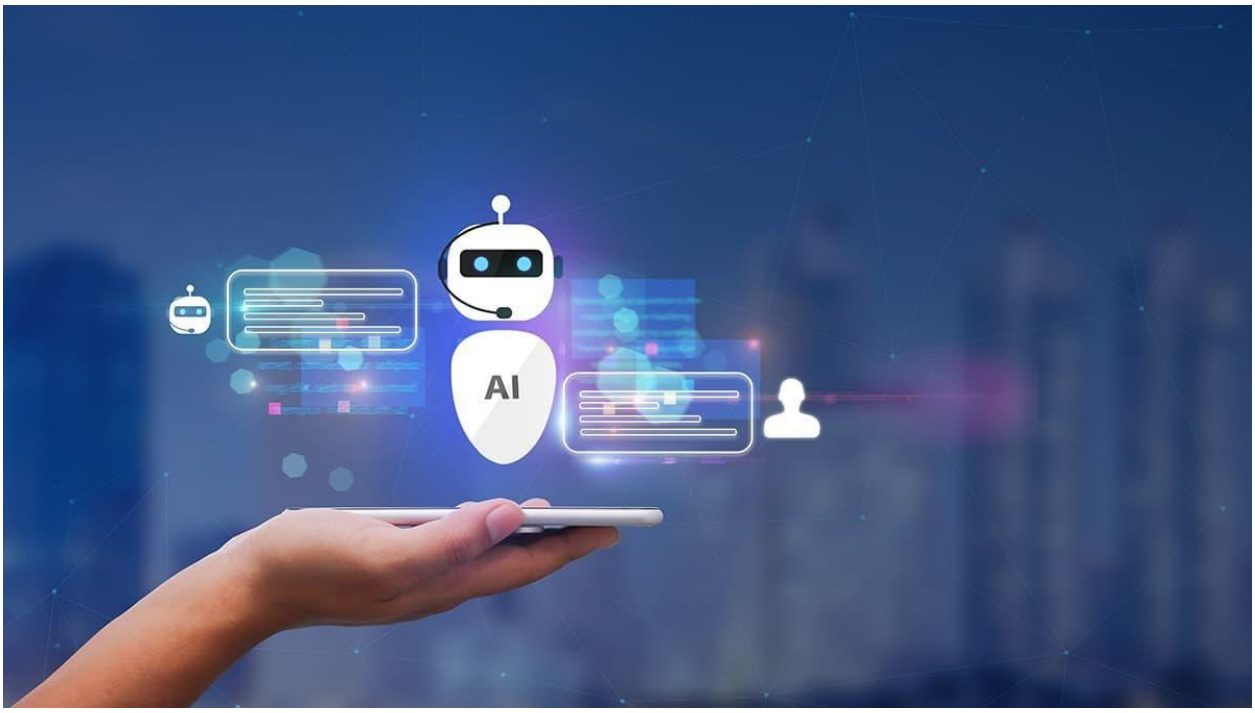
CITIZEN AI: Intelligent Citizen Engagement Platform

1.INTRODUCTION: Citizen AI: Intelligent Citizen Engagement Platform



TEAM LEADER

- **Abdul Fareeth. J**

TEAM MEMBERS

- **M.G AKASH KENNADY**
- **M. AKASH**
- **H. AADHIL**

2.PROJECT OVERVIEW:

The Citizen AI platform is a cutting-edge, AI-driven solution designed to enhance citizen engagement, improve government services, and foster a more responsive and inclusive community. Here's a comprehensive overview

•Conversation Interface

A **conversation interface** in a Citizen AI platform enables citizens to engage with government through natural chat or voice.
It creates an **intelligent engagement system** for faster services, transparent communication, and data-driven governance.

•Policy Summaization

Citizen AI uses policy summarization to present complex rules in **simple, clear language** for citizens.
This makes governance **more transparent, accessible, and easy to understand** for everyone.

•Resources Forecasting

Citizen AI applies **resource forecasting** to predict future needs and demands.
This helps governments ensure **efficient allocation and timely delivery** of public services.

•Eco- Tip Generator

Citizen AI's **eco tip generator** gives citizens daily suggestions to live sustainably. It promotes **green habits and community-wide environmental awareness**.

•Citizen Feedback Loop

Citizen AI's **feedback loop** collects and analyzes public opinions in real time.
This ensures **continuous improvement of services and stronger citizen trust**.

•KPI Fore Casting

Citizen AI uses **KPI forecasting** to predict key performance outcomes of public services. This enables **data-driven planning and proactive governance**.

•Anomaly Detection

Citizen AI's **anomaly detection** spots unusual patterns in citizen data or service use. This helps ensure **early issue detection and quick corrective action**.

•Multimodal Input Support

Citizen AI's **multimodal input support** lets citizens interact via text, voice, or images. It ensures **inclusive, accessible, and user-friendly engagement** for all.

•Streamlit to Gradio Ui

Citizen AI can shift from Streamlit to Gradio UI for more interactive, flexible citizen engagement.

This enables seamless AI demos, faster prototyping, and user-friendly interfaces.

3.ARCHITECTURE:

The **architecture of Citizen AI** integrates data sources, AI models, and citizen-facing interfaces. It ensures **secure, scalable, and efficient intelligent engagement** across services.

•LLM Integration

Citizen AI's **LLM integration** enables natural, human-like conversations with citizens. It provides **context-aware answers and personalized service delivery**.

•Vector Sector

Citizen AI's vector search organizes and retrieves information with high accuracy.

It enables fast, relevant, and context-aware citizen query responses.

•ML Modules

Citizen AI's **ML modules** analyze patterns in citizen data to improve decision-making. They enable **predictive insights and smarter public service delivery**.

4.SETUP INSTRUCTION:

Set up Citizen AI by **deploying AI/ML models with secure data integration** across services. Then configure **multichannel interfaces (chat, voice, web) for citizen interaction**.

•Prerequisites

A citizen engagement platform requires **secure infrastructure and integrated data sources**. It also needs **AI/ML tools with multilingual, accessible user interfaces** for effective interaction.

•Installation Process

The installation process involves **configuring servers, databases, and AI/ML environments**. Next, deploy the **citizen-facing UI with security, multilingual, and accessibility features**.

5.FOLDER STRUCTURE:

The folder structure includes **separate modules for data, models, APIs, and UI components**. It ensures **organized development, easy scaling, and smooth maintenance**.

6.RUNNING THE APPLICATION:

Run the Citizen AI application by **starting backend services and launching the UI interface**.
Citizens can then **interact in real time through chat, voice, or web platforms**.

7.API DOCUMENTATION:

The API documentation provides **detailed endpoints, request/response formats, and authentication methods**.
It enables **developers to integrate and interact with the Citizen AI platform efficiently**.

8.AUTHENTICATION:

Citizen AI uses **secure authentication methods** like OAuth, JWT, or API keys.
This ensures **only authorized users access services and sensitive citizen data**.

9.USER INTERFACE:

The Citizen AI **user interface** offers intuitive chat, voice, and web interactions.
It ensures **easy, accessible, and engaging citizen engagement across platforms**.

10.TESTING:

Testing Citizen AI involves **validating AI responses, system workflows, and data integration**.
It ensures **accuracy, reliability, and seamless citizen interaction**.

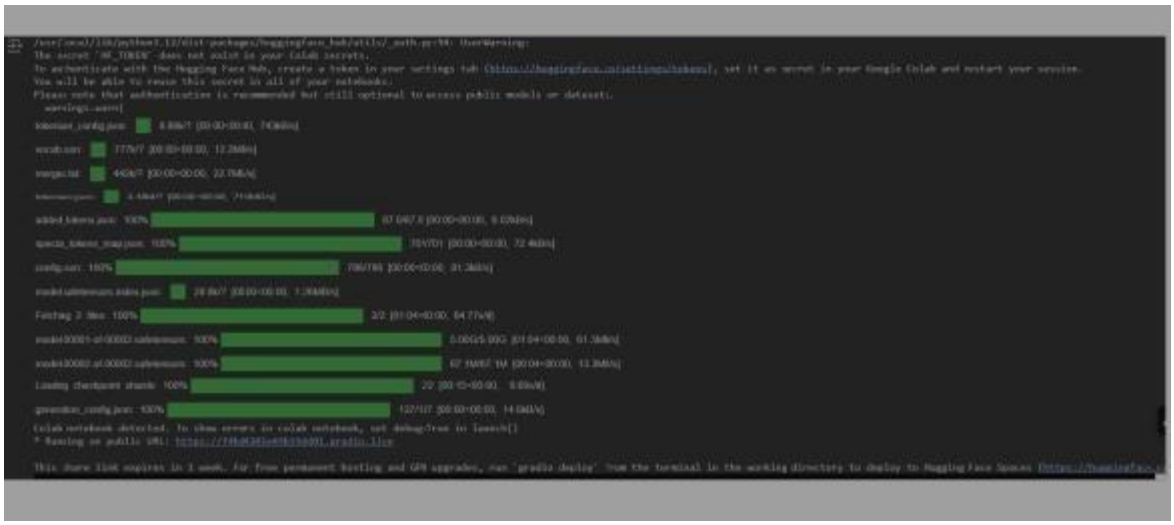
11.KNOWN ISSUES:

Known issues may include **misinterpretation of queries, latency in responses, or integration bugs**.
Addressing them ensures **improved accuracy and smoother citizen engagement**.

12.FUTURE ENHANCEMENT:

Future enhancements include **advanced multilingual support, predictive analytics, and proactive citizen alerts**.
These upgrades aim to **increase engagement, efficiency, and personalized public services**.

13.PROJECT SCREENSHOT:




```python
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=1024):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
            do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )
```

```
    with gr.TabItem("Citizen Services"):
        with gr.Row():
            with gr.Column():
                citizen_query = gr.Textbox(
                    label="Your Query",
                    placeholder="Ask about public services, government policies, civic issues...",
                    lines=4
                )
                query_btn = gr.Button("Get Information")

            with gr.Column():
                citizen_output = gr.Textbox(label="Government Response", lines=15)

        query_btn.click(citizen_interaction, inputs=citizen_query, outputs=citizen_output)

aunch(share=True)
```

THANK YOU