#### Citizen Al

## **Project Documentation**

---

### 1. Introduction

Project Title: Citizen Al

Team Leader: Jeevadharani P

Team Members:

Jenova A

Kalaivani S

Kumuthavalli K

Maha Lakshmi M

---

# 2. Project Overview

### Purpose

Citizen AI is designed to empower citizens and local governance through the use of Artificial Intelligence. The platform acts as a digital civic assistant by providing transparent information, quick access to services, grievance redressal, and AI-driven policy insights.

For Citizens  $\rightarrow$  Easy access to government schemes, complaint registration, and service tracking.

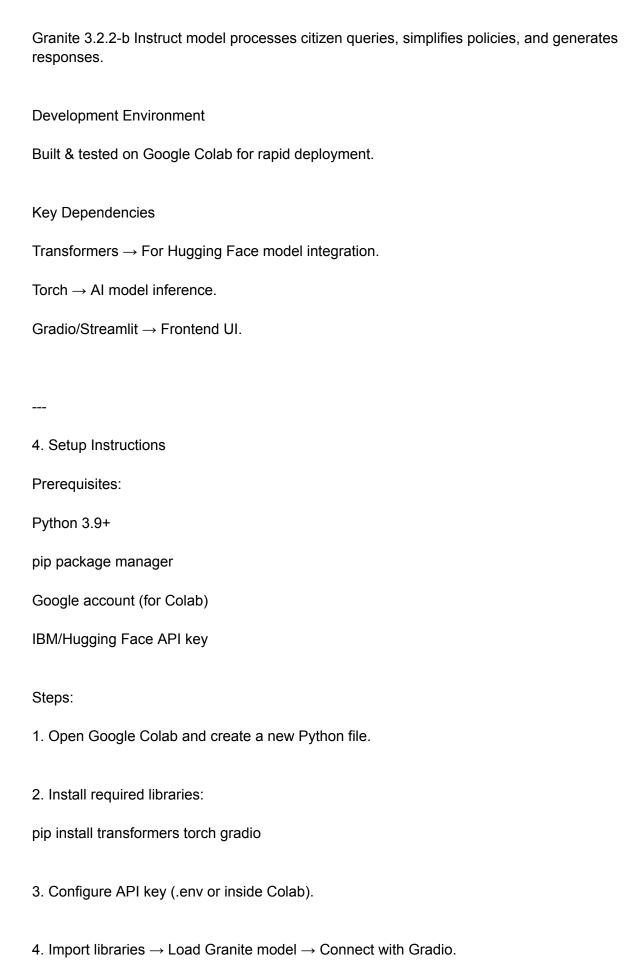
For Government Officials  $\rightarrow$  Al-driven insights into citizen feedback, public demand, and real-time issue tracking.

For Communities  $\rightarrow$  Acts as a bridge for civic engagement, spreading awareness, and promoting active participation.

← Citizen AI combines AI + Governance + Community Engagement to build smarter, transparent, and citizen-friendly societies.

Key Features

Citizen Query Assistant – Al chatbot to answer queries on government services, policies, and schemes.
2. Grievance Management – Citizens can submit issues (e.g., road damage, water supply) and track resolutions.
3. Scheme & Policy Explorer – AI explains government schemes in simple language.
4. Community Dashboard – Real-time updates on local events, resources, and initiatives.
5. Feedback & Suggestions – Collects citizen inputs for government improvement.
<ol> <li>Document &amp; Report Analysis – Summarizes policy documents or announcements for easy understanding.</li> </ol>
7. User-Friendly Interface (Gradio/Streamlit) – Clean, dashboard-style UI for public interaction.
3. Architecture
Frontend (Gradio/Streamlit)
Provides a dashboard for citizen queries, complaints, and updates.
Interactive chat interface + file upload support.
Backend (Hugging Face + FastAPI Layer)
Uses IBM Granite model for language understanding and text generation.
FastAPI ensures modular API handling and scalable design.
LLM Integration



5. Run notebook → Launch Citizen Al dashboard.
6. Citizens can now ask queries, raise complaints, and explore schemes.
5. Folder Structure
CitizenAI/    — CitizenAI.py # Main Colab/Script file    — requirements.txt # Dependencies    — .env # API key configuration    — /utils # Helper functions (prompts, responses)
Frontend → Gradio UI (queries, feedback, complaints).
Backend $ ightarrow$ Granite model integration + response handling.
Config $\rightarrow$ API key setup and environment.
6. Running the Application
1. Open Colab → Create CitizenAl.py.
2. Install dependencies (torch, transformers, gradio).
3. Add API key inside script or .env.
4. Run notebook → Gradio app link generated.
5. Citizens interact via:
Asking about schemes ("Explain PM Kisan Yojana in simple terms").

Filing complaints ("Road damaged in XYZ street").
Getting local updates & suggestions.
7. API Documentation
Citizen AI is built as a Gradio-based app; endpoints are embedded.
Core Functions:
Citizen Query Assistant
Input: Questions about schemes/policies.
Output: Al-generated simple explanation.
Grievance Submission
Input: Complaint text or file upload.
Output: Complaint ID + Al-tracked status.
Document Simplification
Input: Policy document or PDF.
Output: Simplified summary.
8. User Interface
Citizen AI offers a clean, dashboard-style interface with:
1. Citizen Query Chatbot → Ask policy/service-related questions.

- 2. Grievance System  $\rightarrow$  Register complaints & track status.
- 3. Scheme Explorer  $\rightarrow$  Easy-to-understand scheme explanations.
- 4. Community Dashboard  $\rightarrow$  Updates on local issues & initiatives.

← Focused on transparency, accessibility, and engagement between citizens and governance.