

# Citizen AI – Intelligent Citizen Engagement Platform

## Project Documentation

---

### 1. Introduction

- **Project Title** : CitizenAI – Intelligent Citizen Engagement Platform
- **Team Member** : Gowtham S
- **Team Member** : Jagadeesh R
- **Team Member** : Francis Samuel
- **Team Member** : Jagan

This project is designed to bridge the communication gap between citizens and government bodies using Artificial Intelligence. It ensures fast query resolution, structured feedback collection, and intelligent dashboards for administrators. By integrating AI-driven natural language processing and sentiment analysis, the platform improves governance, transparency, and citizen satisfaction.

---

### 2. Project Overview

#### **Purpose :**

The purpose of CitizenAI is to provide a smart, AI-driven engagement platform for citizens. It enables smooth interaction with authorities through chatbots, ensures feedback is classified with sentiment analysis, and provides dashboards to officials for decision-making. The platform improves governance by ensuring transparent, data-driven, and accessible citizen services.

#### **Features :**

##### **Conversational Chatbot**

- **Key Point:** Natural language interaction
- **Functionality:** Citizens can ask queries and receive meaningful answers through the chatbot interface.

## Sentiment Analysis

- **Key Point:** Feedback understanding
- **Functionality:** Classifies citizen feedback as Positive, Neutral, or Negative, helping officials measure public mood.

## Issue Reporting

- **Key Point:** Concern logging
- **Functionality:** Citizens can submit issues directly into the system, which are stored and tracked for resolution.

## Analytics Dashboard

- **Key Point:** Data visualization
- **Functionality:** Provides insights into citizen engagement by displaying reported issues and sentiment distributions.

## User Authentication

- **Key Point:** Secure Access
  - **Functionality:** Allows login/logout for citizens and officials to ensure safe and authorized usage.
- 

## 3. Architecture

### Frontend (Flask Templates):

The frontend is built using HTML, CSS, and Bootstrap integrated with Flask templates. It provides user-friendly interfaces for citizens to chat, submit feedback, and view updates.

### Backend (Flask Application):

Flask handles routing, integration with AI, and managing database operations. It ensures smooth communication between the user interface and AI services.

### AI Layer (IBM Granite NLP):

IBM Granite model is used for Natural Language Processing, enabling intent recognition and sentiment analysis to classify queries and feedback.

**Database (SQLite/MySQL):**

Citizen profiles, feedback, and issues are stored securely for analysis. This layer also powers the analytics dashboard.

**Analytics Layer:**

Provides dashboard views for officials to track citizen engagement, sentiment trends, and issue reports.

---

## 4. Setup Instructions

**Prerequisites:**

- Python 3.9 or later
- pip and virtual environment
- Flask and required libraries from requirements.txt
- IBM Granite API key (if using AI mode)

**Installation Process:**

1. Clone the repository from GitHub.
  2. Create and activate a virtual environment.
  3. Install dependencies from requirements.txt.
  4. Create a .env file and configure credentials.
  5. Run the Flask app (app.py or app\_demo.py).
  6. Open the browser at <http://127.0.0.1:5000>.
-

## 5. Folder Structure

Citizen-AI/

- |— app.py – Main Flask application
  - |— app\_demo.py – Demo version
  - |— requirements.txt – Dependencies
  - |— templates/ – HTML files
  - |— static/ – CSS, JS, Images
  - |— screenshots/ – Screenshots
  - |— README.md – Documentation
- 

## 6. Running the Application

To start the project:

- Activate the virtual environment.
  - Install dependencies using pip.
  - Run `python app.py` or `python app_demo.py`.
  - Access via browser at `http://127.0.0.1:5000`.
  - Navigate across Home, Chatbot, Feedback, and Dashboard pages.
- 

## 7. API Documentation

Available endpoints include:

- **POST /chat** – Processes citizen query and returns an AI-generated answer.
- **POST /submit-feedback** – Stores feedback and analyzes sentiment.
- **POST /report-concern** – Logs citizen concerns for administrative review.
- **GET /dashboard** – Fetches analytics for officials.

Each endpoint can be tested using Postman or Flask's built-in test server.

---

## 8. Authentication

CitizenAI currently runs with a simple login/logout system managed by Flask sessions.

For secure deployment, it can be extended with:

- Token-based authentication (JWT)
  - OAuth2 integration with IBM Cloud credentials
  - Role-based access (admin, citizen)
  - Session history tracking
- 

## 9. User Interface

The user interface is designed to be simple and accessible for all citizens. It includes:

- Sidebar navigation for quick access
- Chatbot interface for queries
- Feedback forms for user input
- Dashboard with charts and issue reports

The design is lightweight, responsive, and ensures accessibility across devices.

---

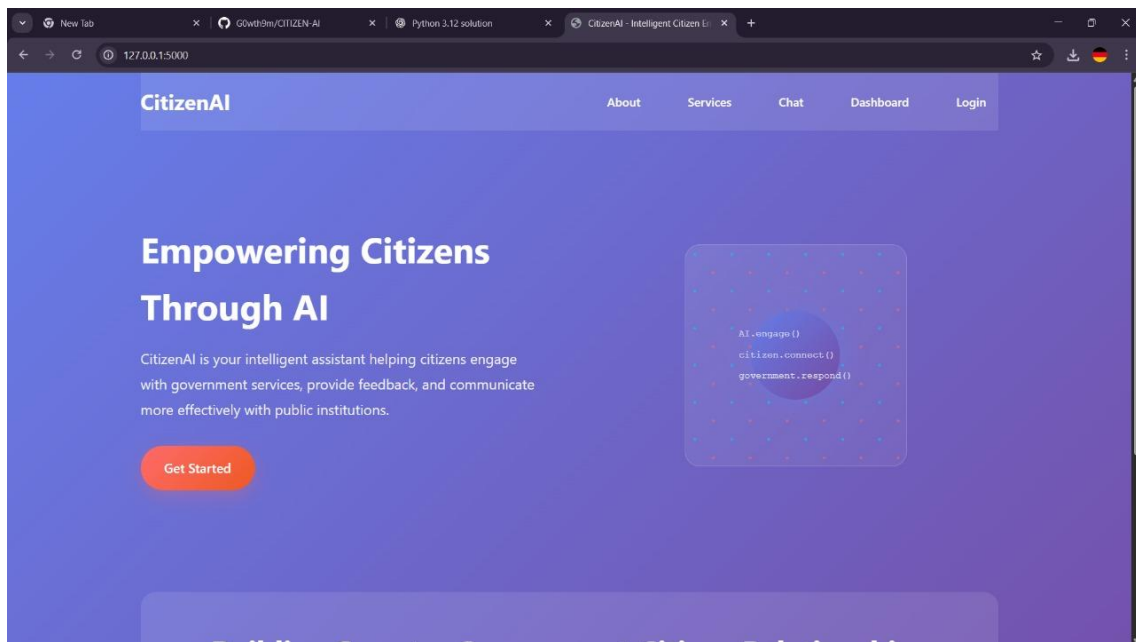
## 10. Testing

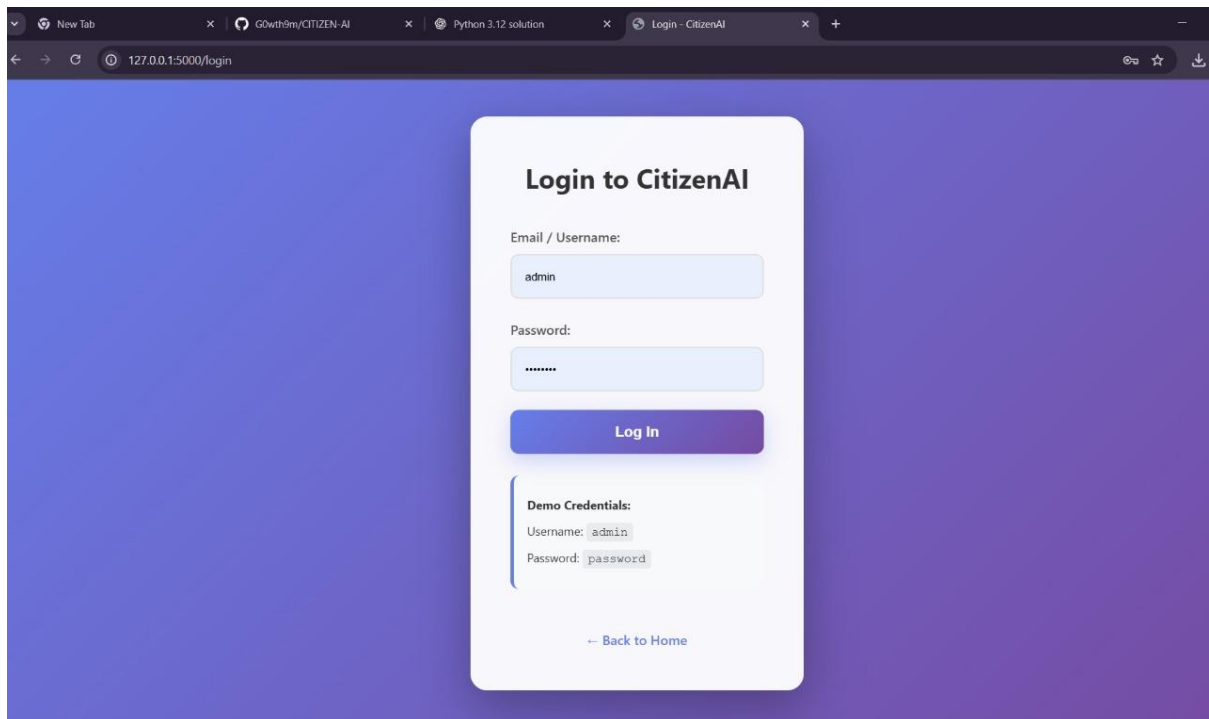
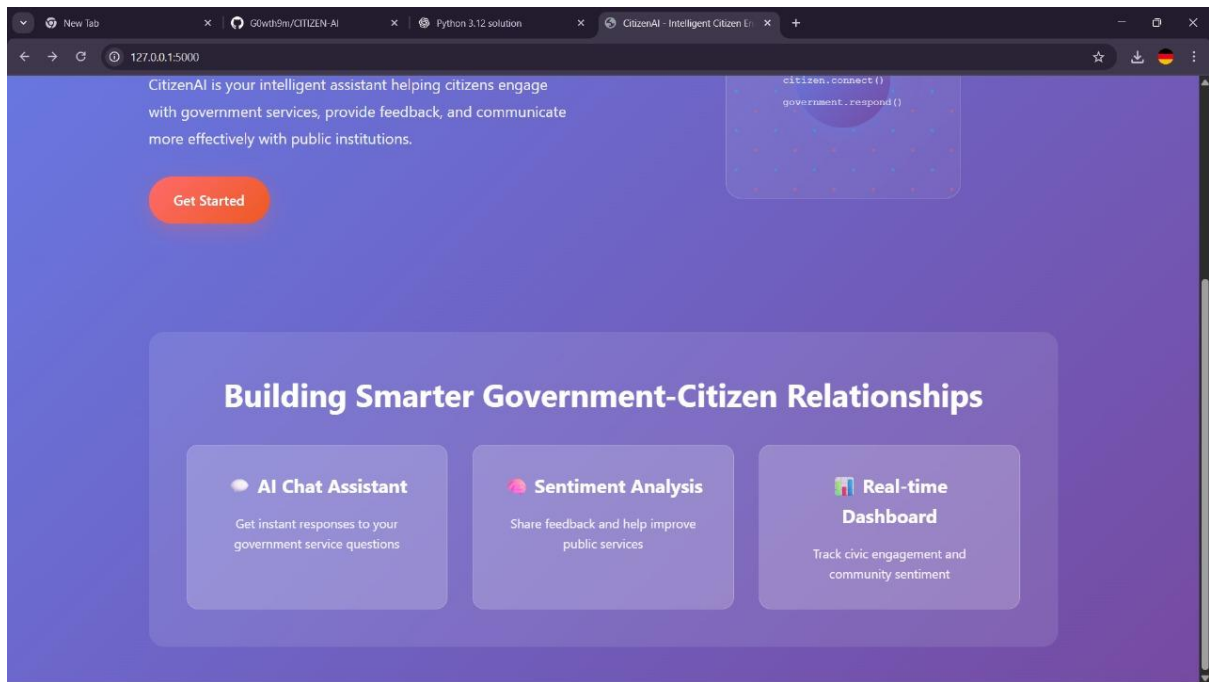
Testing was performed in multiple phases:

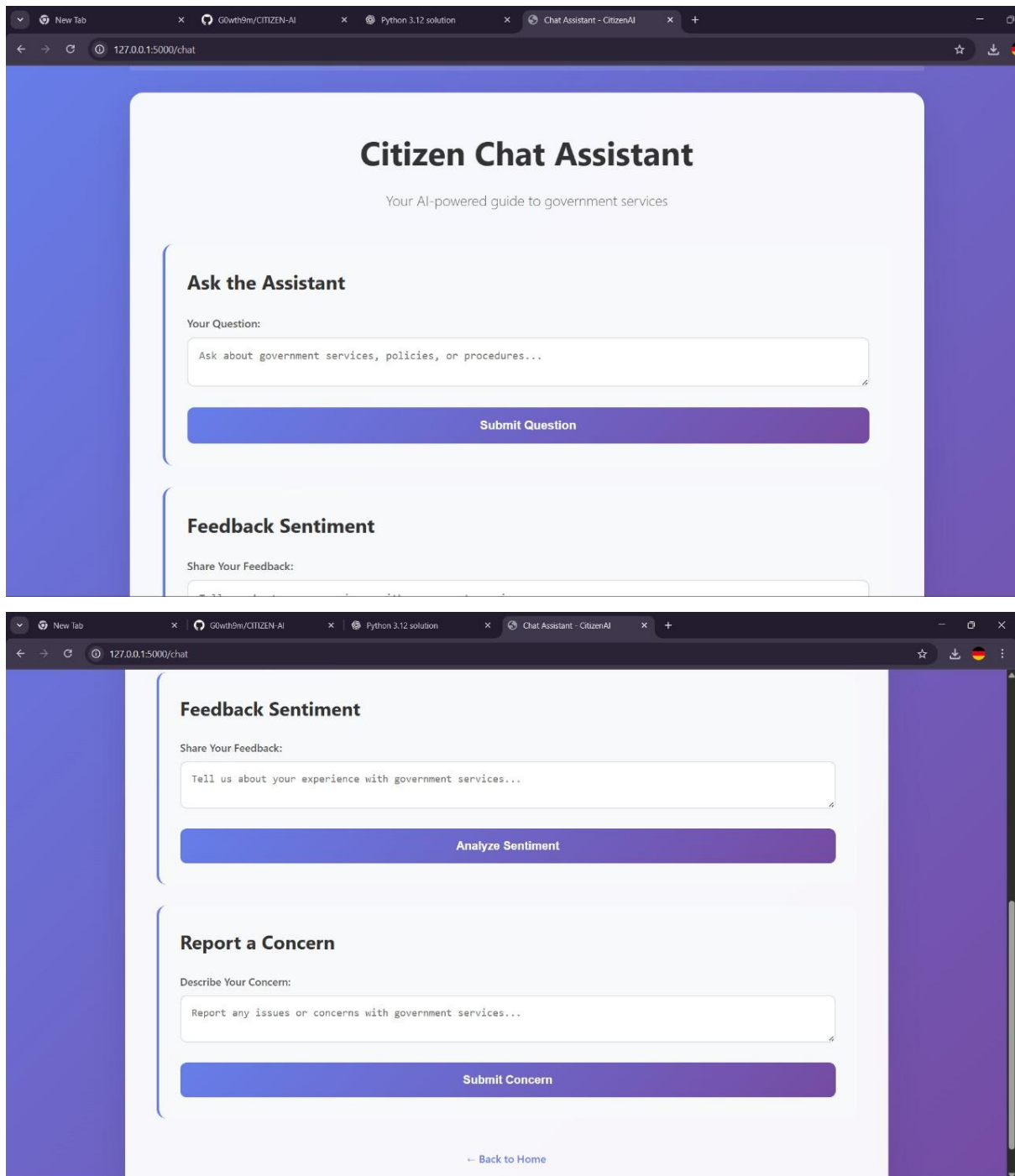
- **Unit Testing:** For chatbot response functions and utilities.
- **API Testing:** Using Postman to validate each endpoint.
- **Manual Testing:** Verified end-to-end flows like citizen queries and feedback logging.
- **Edge Cases:** Tested with empty queries, invalid inputs, and large feedback.

All functions performed reliably under these scenarios.

## 11. Screenshots:







## 12. Known Issues

- Large AI models may impact performance on low-end systems.
  - Demo version currently uses an in-memory database.
  - Limited to text-based input (no voice or multilingual support yet).
-



### **13. Future Enhancements**

Planned improvements for CitizenAI include:

- Voice-enabled chatbot for accessibility.
- Multilingual support for wider reach.
- Advanced role-based admin panel.
- Mobile app integration for real-time use.
- Cloud deployment for scalability and reliability.