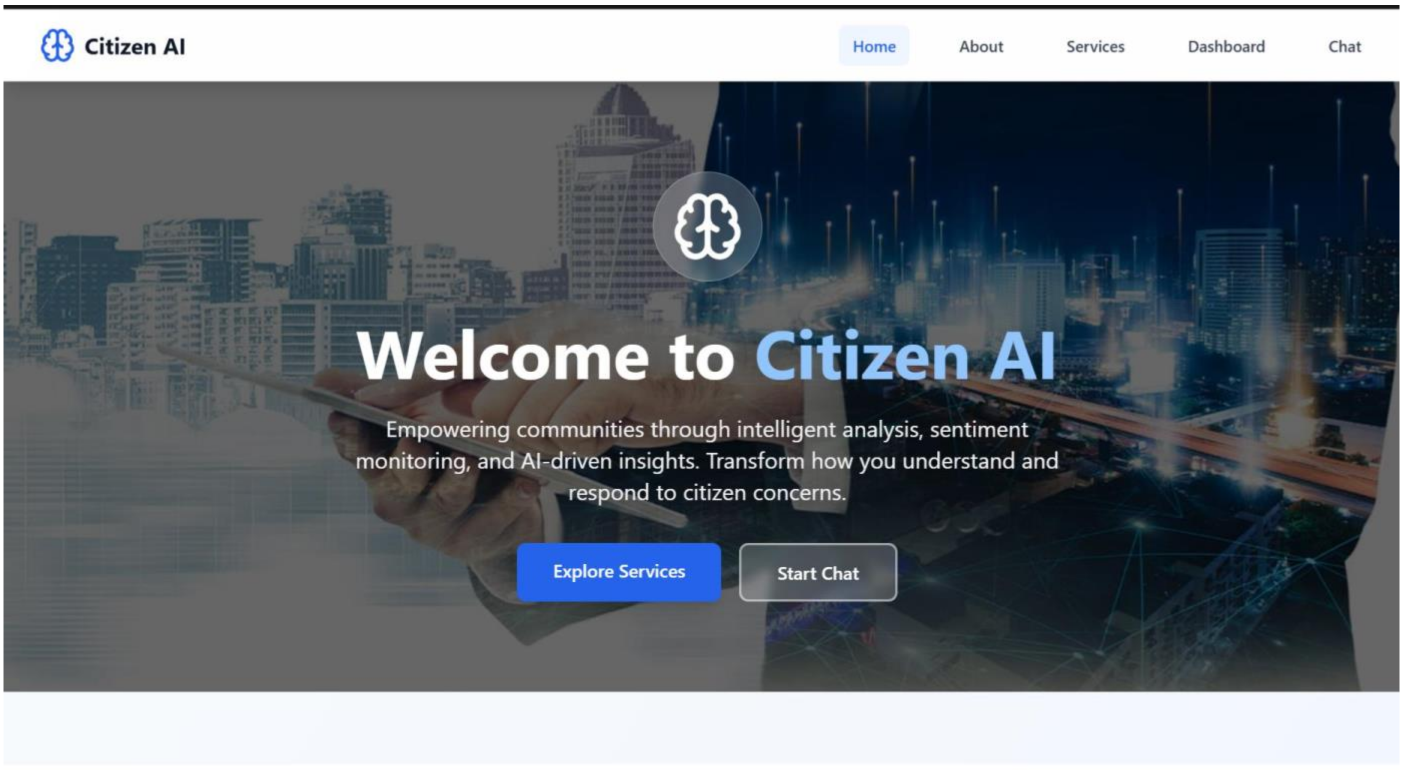
```
interface = gr.Interface(
    fn=chat_interface,
    inputs="text",
    outputs="text",
    title="Citizen AI Assistant"
)
interface.launch()
```

✅ When you run this cell, it will generate a public link. Clicking it will open the **Citizen AI** chatbot in your browser.

🔄 **Step 4: Integrate Full Workflow**

If you’ve created separate files/modules (e.g., for sentiment analysis, service lookup, or feedback tracking), import them into the notebook and connect them to the Gradio interface using your custom logic.

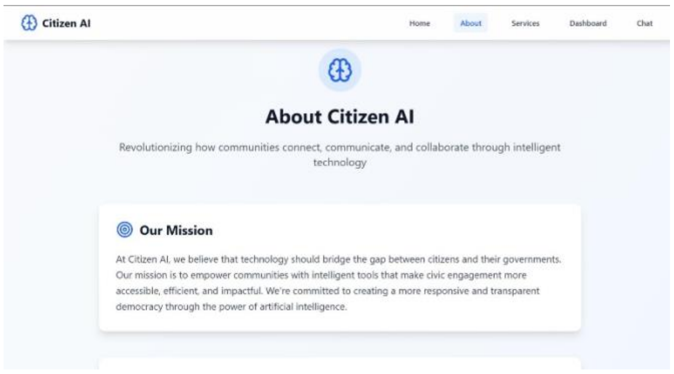
HOME PAGE:



Welcome to Citizen AI

Empowering communities through intelligent analysis, sentiment monitoring, and AI-driven insights. Transform how you understand and respond to citizen concerns.

ABOUT PAGE:

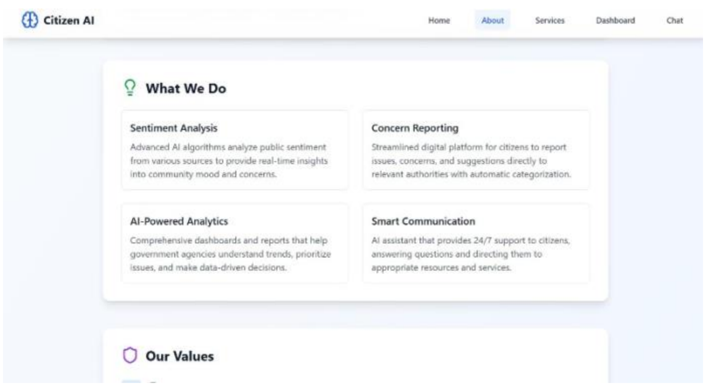


About Citizen AI

Revolutionizing how communities connect, communicate, and collaborate through intelligent technology

Our Mission

At Citizen AI, we believe that technology should bridge the gap between citizens and their governments. Our mission is to empower communities with intelligent tools that make civic engagement more accessible, efficient, and impactful. We're committed to creating a more responsive and transparent democracy through the power of artificial intelligence.



What We Do

Sentiment Analysis

Advanced AI algorithms analyze public sentiment from various sources to provide real-time insights into community mood and concerns.

Concern Reporting

Streamlined digital platform for citizens to report issues, concerns, and suggestions directly to relevant authorities with automatic categorization.

AI-Powered Analytics

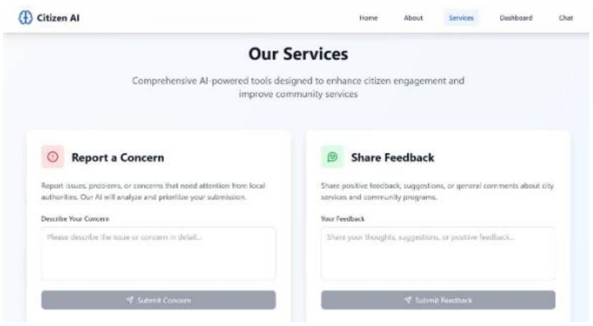
Comprehensive dashboards and reports that help government agencies understand trends, prioritize issues, and make data-driven decisions.

Smart Communication

AI assistant that provides 24/7 support to citizens, answering questions and directing them to appropriate resources and services.

Our Values

SERVICES PAGE:



Our Services

Comprehensive AI-powered tools designed to enhance citizen engagement and improve community services

Report a Concern

Report issues, problems, or concerns that need attention from local authorities. Our AI will analyze and prioritize your submission.

Describe Your Concern

Please describe the issue or concern in detail...

Submit Concern

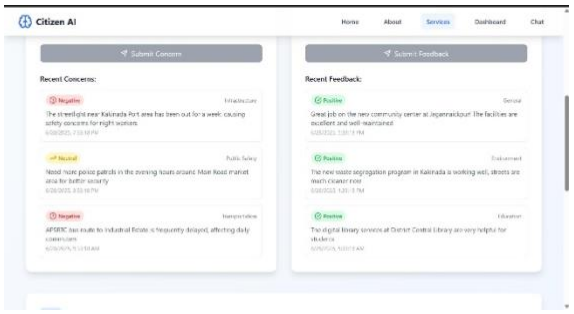
Share Feedback

Share positive feedback, suggestions, or general comments about city services and community programs.

Your Feedback

Share your thoughts, suggestions, or positive feedback...

Submit Feedback



Recent Concerns:

Negative — Infrastructure
The streetlight near Kakinada Port area has been out for a week causing safety concerns for night workers
[illegible]

Neutral — Public Safety
Need more police patrols in the evening hours around Main Road market area for better security
[illegible]

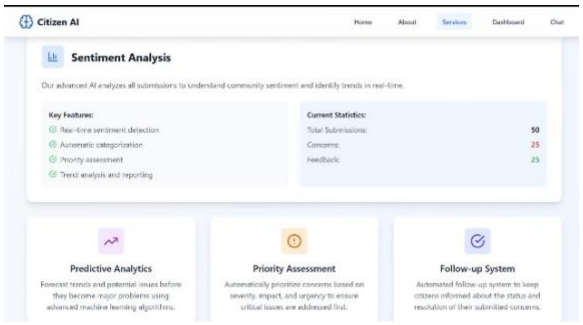
Negative — Transportation
APSRTC bus route to Industrial Estate is frequently delayed, affecting daily commuters
[illegible]

Recent Feedback:

Positive — General
Great job on the new community center at Jagannaickpur! The facilities are excellent and well-maintained
[illegible]

Positive — Environment
The new waste segregation program in Kakinada is working well, streets are much cleaner now
[illegible]

Positive — Education
The digital library services at District Central Library are very helpful for students
[illegible]



Sentiment Analysis

Our advanced AI analyzes all submissions to understand community sentiment and identify trends in real-time.

Key Features:
- Real-time sentiment detection
- Automatic categorization
- Priority assessment
- Trend analysis and reporting

Current Statistics:	
Total Submissions:	50
Concerns:	25
Feedback:	25

Predictive Analytics

Forecast trends and potential issues before they become major problems using advanced machine learning algorithms.

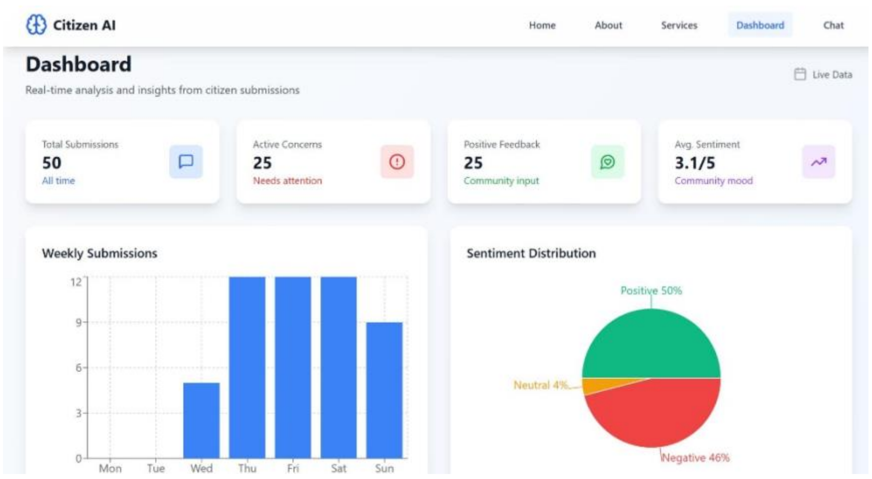
Priority Assessment

Automatically prioritize concerns based on severity, impact, and urgency to ensure critical issues are addressed first.

Follow-up System

Automated follow-up system to keep citizens informed about the status and resolution of their submitted concerns.

DASHBOARD PAGE:


Citizen AI · Home · About · Services · Dashboard · Chat

Dashboard

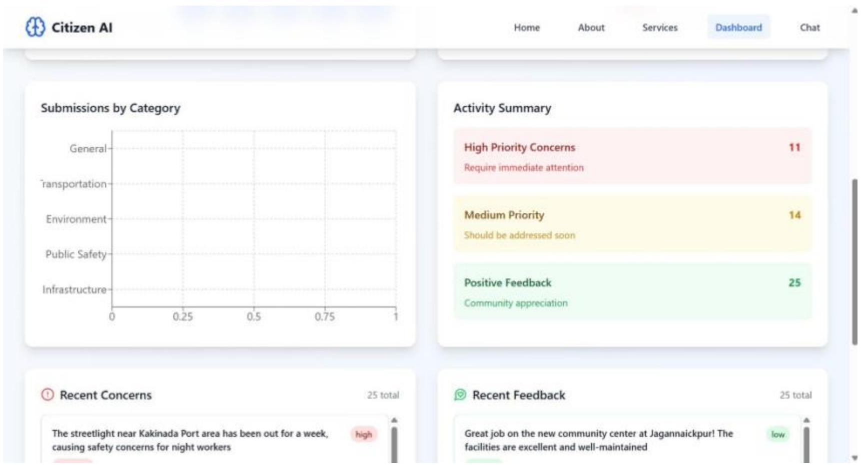
Real-time analysis and insights from citizen submissions

Live Data

Total Submissions	Active Concerns	Positive Feedback	Avg. Sentiment
50	25	25	3.1/5
All time	Needs attention	Community input	Community mood

Weekly Submissions

Sentiment Distribution


Citizen AI · Home · About · Services · Dashboard · Chat

Submissions by Category

Activity Summary

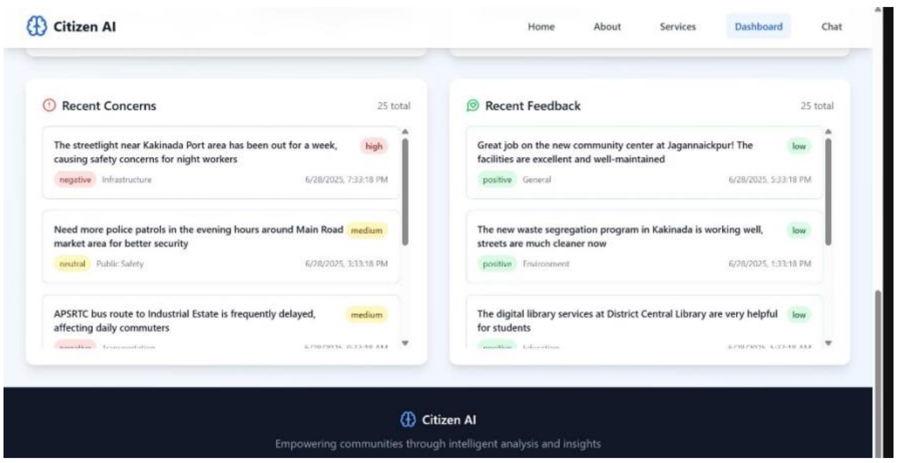
High Priority Concerns Require immediate attention	11
Medium Priority Should be addressed soon	14
Positive Feedback Community appreciation	25

Recent Concerns 25 total

The streetlight near Kakinada Port area has been out for a week, causing safety concerns for night workers high

Recent Feedback 25 total

Great job on the new community center at Jagannaickpur! The facilities are excellent and well-maintained low


Citizen AI · Home · About · Services · Dashboard · Chat

Recent Concerns 25 total

The streetlight near Kakinada Port area has been out for a week, causing safety concerns for night workers high
negative Infrastructure 6/28/2025, 7:33:18 PM

Need more police patrols in the evening hours around Main Road market area for better security medium
neutral Public Safety 6/28/2025, 3:33:18 PM

APSRTC bus route to Industrial Estate is frequently delayed, affecting daily commuters medium

Recent Feedback 25 total

Great job on the new community center at Jagannaickpur! The facilities are excellent and well-maintained low
positive General 6/28/2025, 5:33:18 PM

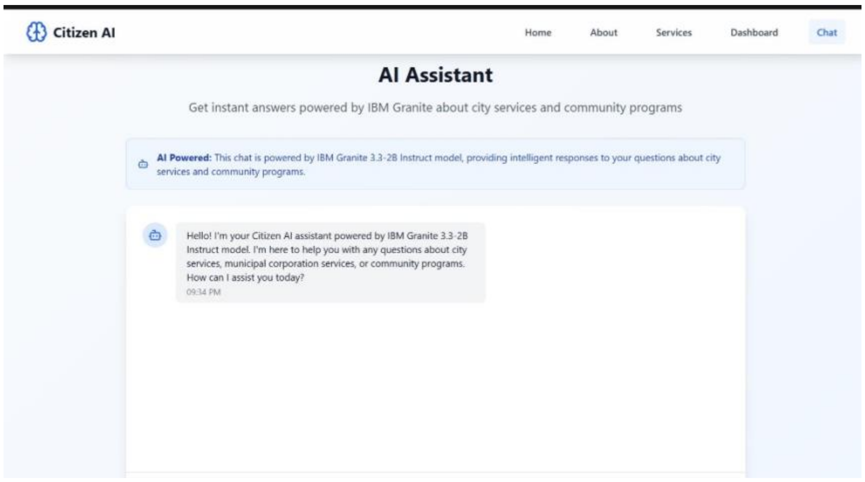
The new waste segregation program in Kakinada is working well, streets are much cleaner now low
positive Environment 6/28/2025, 1:33:18 PM

The digital library services at District Central Library are very helpful for students low

Citizen AI

Empowering communities through intelligent analysis and insights

CHAT ASSISTANT PAGE:

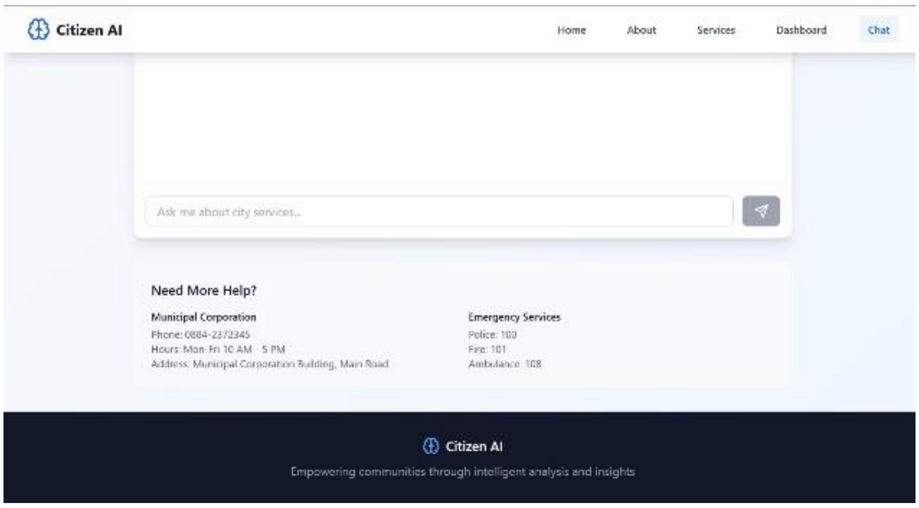


AI Assistant

Get instant answers powered by IBM Granite about city services and community programs

AI Powered: This chat is powered by IBM Granite 3.3-2B Instruct model, providing intelligent responses to your questions about city services and community programs.

Hello! I'm your Citizen AI assistant powered by IBM Granite 3.3-2B Instruct model. I'm here to help you with any questions about city services, municipal corporation services, or community programs. How can I assist you today?
09:34 PM



Need More Help?

Municipal Corporation
Phone: 0884-2372345
Hours: Mon-Fri 10 AM - 5 PM
Address: Municipal Corporation Building, Main Road

Emergency Services
Police: 100
Fire: 101
Ambulance: 108

Citizen AI
Empowering communities through intelligent analysis and insights

7)<u>API DOCUMENTATION</u>

Endpoints Overview

Method	Endpoint	Description
POST	/chat	Get AI-generated response for civic queries
POST	/feedback	Submit citizen feedback with sentiment info
GET	/status	Check if Citizen AI backend is running

✅ POST /chat

- Description: Generates a civic-related response using the IBM Granite AI model.
- URL: /api/chat

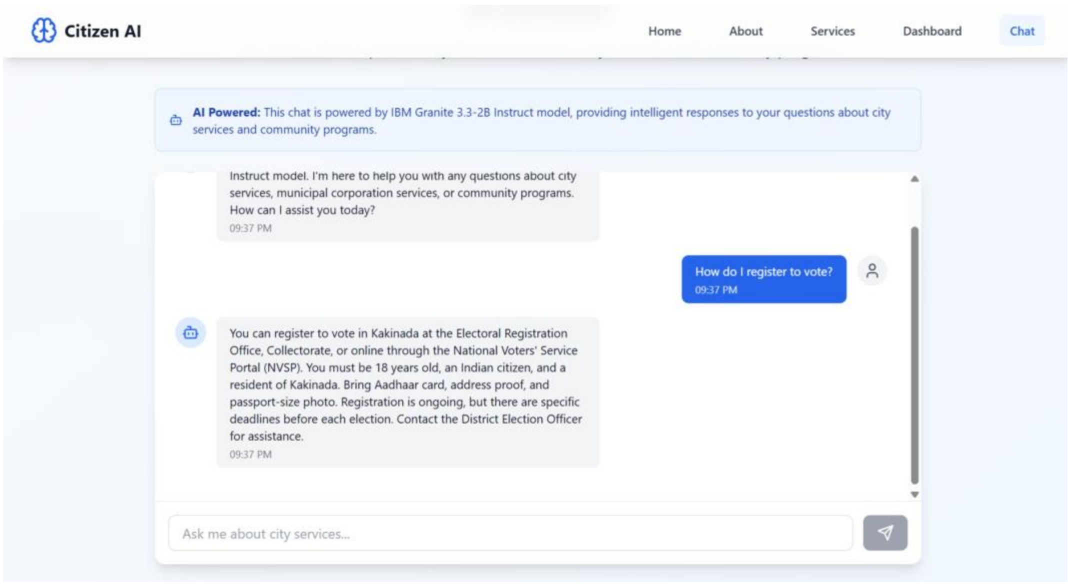
- Method: POST

Request Example:

```
{
  "query": " How do I register to vote? "
}
```

Response Example:

```
{
  "response": " You can register to vote , at the Electoral Registration Office, Collectorate, or online through the National Voters' Service Portal (NVSP). You must be 18 years old, an Indian citizen, and a resident of Kakinada. Bring Aadhaar card, address proof, and passport-size photo. Registration is ongoing, but there are specific deadlines before each election. Contact the District Election Officer for assistance. "
```



AI Powered: This chat is powered by IBM Granite 3.3-2B Instruct model, providing intelligent responses to your questions about city services and community programs.

Instruct model. I'm here to help you with any questions about city services, municipal corporation services, or community programs. How can I assist you today?
09:37 PM

How do I register to vote?
09:37 PM

You can register to vote in Kakinada at the Electoral Registration Office, Collectorate, or online through the National Voters' Service Portal (NVSP). You must be 18 years old, an Indian citizen, and a resident of Kakinada. Bring Aadhaar card, address proof, and passport-size photo. Registration is ongoing, but there are specific deadlines before each election. Contact the District Election Officer for assistance.
09:37 PM

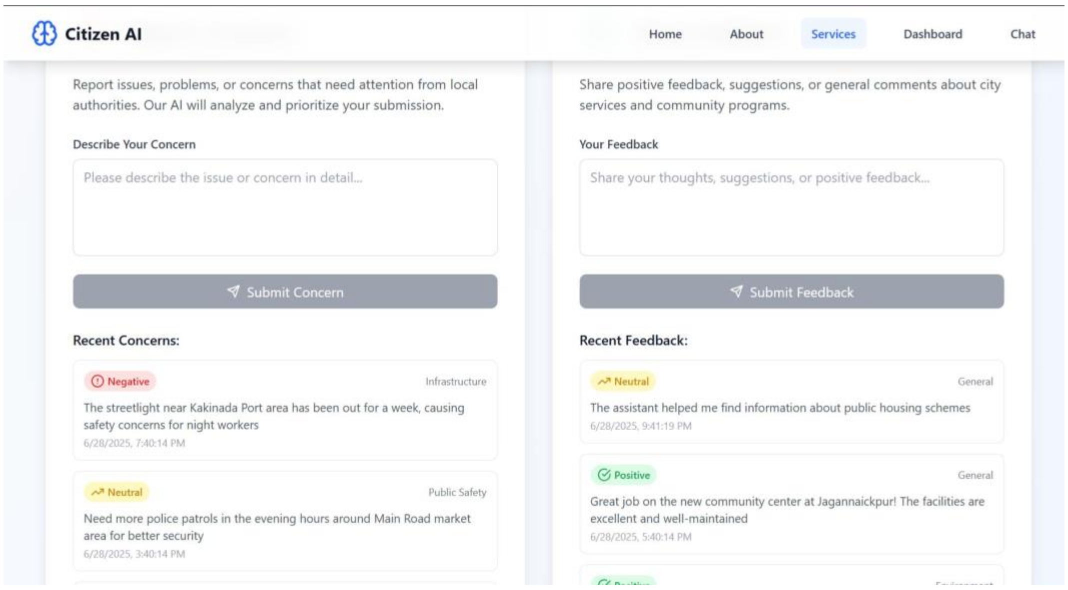
Ask me about city services...

📝 POST /feedback

- Description: Stores user feedback with detected sentiment (positive/neutral/negative).
- URL: /api/feedback
- Method: POST

Request Example:

```
{
  "message": "The assistant helped me find information about public housing schemes.",
  "sentiment": "neutral"
}
```


Citizen AI

Home About Services Dashboard Chat

Report issues, problems, or concerns that need attention from local authorities. Our AI will analyze and prioritize your submission.

Describe Your Concern

Please describe the issue or concern in detail...

Submit Concern

Recent Concerns:

Negative — Infrastructure
The streetlight near Kakinada Port area has been out for a week, causing safety concerns for night workers
6/28/2025, 7:40:14 PM

Neutral — Public Safety
Need more police patrols in the evening hours around Main Road market area for better security
6/28/2025, 3:40:14 PM

Share positive feedback, suggestions, or general comments about city services and community programs.

Your Feedback

Share your thoughts, suggestions, or positive feedback...

Submit Feedback

Recent Feedback:

Neutral — General
The assistant helped me find information about public housing schemes
6/28/2025, 9:41:19 PM

Positive — General
Great job on the new community center at Jagannaickpur! The facilities are excellent and well-maintained
6/28/2025, 5:40:14 PM

Response Example:

```
{
  "status": "success",
  "message": "Feedback recorded. Thank you for your input!"
}
```

- GET /status

- Description: Returns the backend status to confirm it is operational.
- URL: /api/status
- Method: GET

Response Example:

```
{
  "status": "ok",
  "message": "Citizen AI backend is active."
}
```

🧠 Backend Code – Citizen AI (FastAPI + IBM Granite)

```
from fastapi import FastAPI

from pydantic import BaseModel

from transformers import AutoTokenizer, AutoModelForCausalLM

# Initialize FastAPI app
```

```
app = FastAPI(title="Citizen AI API")
# Load IBM Granite Model
model_name = "ibm-granite/granite-3b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)
# Request schemas
class ChatRequest(BaseModel):
    query: str
class FeedbackRequest(BaseModel):
    message: str
    sentiment: str  # "positive", "neutral", or "negative"
# API: Status check
@app.get("/api/status")
def status_check():
    return {"status": "ok", "message": "Citizen AI backend is active."}
# API: Civic query response
@app.post("/api/chat")
def civic_response(request: ChatRequest):
    inputs = tokenizer(request.query, return_tensors="pt")
    outputs = model.generate(**inputs, max_new_tokens=100)
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return {"response": response.strip()}
# API: Feedback submission
@app.post("/api/feedback")
def save_feedback(request: FeedbackRequest):
    # In production, store this in a DB or file
    return {
        "status": "success",
        "message": "Feedback recorded. Thank you for your input!"
    }
```

8)<u>AUTHENTICATION FLOW</u>

1. Basic Authentication (for Prototypes)

- Simple login using a **username and password** stored in memory or a local file.

- Can be implemented directly in **Gradio**, **Flask**, or **FastAPI** backend.
- ⚠️ **Not recommended for production** due to limited security.
- Suitable only for internal demos or testing citizen query flows.

2. OAuth2.0 / Social Login

- Allow users to log in using **Google**, **GitHub**, or **Microsoft** accounts.
- Easily integrated using tools like:
  - **Firebase Authentication**
  - **Google Identity Services**
  - **Auth0**
- ✅ **Secure**, trusted by users, and scalable for civic platforms.

3. Token-Based Authentication (JWT)

- After login, the system returns a **JWT (JSON Web Token)** to the user.
- All protected API requests must include the token in the request header:

http

CopyEdit

```
GET /api/chat

Authorization: Bearer <your_token_here>
```

- Ideal for securing FastAPI-based backends where citizen data or feedback is sensitive.

- ## 🛠️ Tools for Authentication

Tool	Use Case
Firebase Auth	Email/password or social logins like Google
FastAPI OAuth2	Secure backend APIs using token validation
Auth0	Enterprise-grade login system with roles/policies
Gradio login	Basic login screen (limited, only for simple demos)

✅ **Recommended Auth Flow for Citizen AI:**

1. **User opens** the Citizen AI Gradio interface (e.g., gradio.live link or embedded widget).
2. A **login prompt** appears — offering **Google login** or **email/password**.
3. On successful login:
   - Store the **session token** or **JWT** in memory.
   - Grant access to **chat, feedback, sentiment analysis**, and other features.
4. On the **backend (FastAPI)**:
   - Verify the JWT or session token with every request.
   - Only allow access to APIs if the token is valid.

✅ **Why Authentication Matters in Citizen AI**:

- Prevents misuse of the civic query system.
- Enables **personalized responses** based on user role or history.
- Protects feedback and sentiment data submitted by users.
- Ensures compliance with ethical and secure AI deployment standards.

9)<u>TESTING</u>

🧪 **Purpose of Testing – Citizen AI**

To ensure that the **Citizen AI system**:

- ✅ Works correctly across all core features (chat, civic question answering, feedback handling, sentiment analysis)
- ✅ Handles diverse, real-world queries from users (e.g., schemes, rights, public services)
- ✅ Provides accurate, relevant, and ethical responses using AI
- ✅ Maintains stability across edge cases (long queries, unclear questions, empty input)

🧷 **Types of Testing Used or Recommended**

---

1. Unit Testing

- **What it tests:**
  Individual backend functions like:
  - generate_response()
  - analyze_sentiment()
  - validate_query()
- **Tools:** unittest or pytest in Python
- **Example:**

```
def test_generate_response():
    assert "voter ID" in generate_response("How to get a voter ID?")
```

---

2. Integration Testing

- **What it tests:**
  The flow between frontend (Gradio) and backend (IBM Granite model + logic)
- **Goal:**
  Ensure complete interaction works:
  - User input → Model processing → AI-generated response → Display in UI

- **Example:**
  Typing "How to file RTI?" in the Gradio UI returns correct process steps.

---

3. Functional Testing

- **What it tests:**
  That each major **feature behaves as expected** from the user's point of view:
  - Civic chat (basic and complex queries)
  - Feedback submission and sentiment logging
  - Public scheme lookup (if included)
- **Method:**
  Manual testing in Gradio UI or automated functional test cases.

---

4. Edge Case Testing

- **Purpose:**
  Check how the system handles unexpected or extreme inputs, such as:
  - Empty strings
  - Irrelevant text (e.g., "asdfgh" or emojis)
  - Very long or multi-part questions
- **Expected Behavior:**
  Graceful error handling like:

"Please enter a valid civic question to continue."

---

5. Performance Testing (Optional)

- **What it tests:**
  - Average **response time** (e.g., < 3 seconds)
  - Can the system handle **multiple users** simultaneously?
  - Latency in Colab or cloud deployment
- **Usefulness:**
  Helps you prepare for future scaling or integration in public platforms.

🛠 Testing Tools You Can Use

Tool	**Purpose**
pytest	Unit + Integration Testing
Postman	API endpoint testing (if FastAPI used)

Tool	**Purpose**
Gradio UI	Manual testing of full user experience
Google Colab	Live testing in development notebook

10)KNOWN ISSUES

1. Inaccurate or Outdated Responses

- AI may sometimes generate responses that:
    - Refer to outdated laws, schemes, or portals.
    - Misinterpret regional procedures.
- **Reason**: The IBM Granite model relies on pre-trained data and doesn't access real-time government databases.

🛠 *Fix:* Add disclaimers like:

"Please verify this information with your local authorities or official government portals."

---

2. Lack of Local Language Support

- Currently supports **English only** (unless explicitly fine-tuned).
- Non-English or mixed-language queries may return irrelevant answers.

🛠 *Fix:* Add language detection + translation support (e.g., using Google Translate API).

---

3. Generic or Vague Replies

- Some queries like "Tell me about government help" may result in broad or unhelpful responses.

🛠 *Fix:* Implement prompt rewriting or clarifying follow-up logic:

"Do you need help with education, housing, healthcare, or something else?"

---

4. No Real-Time Policy Updates

- AI cannot detect **recent policy changes**, court rulings, or new government initiatives without manual dataset refresh or fine-tuning.

🛠 *Fix:* Integrate periodic script-based updates from gov.in portals or use a hybrid RAG (Retrieval Augmented Generation) system.

Tool	Purpose

5. No Identity Verification / Personalization

- The current prototype does not:
    - Authenticate user identity beyond simple login.
    - Offer personalized scheme eligibility or user-specific history.

🛠 *Fix:* Integrate token-based authentication and connect to user profile DBs in future versions.

---

6. Gradio Session Limits

- On Google Colab, Gradio sessions:
    - Expire after 90 minutes of inactivity.
    - May slow down with too many parallel users.

🛠 *Fix:* Consider migrating to a hosted server (e.g., Render, Hugging Face Spaces, or Firebase backend).

---

7. No Legal or Official Authority

- Citizen AI cannot provide legally binding answers or act as a replacement for government help centers.

🛠 *Fix:* Display a persistent disclaimer:

"Citizen AI is for informational purposes only. For official guidance, consult your local authority.

11)FUTURE ENHANCEMENTS

1. Multilingual Support

- Enable the system to handle regional languages like Hindi, Telugu, Tamil, etc.
- Improves accessibility for users across different states and literacy levels.
- Can be implemented using translation APIs or multilingual fine-tuning.

---

2. Voice Input and Output

- Add speech-to-text and text-to-speech features to support users with visual or literacy challenges.
- Useful for rural or elderly populations who may prefer speaking over typing.

---

3. Scheme Eligibility Checker

- Allow users to enter age, income, and occupation details to get personalized eligibility for government schemes.
- Integrate a rules engine that maps user attributes to active benefits.

---

4. Real-Time Government Data Integration

- Pull live data from official portals (e.g., RTI status, Aadhaar services, election info).
- Keep responses updated using APIs or web scraping (with permissions).

---

5. Feedback Analytics Dashboard

- Visualize common user queries, sentiment trends, and service gaps.
- Helps in reporting, optimization, and even policymaker insights.

---

6. Mobile App Version

- Package the app into an Android/iOS mobile application using Flutter or React Native.
- Expands reach to users without desktop access.

---

7. Offline Chat Support

- Offer limited functionality even in low-connectivity areas using a lightweight local model.
- Important for tribal or remote populations.

---

8. Role-Based Access Control

- Add admin features to review chats, approve suggestions, or train the system with new rules.
- Enable differentiated views for citizens, volunteers, and officials.

---

9. Chat History & Bookmarking

- Allow users to save or revisit previous questions and answers.
- Can be tied to a login system or stored locally in the browser.

---

10. Explainable AI Integration

- Show users *how* or *why* the AI arrived at an answer, improving trust and transparency.

12) <u>APPENDIX</u>

GitHub_Repository_Link: https://github.com/Karthikeya907/CITIZEN-AI..git

GitHub/Project Demo: https://courageous-puffpuff-98fbc9.netlify.app