PROJECT DOCUMENTATION

CITIZEN AI

TEAM ID : LTVIP2025TMID29997

CitizenAI - Intelligent Citizen Engagement Platform

TEAM ID - LTVIP2025TMID29997

---

1. Introduction

Project Title:

CitizenAI: An Intelligent Citizen Engagement Platform

Team Members:

- Thippugari Rohitha
- M Srinivasa kalyan
- Maddi Varshita Reddy
- Devalla Satwik

---

2. Project Overview

- **Purpose:** The CitizenAI platform is designed to bridge the communication gap between government officials and the public. By leveraging generative AI and data analytics, it provides an accessible, efficient, and insightful channel for citizen engagement. The platform aims to automate query resolution, provide real-time feedback analysis for officials, and foster a more responsive and transparent governance model.
- **Features:** The platform is built around three core, interconnected functionalities:
    1. **AI Chat Assistant for Citizen Queries:** A public-facing chatbot where citizens can ask questions about government services, policies, and local events. The AI, powered by IBM Watsonx, provides instant, accurate answers based on a trusted knowledge base.
    2. **Real-Time Sentiment Tracking:** The platform analyzes citizen feedback and queries from various sources (e.g., chat logs, social media feeds) to determine the overall public sentiment (positive, negative, neutral) on key issues.

[1]. **Analytics Dashboard for Officials:** A secure, private dashboard for government officials. This dashboard visualizes key metrics, including query trends, frequently asked questions,and real-time sentiment analysis, presented through interactive charts and graphs using Plotly.

- **Frontend:**
  - The user interface is built with standard **HTML5** and styled with **CSS** .
  - Dynamic content and data visualizations on the officials' dashboard are rendered using **Plotly.js** , which is integrated directly into the HTML templates.
  - The application uses the **Jinja2** templating engine to dynamically generate HTML pages on the server side.
- **Backend:**
  - The backend server is built using **Flask** , a lightweight and powerful Python web framework. It handles routing, request processing, and business logic.
  - The application is structured into clear routes for different functionalities (e.g., public chat, official's dashboard).
  - It uses **Flask-Login** for managing secure sessions and authenticating officials for access to the analytics dashboard.
- **AI Service & Data Processing:**
  - Natural Language Processing and Generative AI capabilities are powered by **IBM watsonx.ai** .
  - The application uses the ibm-watson and ibm_watsonx_ai Python SDKs to interact with IBM's services.
  - The **IBM Granite** model is used for the chat assistant's conversational logic and for summarizing sentiment data.
  - Data manipulation and aggregation for the analytics dashboard are handled efficiently using the **Pandas** and **NumPy** libraries.

---

[1] . Architecture

This project is implemented as a full-stack web application using a classic and robust Python-based stack.

4. Setup Instructions

- **Prerequisites:**
    1. Python 3.10 or higher
    2. An active IBM Cloud Account
    3. Visual Studio Code (or another code editor)
- **Installation:**
    1. **Clone the Repository:**
    2. Generated bash

```
git clone [your-repository-url]
cd CitizenAI-Platform
```

content_copy

download

Use code with caution .Bash

8. **Create and Activate a Virtual Environment:**
9. Generated bash

```
# Create the environment
python -m venv .venv
# Activate on Windows (PowerShell)
.\.venv\Scripts\activate
```

content_copy

download

Use code with caution .Bash

14.

15. **Install Dependencies:** Install all required Python packages from the requirements.txt file.

16. Generated bash

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

content_copy

18.

download

19.

20. Use code with caution .Bash

21.

22. **Set Up Environment Variables:**

- Create a file named .env in the root project directory.
- Add your IBM Cloud credentials and a secret key for Flask sessions: ■ Generated code

```
IBM_CLOUD_API_KEY= "your_api_key_here"
WATSONX_PROJECT_ID= "your_project_id_here"
WATSONX_URL= "https://us-south.ml.cloud.ibm.com"
■ FLASK_SECRET_KEY= "a_long_random_string_for_security"
```

■ content_copy

■ download

■

■ Use code with caution .

---

5. Folder Structure

- **CitizenAI-Platform/** (Root Directory)
  - **.venv/** : The Python virtual environment folder.
  - **app/** : Contains the core Flask application source code.

- **static/** : Contains static assets like CSS files and images.
- **templates/** : Contains all HTML templates (index.html , dashboard.html , login.html).
- __init__.py : Initializes the Flask application.
- routes.py : Defines all application routes (e.g., / , /dashboard , /login).
- models.py : Defines user models for Flask-Login (e.g., an Official class).
- services.py : Contains business logic for interacting with the IBM Watsonx API and processing data.

- **.env** : Stores secret credentials.
- **requirements.txt** : Lists all Python dependencies.
- **run.py** : The main script to start the Flask application.

---

6. Running the Application

- Ensure your virtual environment is activated.
- From the root project directory, run the following command to start the Flask server:
- Generated bash
- flask run
  content_copy
- download
- Use code with caution .Bash
- Open a web browser and navigate to **http://127.0.0.1:5000** to access the public chat assistant.
- Navigate to **http://127.0.0.1:5000/login** to access the official's login page.

---

7. API Documentation

The backend is a server-rendered application rather than a REST API, but it exposes the following key routes:

- **Route:** /
  - **Method:** GET , POST
  - **Description:** The main public page for the AI Chat Assistant. GET loads the page; POST submits a user's query and returns the page with the AI's response.
- **Route:** /login
  - **Method:** GET , POST
  - **Description:** Displays the login form for officials. Handles login attempts and redirects to the dashboard upon successful authentication.
- **Route:** /dashboard
  - **Method:** GET
  - **Description:** The main analytics dashboard. **Requires authentication.** Displays query trends and sentiment analysis visualizations.
- **Route:** /logout
  - **Method:** GET
  - **Description:** Logs the official out and redirects to the login page.

---

8. Authentication

- Authentication is implemented for the officials' dashboard to protect sensitive analytics data.
- The system uses the **Flask-Login** library to manage user sessions.
- When an official successfully logs in, a secure session cookie is created in their browser. This cookie is used to verify their identity on subsequent requests to protected routes like /dashboard .
- Passwords for official accounts would be stored as secure hashes (e.g., using Werkzeug's security helpers) in a production environment.

---

9. User Interface

The UI is split into two distinct experiences: a simple, welcoming chat interface for citizens and a data-rich, professional dashboard for government officials.

(Insert screenshots of the citizen chat page and the official's dashboard with its Plotly charts.)

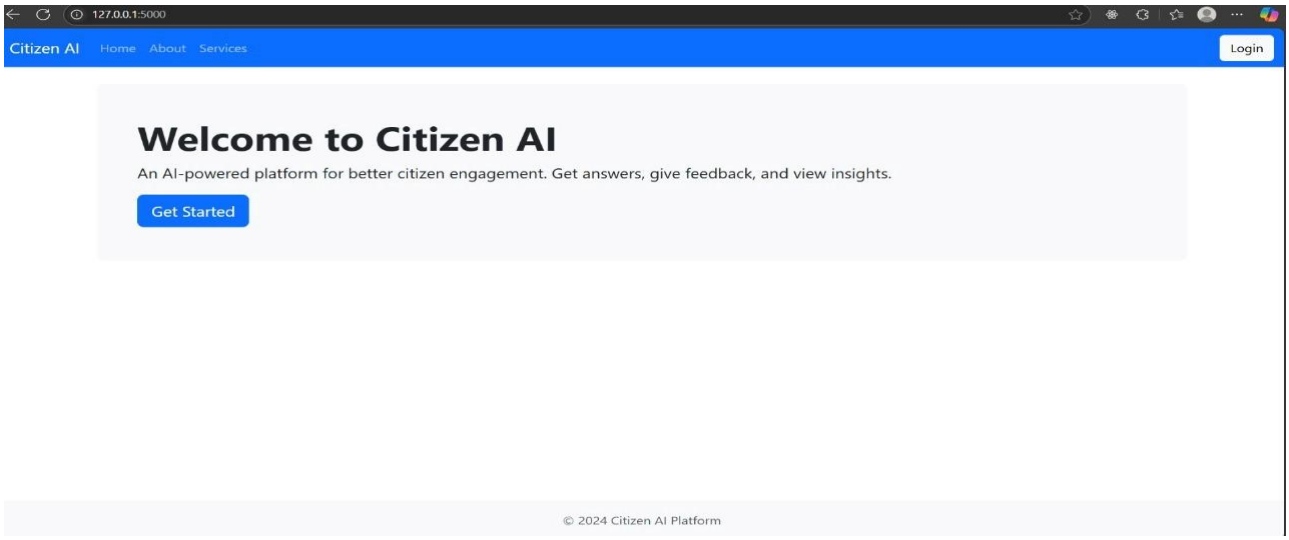
---

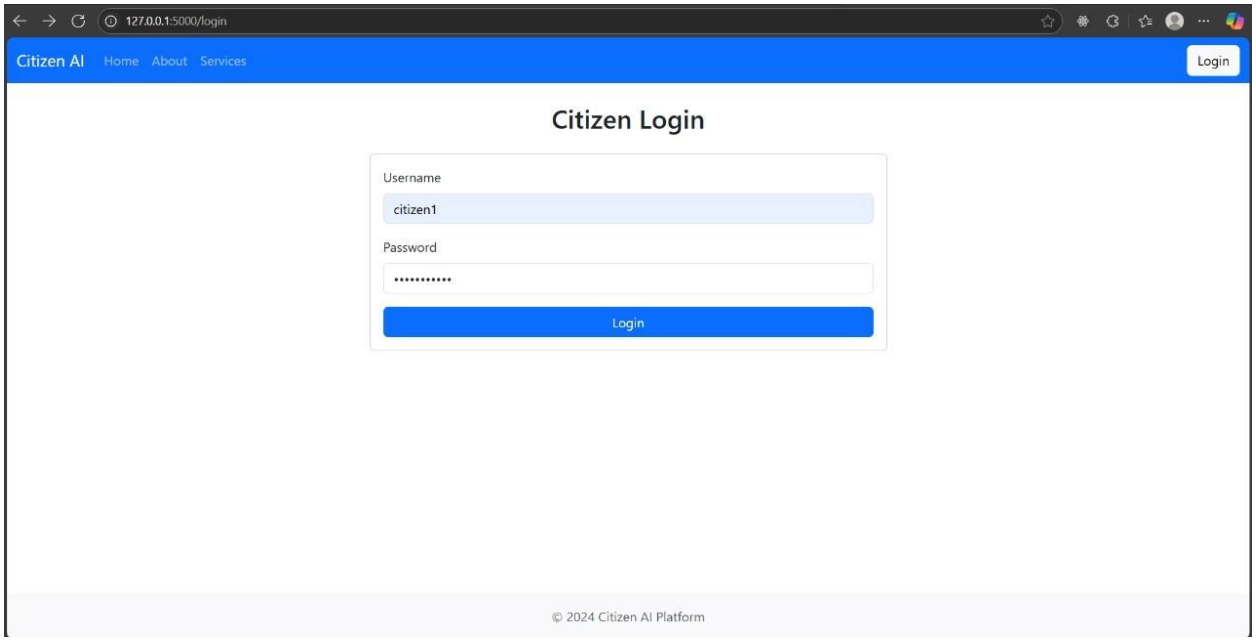
10. Testing

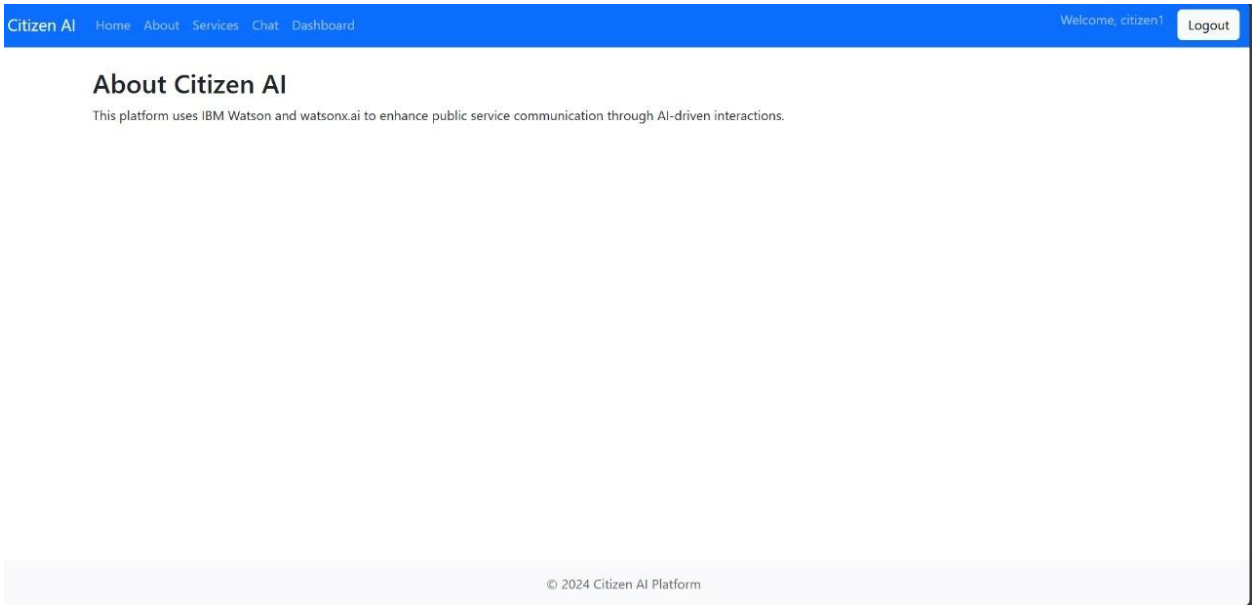
- **Manual Testing:** The application was tested manually by:
    - Interacting with the chatbot using a variety of questions.
    - Testing the login and logout functionality for the official's portal.
    - Verifying that the data and charts on the dashboard accurately reflect user interactions.
    - Ensuring protected routes are inaccessible without a valid login.

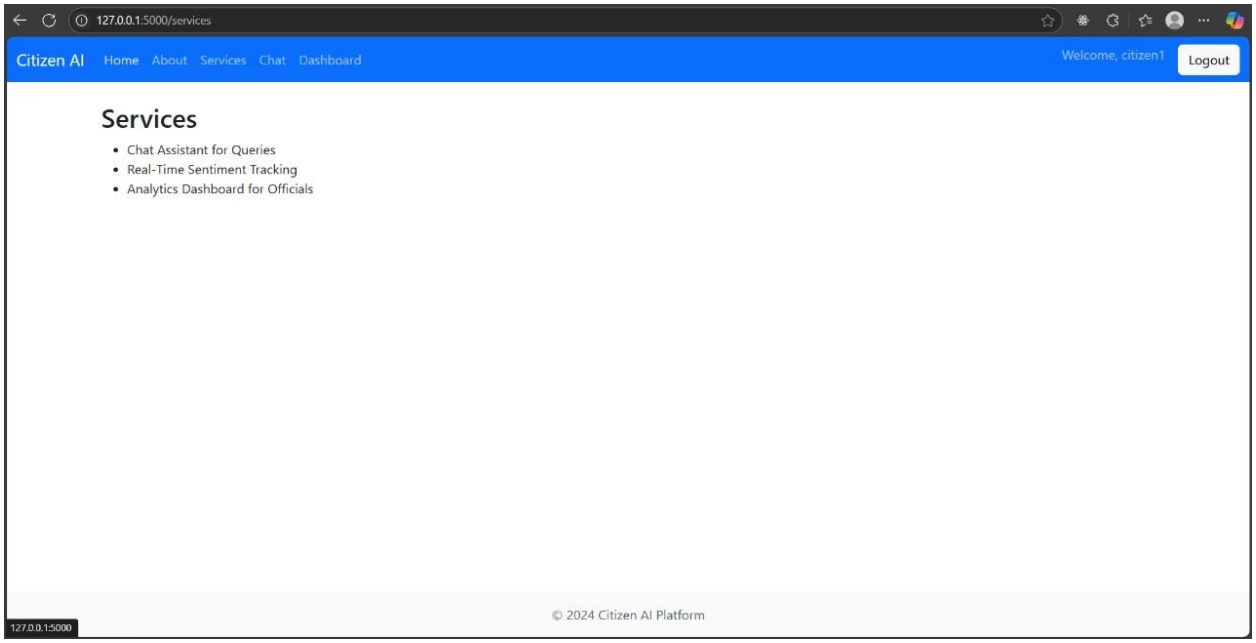
---

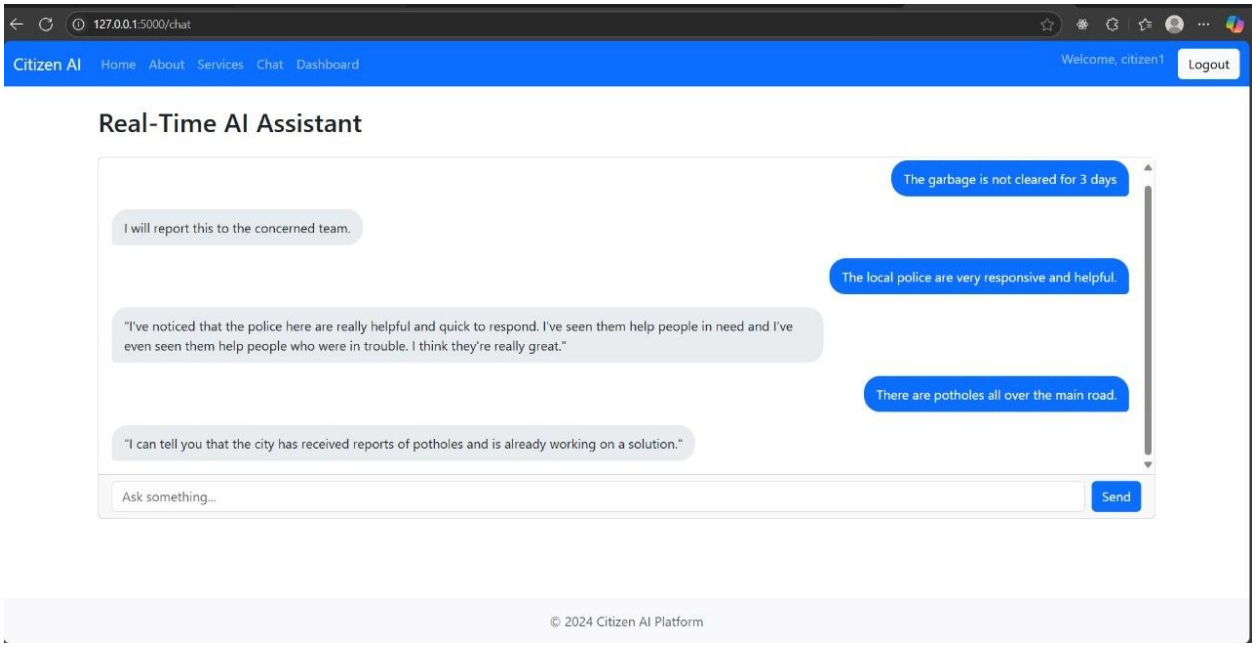
11. Screenshots

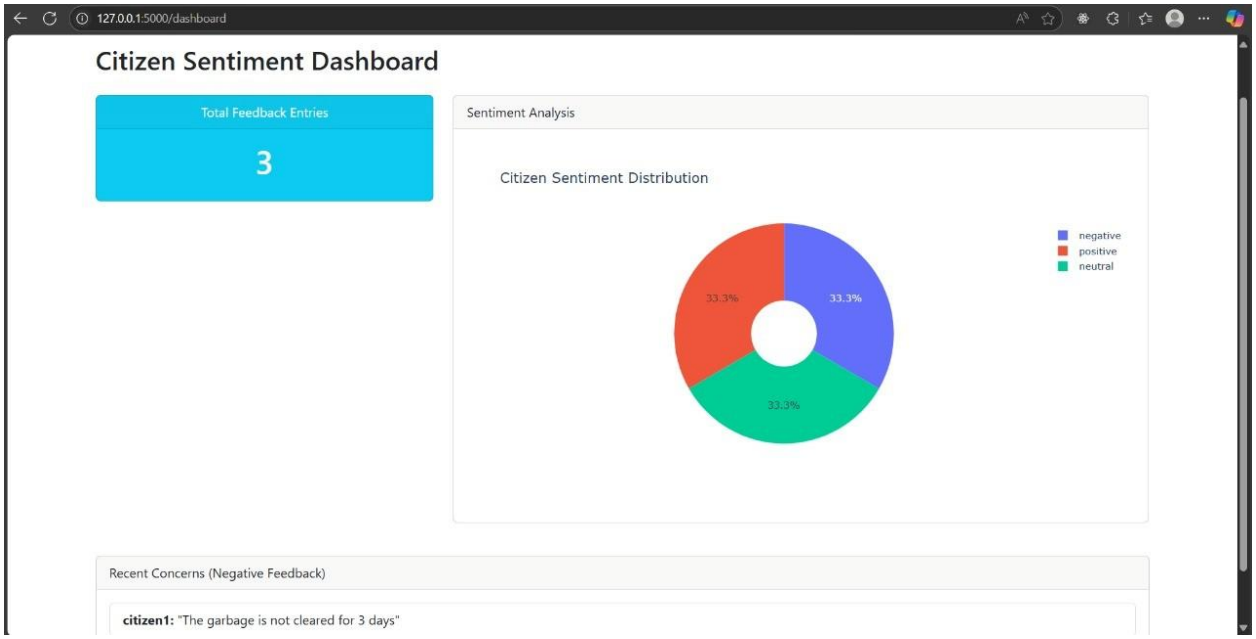



127.0.0.1:5000/login
Citizen AI
Home
About
Services
Login
Citizen Login
Username
citizen1
Password
Login
© 2024 Citizen AI Platform


Citizen AI
Home
About
Services
Chat
Dashboard
Welcome, citizen1
Logout
About Citizen AI
This platform uses IBM Watson and watsonx.ai to enhance public service communication through AI-driven interactions.
© 2024 Citizen AI Platform


127.0.0.1:5000/services
Citizen AI
Home
About
Services
Chat
Dashboard
Welcome, citizen1
Logout
Services
Chat Assistant for Queries
Real-Time Sentiment Tracking
Analytics Dashboard for Officials
© 2024 Citizen AI Platform
127.0.0.1:5000


127.0.0.1:5000/chat
Citizen AI
Home
About
Services
Chat
Dashboard
Welcome, citizen1
Logout
Real-Time AI Assistant
The garbage is not cleared for 3 days
I will report this to the concerned team.
The local police are very responsive and helpful.
"I've noticed that the police here are really helpful and quick to respond. I've seen them help people in need and I've even seen them help people who were in trouble. I think they're really great."
There are potholes all over the main road.
"I can tell you that the city has received reports of potholes and is already working on a solution."
Ask something...
Send
© 2024 Citizen AI Platform



---

12. Known Issues

- **Data Persistence:** The current implementation does not store chat logs or sentiment data in a persistent database. All analytics are based on in-memory data and will be lost upon server restart.
- **Sentiment Analysis Scope:** The sentiment analysis is generalized. It does not yet drill down into specific topics or departments without manual configuration.

---

13. Future Enhancements

- **Database Integration:** Integrate a database (like PostgreSQL or SQLite) to permanently store all citizen interactions, allowing for historical trend analysis.
- **Topic Modeling:** Implement topic modeling on citizen queries to automatically identify emerging issues or key areas of public concern (e.g., "potholes," "park maintenance").
- **Multi-Channel Input:** Expand the platform to ingest and analyze data from public social media feeds (like Twitter/X) or official email inboxes.
- **Automated Reporting:** Add a feature to generate and email weekly or monthly summary reports to officials.

14. License

This project is for educational purposes only.
No part of this code may be used, copied, or distributed without explicit permission from the authors.

15. Contact

For questions or contributions, please contact the team via GitHub issues or [devallasatwik.22.cse@anits.edu.in].