Citizen AI - Intelligent Citizen Engagement Platform

Documentation Format

1. Introduction

Project Title: Citizen AI - Intelligent Citizen Engagement Platform

Team Members and Roles:

- Kalla Venkata Siva Meghana -Final report Developer
- Kalluri Chiranjeevi Collected necessary data
- Bommisetty V K L Vyshnavi Project Planning and Backend developer
- Konithala Nagavidya Desiging and Template Creator

2. Project Overview

Purpose:

The Citizen AI platform aims to streamline citizen management by providing a centralized and intelligent system for accessing information and services. It improves efficiency, transparency, and citizen engagement through a user-friendly interface and automated processes.

Features:

- Citizen Profile Management: Secure storage and management of citizen data.
- Service Request Submission: Online submission of service requests with tracking.
- Automated Notifications: Real-time updates and notifications on service requests and important information.
- Data Analytics and Reporting: Insights into citizen needs and service performance.
- **Secure Authentication:** Secure access to the platform with role-based permissions.

3. Architecture

The Citizen AI platform follows a three-tier architecture:

- **Frontend:** React Provides a responsive and interactive user interface.
- Backend: Node.js/Express.js Handles API requests, business logic, and data processing.
- Database: MongoDB Stores citizen data, service requests, and other relevant information.

4. Setup Instructions

Prerequisites:

- Python
- Flask
- PyTorch
- Hugging Face Libraries
- Sufficient Hardware
- RAM
- GPU (Recommended)
- Internet Connection

Project Structure

Installation Steps:

- 1. Clone the repository:
- 2. Install backend dependencies:
- 3. Install frontend dependencies:
- 4. Set up environment variables. Create files in both and directories.

5. Folder Structure

Client (React):

- src/: Contains React components, pages, hooks, services, and assets.
- public/: Static assets.
- components/: Reusable UI components.
- pages/: Top-level components representing different views.
- services/: API integration logic.
- utils/: Utility functions.
- assets/: Images, icons, etc.

Server (Node.js):

- api-gateway/: API Gateway configuration.
- config/: Global configurations.
- models/: Database schemas.
- routes/: API endpoint definitions.
- controllers/: Business logic.
- middlewares/: Express.js middleware.
- utils/: Server-side utility functions.

6. Running the Application

Frontend:

Navigate to the client directory and run: npm start.

Backend:

http://127.0.01:8000

7. API Documentation

Tooling: API documentation will be generated using tools like **Postman Collections**.

Endpoints: Detailed documentation for all RESTful API endpoints exposed by each microservice, including:

8. Authentication

Upon successful login (email/password, Google, or Facebook), the backend User Management Service generates a JWT.

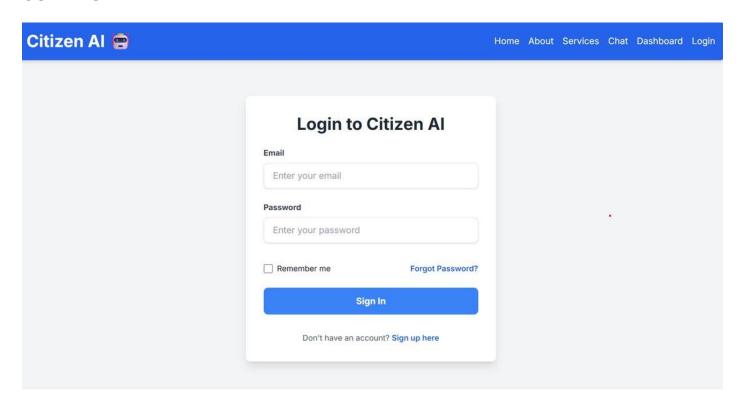
For subsequent API requests, the client includes the JWT in the Authorization header.

9. User Interface

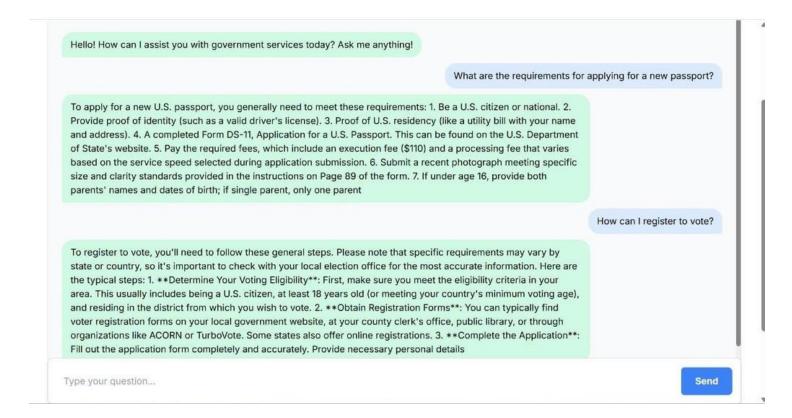
HOME PAGE:



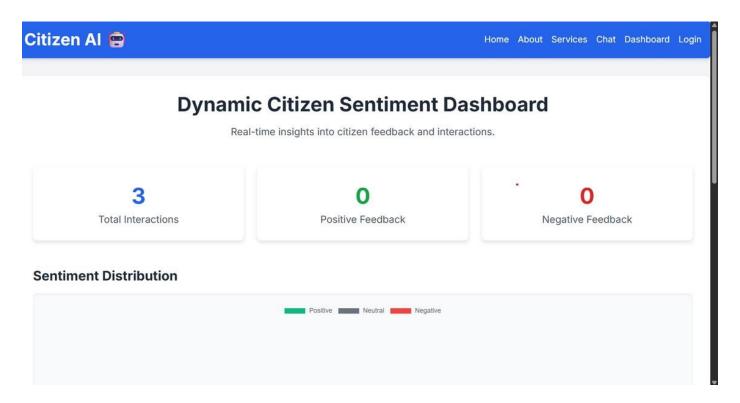
LOGIN PAGE:



CHAT PAGE:



DASHBOARD:



10. Testing

- **Unit Testing:** Testing individual components and functions in isolation.
- Integration Testing: Testing the interaction between different modules and services.
- End-to-End Testing: Testing the entire application workflow from the user's perspective.

11. Screenshots/Demo link

Demo link:

https://drive.google.com/file/d/14Q_BWcLmfW58stA50eefBh3Cf7UrG7s-/view?usp=drive_link

Github link:

https://github.com/Bommisetty-Vyshnavi/Citizen-Al

12. Known Issues

Initial version of the chatbot has limited conversational scope.

13. Future Enhancements

- **Offline Mode:** Enable limited functionality for issue reporting even without an internet connection.
- Implement a more robust notification system (e.g., email, SMS).
- Add support for multiple languages.
- Integrate with third-party services (e.g., payment gateways, mapping services).
- Implement advanced data analytics and reporting features.
- Improve user interface accessibility.