Citizen AI Assistant

Project Documentation Format

---

1. Introduction

• Project Title: **Citizen AI – Intelligent Citizen Engagement Platform.**

• Team Members:

Team Leader : Darveji Thrisha

Team member : Kanikicharla Chaitanya Naga Lakshmi

Team member : Arikatla Surya Prakash Reddy

Team member : Goriparthi Siva Venkata Sai Anjaneyulu

---

2. Project Overview

• Purpose:
To create an AI-powered civic assistant that allows citizens to raise public issues, get accurate answers to queries, and visualize community sentiment.

• Features:

- AI Chat Assistant
- Predefined civic issue handling
- Sentiment analysis
- Real-time Dashboard
- Login system
- Issue Reporting

---

3. Architecture

• Frontend:
Streamlit wit styled pages using HTML/CSS.

• Backend:
Python functions managing issue checks, Watsonx AI, and data logging.

• **Database:**
JSON files (data.json, issues.json, predefined_issues.json) for storage.

---

4. Setup Instructions

• **Prerequisites:**
Python 3.10+, Streamlit, IBM Watsonx API Key, Git

• **Installation:**

1. git clone https://github.com/chaitanyakanikicharla789/Citizen-ai-assistant
2. cd citizen_ai_assistant
3. pip install -r requirements.txt
4. Create .streamlit/secrets.toml with IBM credentials
5. Run: streamlit run app.py

---

5. Folder Structure

```
CITIZEN_AI_ASSISTANT/
├── app.py                    # Main entry point for Streamlit app
├── ibm_ai.py                 # Handles Watsonx AI model API calls
├── data.json                 # Stores sentiment data and user feedback
├── issues.json               # Stores reported questions and AI responses
├── predefined_issues.json    # Contains predefined issue-response mappings

├── .streamlit/
│   └── secrets.toml          # Contains IBM Watsonx credentials (API Key, Project ID)

├── pages/                    # All sub-pages of the Streamlit app
│   ├── Home.py               # Home page UI
│   ├── About.py              # About the assistant
│   ├── Chat.py               # Chat Assistant interface
│   ├── Dashboard.py          # Dashboard showing sentiment & issues
│   └── Login.py              # Login functionality

├── templates/                # Optional HTML template pages if used
│   ├── index.html
│   ├── about.html
│   ├── chat.html
│   ├── dashboard.html
│   └── login.html

├── static/                   # Static assets like CSS, images
```

```
|   ├── styles.css              # Styling file for UI

├── requirements.txt            # All Python dependencies
├── README.md                   # Project description (optional for GitHub)
└── documentation/ (optional)   # Reports, PDFs, Word files, Screenshots
    ├── Citizen_AI_Documentation.docx
    ├── Citizen_AI_Screenshots/
    └── Project_Report.pdf
```

---

6. Running the Application

• Frontend:

```
streamlit run app.py
```

• Backend:
Handled within the same Streamlit app using Python logic

---

7. API Documentation

Endpoint	Description	Method	Parameters	Response
IBM Watsonx AI	Fallback for queries	POST	prompt, api_key	JSON (AI response)
Sentiment Check	Sentiment of user input	Local	user_input	Sentiment label
Issue Lookup	Find matching issue answer	GET	query	Predefined answer

---

8. Authentication

• Simulated login system with basic credentials
• Stored in JSON or secure config
• No session/token security currently

---

9. User Interface

• Built using Streamlit with CSS styling
• Pages: Home, About, Chat, Dashboard, Login
• Dynamic chat and real-time dashboard updates
(*Screenshots can be inserted manually here*)

---

10. Testing

• Manual input testing for chat, issue reporting, and sentiment recording
• JSON handling tested under concurrent inputs
• Chat tested for AI fallback and predefined matches

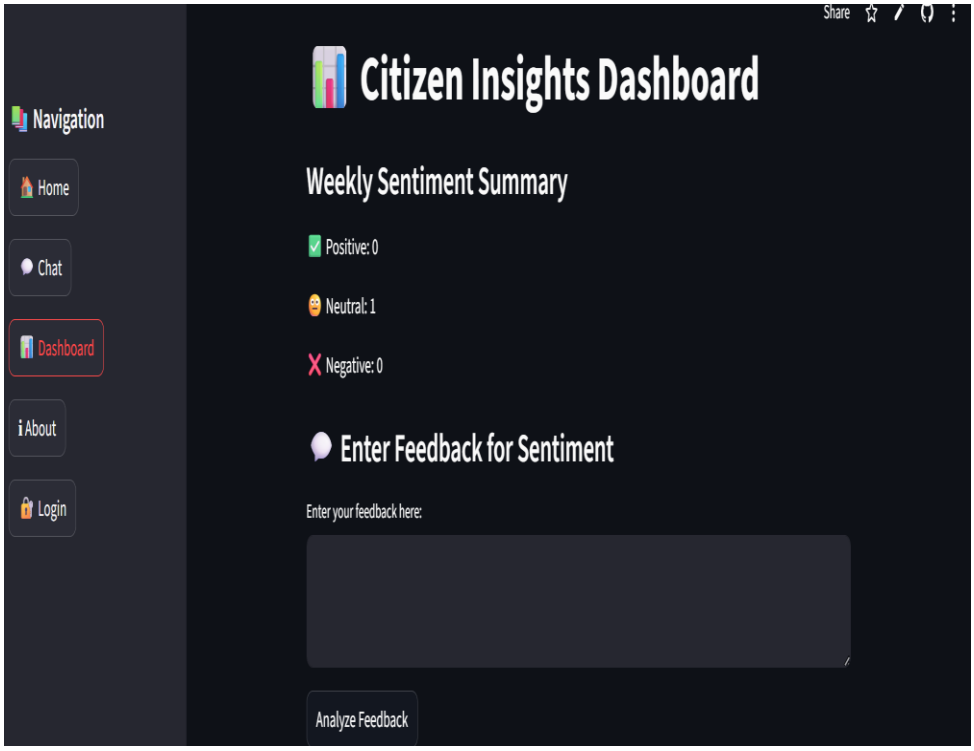
---

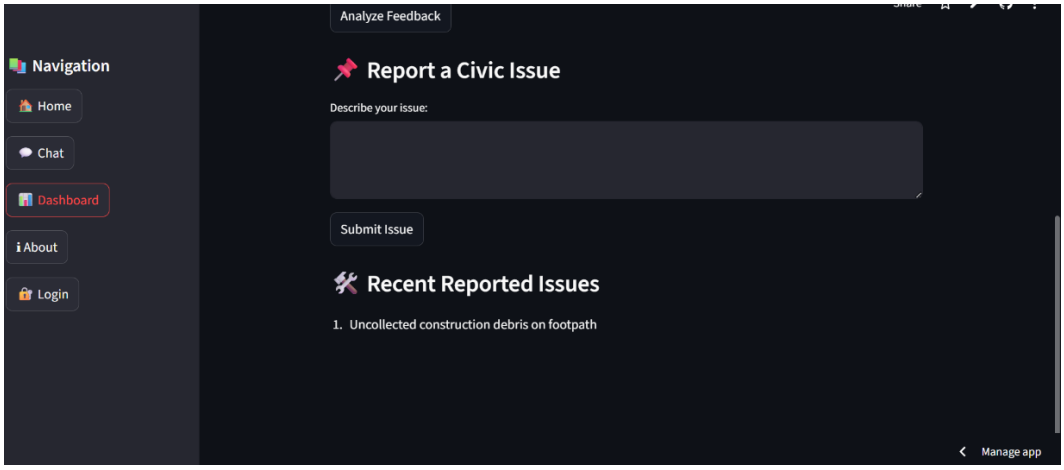
11. Screenshots or Demo

• Include screenshots of:

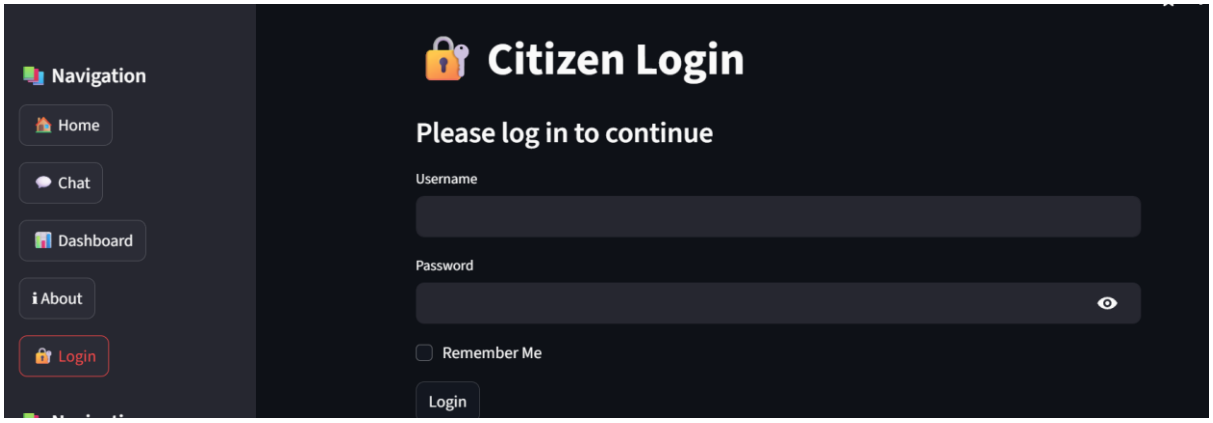
- **Chat interface**:
-

- **Dashboard view:**

- 

- 

- **Login screen:**



• **Demo Link:** https://github.com/chaitanyakanikicharla789/Citizen-ai-assistant

---

12. Known Issues

- No real user authentication
- JSON file-based storage is not scalable
- Requires internet connection for Watsonx API
- Keyword matching can be improved

---

13. Future Enhancements

- Add secure login with sessions or tokens
- Integrate MongoDB or Firebase for scalable data storage
- Add WhatsApp/Voice chatbot version
- Support for regional languages
- Real-time analytics dashboard with maps and trends