Project Title

Citizen AI: Intelligent Citizen Engagement Platform

1. Introduction

* Project Title: Citizen AI: Intelligent Citizen Engagement Platform

* Team member : ANUSRI J

* Team member : APSARA BANU U

* Team member : DURGA PRIYA S

* Team member : JOICE V

2. Project Overview

* Purpose:

Citizen AI is designed to transform how governments and citizens interact by enabling transparent, efficient, and AI-powered citizen engagement. It provides a unified platform where citizens can raise grievances, access policies and services in simplified language, track requests, and give feedback. For officials, it acts as a decision-support system, offering analytics, sentiment insights, and summarizations of citizen concerns to improve governance and service delivery.

* Features:

- Conversational Interface: Allows citizens to ask questions, raise grievances, and get updates in natural language.
- Policy Summarization: Converts lengthy government policies into easy-to-understand summaries.
- Service Request Tracking: Enables citizens to submit and track applications, complaints, and requests.
- Grievance Redressal Support: AI-powered categorization and routing of complaints to relevant departments.
- Citizen Feedback Loop: Collects feedback and analyzes sentiment to guide policy decisions.
- Multilingual Support: Supports multiple languages to ensure inclusivity across regions.
- Analytics Dashboard: Provides insights for officials with trends, sentiment analysis, and KPI tracking.
- Anomaly Detection: Identifies unusual spikes in complaints or service delays for early intervention.
- Document & Data Support: Accepts text, PDFs, and CSVs for analysis and citizen-related queries.
- User-Friendly Interface: Interactive dashboards for citizens and officials with accessibility features.

3. Architecture

Frontend (Streamlit): Provides a user-friendly citizen dashboard with chat, service forms, and updates.

Backend (FastAPI): Handles APIs for citizen queries, grievances, policy search, and analytics.

LLM Integration (IBM Watsonx Granite): Generates summaries, conversational responses, and insights.

Vector Search (Pinecone): Enables semantic search across policy and grievance documents.

ML Modules: Forecast service demands, detect anomalies in grievances, and analyze citizen sentiment.

4. Setup Instructions

Prerequisites:

- Python 3.9+
- pip and virtual environment tools
- API keys for IBM Watsonx and Pinecone
- Internet access for cloud services

Installation:

- Clone repository & install dependencies
- Configure .env with credentials
- Run FastAPI backend
- Launch Streamlit frontend
- Upload documents and interact with modules

5. Folder Structure

app/ – FastAPI backend (chat, feedback, grievance, policy search)

app/api/ – Modular API routes for engagement

ui/ – Streamlit frontend with citizen dashboards

citizen_dashboard.py – Main Streamlit entry point

granite_llm.py – Handles IBM Watsonx communication

document_embedder.py – Embeds citizen-related documents

analytics_reporter.py – Generates AI-based governance insights

grievance_analyzer.py – Routes and tracks complaints

6. Running the Application

- Start FastAPI backend server

- Run Streamlit dashboard

- Navigate via sidebar (chat, grievances, requests, feedback, analytics)

- Upload policies, complaints, or CSVs

- Interact with real-time citizen assistant

7. API Documentation

POST /chat/ask – Citizen Q&A and grievance submission

POST /upload-doc – Upload and embed policy/service documents

GET /search-docs – Semantic search of citizen-facing documents

POST /submit-feedback – Store and analyze citizen feedback

GET /analytics – View sentiment and service KPIs

8. Authentication

- Token-based authentication (JWT or API keys)

- OAuth2 with IBM Cloud credentials

- Role-based access (citizen, official, admin)

- Planned: session management & user history

9. User Interface

The UI is designed for inclusivity and ease of use with:

- Sidebar navigation

- Multilingual chat support

- Complaint and request tracking

- Sentiment dashboards for officials

- Real-time updates and notifications

- Downloadable reports

10. Testing

Unit Testing: For AI response generation and utilities

API Testing: Swagger UI & Postman

Manual Testing: File uploads, service requests, feedback

Edge Cases: Malformed inputs, unsupported formats, invalid credentials

11. Known Issues

* Limited offline functionality

* Dependency on external APIs

12. Future Enhancements

- Advanced predictive analytics for service demand
- Voice-based conversational support
- Expanded multilingual support
- Integration with government ERP systems
- Mobile application version for citizens