**GENERATIVE AI WITH IBM CLOUD**

# 1. Introduction

* **Project Title:** CITIZEN AI-INTELLIGENT CITIZEN ENGAGEMENT PLATFORM
* **Team Members:**
* Team ID: NM2025TMID08924
* Team Size: 5
* Team Leader: Abinaya S
* Team member: ROHINI M
* Team member: ASHVITHA P
* Team member: Keerthana V
* Team member: YUVARANI P

# 2. Project Overview

* **Purpose:**

 **To simplify citizen access to government services**  
– Provide easy, AI-powered assistance to help people understand and use public services.

 **To enable real-time, two-way communication**  
– Allow citizens to ask questions, raise concerns, and receive instant responses through a conversational interface.

 **To improve grievance redressal**  
– Offer a faster, smarter, and more transparent way to submit complaints and track their resolution.

 **To promote inclusive governance**  
– Support multiple languages and accessibility features so that all citizens, including those from remote or marginalized communities, can engage effectively.

 **To analyze public sentiment for better policymaking**  
– Use AI and sentiment analysis to gather insights from citizen feedback to improve government decisions.

 **To increase transparency and trust in governance**  
– Help citizens understand government policies, schemes, and decisions in simple, clear language.

 **To reduce dependency on physical visits and manual processes**  
– Offer digital alternatives to standing in lines or filling out paper forms.

 **To empower citizens with knowledge and tools**  
– Educate users about their rights, entitlements, and responsibilities using personalized AI guidance.

* **Features:**
* **🧠 AI-Powered Virtual Assistant**  
  – Real-time chatbot to answer citizen queries about government services, schemes, and documents.
* **📥 Grievance & Feedback Submission**  
  – Easy-to-use interface to submit complaints, feedback, and suggestions directly to authorities.
* **😊 Sentiment Analysis Engine**  
  – Detects the emotional tone of citizen input to prioritize urgent or sensitive issues.
* **📄 Policy and Scheme Summarization**  
  – Uses AI to convert complex government policies and circulars into simple, understandable summaries.
* **📊 Analytics Dashboard (for Authorities)**  
  – Visual dashboard showing trends in public sentiment, complaint categories, response rates, etc.
* **🌐 Multilingual Support**  
  – Supports major Indian languages and dialects for broader citizen access.
* **📱 Mobile & Web Access**  
  – Fully responsive interface available on smartphones, tablets, and desktops.
* **🔁 Two-Way Communication**  
  – Citizens can ask follow-up questions or interact in a chat-like flow, unlike static forms.
* **🔒 Secure User Data Handling**  
  – Follows privacy protocols and encryption to protect citizen information.
* **📍 Location-Based Alerts**  
  – Sends localized notifications about emergencies, events, or nearby services.
* **🧾 Document Guidance**  
  – Helps citizens understand which documents are needed for specific services or schemes.

# 3. Architecture



# 4. Setup Instructions

**Prerequisites:**

* **Python:** You need a working Python 3.7+ environment installed on your system.
* **Gradio: The Gradio framework is required to application.**
* **Hugging Face Libraries:** The transformers, accelerate, and bitsandbytes libraries are essential for loading and utilizing the IBM Granite model, especially with quantization.
* **Sufficient Hardware:** Running a large language model like IBM Granite

**3.3B requires significant resources. You will need:**

* **RAM:** A substantial amount of RAM (typically 16GB or more is recommended, even with quantization).
* **GPU (Recommended):** A compatible NVIDIA GPU with sufficient VRAM (8GB or more is highly recommended, especially for the 8B model, even with 4-bit quantization) and correctly installed CUDA drivers for reasonable inference speed. Running solely on a CPU will be very slow**.**
* **Internet Connection:** The first time you run the application, the IBM Granite model files will be downloaded from the Hugging Face Hub. You need an active internet connection for this.

# 5. Folder Structure



# 6. Running the Application

# Install Gradio

!pip install gradio

# Import Gradio and define your app

import gradio as gr

def citizen\_ai\_bot(message):

# Example AI response (replace with real model or Hugging Face API)

if "water" in message.lower():

return "Thank you for reporting. The water issue has been recorded."

else:

return "Your query has been submitted. We will get back to you shortly."

# Launch Gradio Interface

iface = gr.ChatInterface(fn=citizen\_ai\_bot, title="Citizen AI – Virtual Assistant")

iface.launch(share=True)

# 7. API Documentation

* Used hugging face model-ibm-granite/granite-3.3-2b-instruct.

# 8.Authentication

By default, **Gradio apps are public**. To implement authentication:

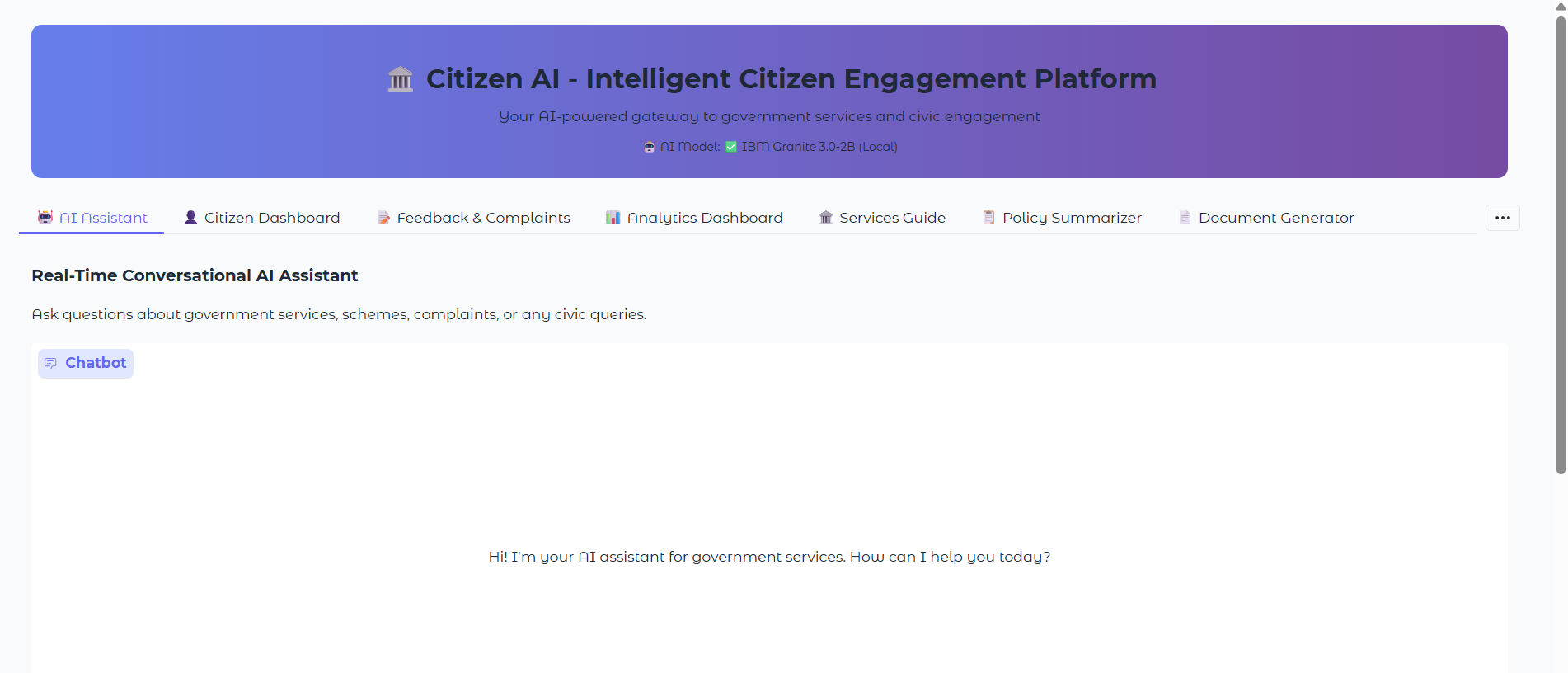
**Basic Authentication**

iface.launch(share=True, auth=("admin", "password123"))

* Prompts user to log in before accessing the app.
* Useful for limited-access testing or admin panels.

1. **User Interface**

* Ai assistant
* Citizen dashboard
* Feedback & complaints
* Analytics dashboard
* Service guide
* Policy summarizer
* Document generater



1. **Testing**

**1. Unit Testing**

* **Objective**: Test individual components like APIs, sentiment analysis modules, chatbot logic, etc.
* **Approach**: Write test cases for each function in the backend (e.g., user input parsing, API call results).
* **Tools**:
  + unittest (Python standard module)
  + pytest for more advanced testing and fixtures

**2. Integration Testing**

* **Objective**: Ensure that different modules (e.g., chatbot + dashboard + database) work together correctly.
* **Approach**: Validate integration of:
  + Hugging Face models
  + FastAPI routes
  + Database interactions (MongoDB/SQLite)
* **Tools**:
  + Postman (for API endpoint testing)
  + pytest with test clients like TestClient from FastAPI

1. **Screenshots or Demo** 
   * Provide screenshots or a link to a demo to showcase the application.
2. **Known Issues**

While *Citizen AI* has undergone multiple rounds of testing, a few known bugs and limitations are documented below to keep developers and users informed. These issues are being tracked and prioritized for future updates.

# 13. Future Enhancements

**1. Mobile App Deployment**

* **Goal**: Make the platform accessible on Android and iOS.
* **Benefit**: Enhance accessibility for rural and less digitally-inclined citizens.
* **Tech Stack**: Flutter / React Native (cross-platform development)

**2. Multilingual Conversational Support**

* **Goal**: Enable real-time voice and text interactions in regional languages (Hindi, Telugu, Tamil, etc.).
* **Benefit**: Inclusivity for non-English speaking users.
* **Tech Stack**: Indic NLP libraries, Whisper API for speech recognition, IBM Granite for translation/context

**3. Aadhaar/UIDAI and DigiLocker Integration**

* **Goal**: Secure identity verification and instant document access.
* **Benefit**: Personalized services, faster grievance processing, automated data entry.

**4. Advanced Analytics for Authorities**

* **Goal**: Generate visual insights for decision-makers (e.g., heatmaps of complaints, service delivery time, sentiment trends).
* **Benefit**: Data-driven governance.
* **Tech Stack**: PowerBI or Streamlit-based analytics dashboard