CITIZEN AI-Intelligence Citizen Engagement Platform

Project Documentation format

• Introduction

• **Project Title:** CITIZEN AI-Intelligence Citizen Engagement Platform

• **Team Members:**

Team Members	Role
Chembeti Gnana Prasunamba	Team Leader and Project Manager
Moghal Nikkhath Tabassum	Frontend Developer
K Reshma	Backend Developer
G Uday Kumar	AI Integrator

• Project Overview

Purpose: The purpose of **Citizen AI** is a next-generation AI-powered platform aimed at transforming how governments engage with their citizens. By leveraging **generative AI (IBM Granite models)**, **natural language processing**, and **sentiment analysis**, it allows government bodies to offer real-time, personalized, and context-aware services to the public. Built using **Python (Flask framework)** and supported by IBM's AI ecosystem, Citizen AI promotes transparency, responsiveness, and inclusivity in governance.

Features:

- Improve citizen experience in interacting with government services.
- Provide 24/7 AI-driven responses to citizen queries.
- Monitor public sentiment to inform policy and service delivery.
- Deliver data-driven insights via a dynamic dashboard for officials.
- Enable transparent and real-time civic engagement.

• Architecture

• **Frontend:** The frontend is built using Streamlit (a Python-based framework for UI), structured to display three main features: Requirement Classifier, Bug Fixer, and Code Generator. Each feature has its own input/output panel for user interaction.

• **Backend:** The backend consists of Python FastAPI services organized into AI modules. Each module (classifier, fixer, generator) is independently callable through RESTful endpoints. Communication between Streamlit frontend and backend is handled via API calls.

• **Database:** A simple SQLite database is used to store feedback, logs, and example inputs/outputs. The database is connected to the backend using SQLAlchemy for easy object-relational mapping.

• **Setup Instructions**

• **Prerequisites:**

Python 3.10+

pip (Python package manager)

Ngrok (for public URL)

Streamlit

Git

• **Installation:**

git clone https://github.com/your-repo/smartsdlc.git

cd citizen ai

pip install -r requirements.txt

Set up environment variables (e.g., OpenAI/Hugging Face API keys) in a .env file:

OPENAI_API_KEY= os.getenv("WATSONX_API_KEY")

HUGGINGFACE_TOKEN= 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

• **Folder Structure**

• **Client:**

Citizen_ai_frontend/

├── app.py # Main dashboard

├── pages/ # Subpages for each feature

├── assets/ # Images, logos

└── style.css # Custom styling

- **Server:**

Citizen_ai_backend/

├── main.py # FastAPI app

├── app/

│ ├── models/ # Data models

│ ├── routes/ # API routes for classifier, fixer, generator

│ └── services/ # Core AI logic

└── db/ # SQLite database + init scripts

- **Running the Application**

commands to start the frontend and backend servers locally

o Frontend:

cd Citizen_ai_backend

uvicorn main:app --reload

o Backend:

cd Citizen_ai frontend

streamlit run app.py

- **API Documentation**

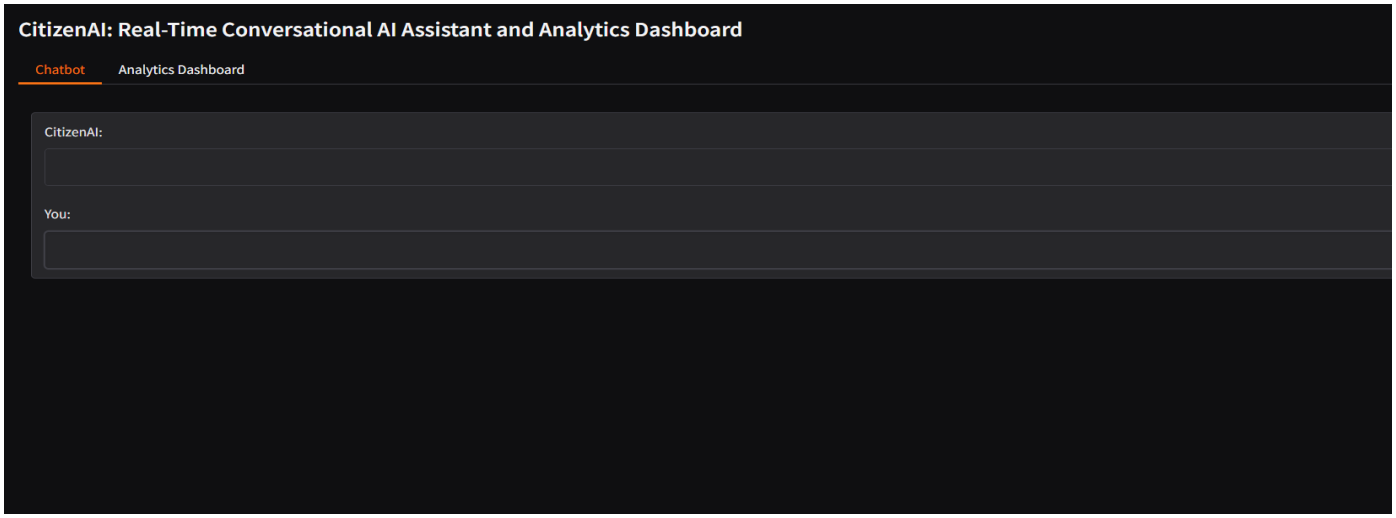
Endpoint	**Method**	**Description**
/classify	POST	Classify input requirements into functional, non-functional, UI
/fix-code	POST	Accept buggy code and return AI-fixed version
/generate- code	POST	Convert natural language requirement into code
/feedback	POST	Submit feedback data
/feedback	GET	View all submitted feedback

- **Authentication**

Token-based authentication (JWT)

Admin access for viewing user feedback and logs

- **User Interface**
  - Simple and clean Streamlit-based UI
  - Three sections: Requirement Classifier, Bug Fixer, and Code Generator
  - Users can:
    - Paste input
    - Click "Submit"
    - View and copy/download results
  - Feedback submission form included for user input
- **Testing**
  - **Tools Used:**
    - Manual Testing via UI
    - Postman for API validation
    - Python unittest for backend services
    - Mock input testing for AI components
  - **Strategy:**
    - Functional testing of each AI module independently
    - End-to-end flow testing from frontend to backend
    - Edge cases: missing input, invalid code, unsupported languages
- **Screenshots or Demo**



Demo link : https://drive.google.com/file/d/1HEaTpsC1L9DVEujrewaP-U3DDPVkctHA/view?usp=drivesdk

- **Known Issues**
  - May fail on highly complex or domain-specific requirements
  - Bug fixer sometimes misses syntax nuances in rare languages
  - Code generation is basic and ideal for simple use cases
  - Ngrok link needs to be refreshed every session (unless premium)
- **Future Enhancements**
  - Add login/authentication with session management
  - Support more programming languages in code generation
  - Enable drag-and-drop file input
  - Connect to cloud database (Firebase/MongoDB Atlas)
  - Add analytics dashboard for admin insights
  - Offline export feature for Citizen ai documents