CITIZEN AI – INTELLIGENT CITIZEN ENGAGEMENT PLATFORM

1.introduction

- **Citizen AI – intelligent citizen engagement platform**

Team members:

Anuradha.G

Rekha.A

Kaviyadharshini.S

Swathi.S

Mahalakshmi.M

2.project Overview:

Purpose:

- **Citizen AI is a smart, AI-driven platform designed to enhance communication, participation, and collaboration between citizens and government bodies. The project aims to leverage artificial intelligence, data analytics, and conversational interfaces to make governance more transparent, inclusive, and responsive.**

- **Through Citizen AI, citizens can easily access government services, report civic issues, provide feedback, and participate in decision-making processes. The platform will also help authorities analyze public sentiment, prioritize issues, and respond effectively.**

- **By bridging the gap between citizens and authorities, Citizen AI fosters a more participatory democracy, improves service delivery, and builds trust in governance systems.**

conversational interface:

- **It supports both text and voice communication. Provides instant access to government services and information Allows citizens to report issues and track their status. Available 24/7 and supports multiple languages for easy engagement.**

Policy Summarization:

- **“Policy summarization uses AI to automatically extract and present the key points of policies or documents, making them easier for citizens to understand quickly.”**

Resource Forecasting:

- **“Resource forecasting uses AI to predict future needs of personnel, materials, or services, helping authorities plan efficiently and avoid shortages or overuse.”**

Eco-Tip Generator:

- **“An AI-powered eco-tip generator provides personalized suggestions to citizens on how to adopt environmentally friendly habits in daily life.”**

Citizen Feedback Loop:

- **“A citizen feedback loop collects opinions, suggestions, and complaints from citizens, analyzes them using AI, and helps authorities improve services continuously.”**

KPI Forecasting:

- **“KPI forecasting uses AI to predict future performance metrics, helping organizations track progress, set targets, and make data-driven decisions.”**

Anomaly Detection:

- **“Anomaly detection uses AI to identify unusual patterns or outliers in data, helping spot errors, fraud, or unexpected events quickly.”**

Multimodal Input Support:

- **“Multimodal input support allows users to interact with the platform using multiple forms of input—like text, voice, and images—for a more flexible and intuitive experience.”**

Streamlit or Gradio UI:

- **“Streamlit and Gradio provide easy-to-build, interactive web interfaces for AI applications, allowing users to test and use features without coding.”**

3. Architecture

Frontend (Streamlit):

- **Streamlit is used to build the web-based user interface.**
- **Provides a simple, interactive dashboard for citizens to interact with AI features.**
- **Supports conversational interface, policy summarization, feedback submission, and eco-tip generator.**
- **Ensures user-friendly design with text, voice, and image input (multimodal support).**
- **Displays results like summaries, forecasts, and insights in a clean, visual format.**

Backend (Fast API):

- **Handles API requests between frontend and AI models.**
- **Manages user data and service integration.**
- **Provides fast and secure communication.**
- **Ensures scalability and reliability.**

LLM Integration (IBM wastsonx Granite):

- **Integrates IBM Watsonx Granite LLM for advanced AI capabilities.**
- **Powers conversational interface, summarization, and forecasting.**
- **Ensures accurate, context-aware responses.**
- **Scales easily for large citizen interactions.**

Vector Search (Pinecone):

- **Stores embeddings of policies, FAQs, and citizen data.**
- **Enables fast, semantic search for relevant information.**
- **Improves accuracy of answers in conversational interface.**
- **Scales to handle large volumes of citizen queries.**

ML Modules (Forecasting & Anomaly Detection):

- **Forecasting: Predicts future trends, resources, and KPIs.**
- **Anomaly Detection: Identifies unusual patterns, fraud, or errors.**
- **Supports data-driven decision-making.**
- **Enhances efficiency and reliability of citizen services.**

4. Setup Instructions:

Prerequisites:

- **Python 3.9+ installed**
- **Pip / conda for package management**
- **Streamlit (for frontend)**
- **FastAPI (for backend)**
- **IBM Watsonx API access (Granite LLM)**
- **Pinecone account & API key (for vector search)**
- **Required Python libraries: pandas, numpy, scikit-learn, transformers, etc.**
- **Git & IDE (VS Code / PyCharm)**

Installation Process:

- **Clone the project repository.**
- **Create and activate a virtual environment.**
- **Install all required dependencies from requirements.**
- **Configure environment variables (API keys for Watsonx and Pinecone).**
- **Start the backend server with FastAPI.**
- **Launch the frontend interface with Streamlit.**

5.Folder structure :

- **Frontend – Streamlit UI for citizen interaction**
- **Backend – FastAPI backend with routes and services**
- **Ml_modules – Forecasting and anomaly detection models**

- **Models – LLM integration (IBM Watsonx Granite**
- **Vectored – Pinecone vector search handler**
- **Data – Sample datasets and policy documents**
- **Requirements.txt – Project dependencies**
- **README.md – Project overview and setup guide**
- **env.efolder structure in words only (no code blocks):**
- **frontend – Streamlit UI for citizen interaction**
- **backend – FastAPI backend with routes and services**
- **ml_modules – Forecasting and anomaly detection models**
- **models – LLM integration (IBM Watsonx Granite)**
- **vector_db – Pinecone vector search handler**
- **data – Sample datasets and policy documents**

- **requirements.txt – Project dependencies**
- **README.md – Project overview and setup guide**
- **env.example – Environment variables (API keys)**

6.Running the Application:

To Start the Project:

- **Activate the virtual environment.**
- **Run the backend server with FastAPI.**
- **Launch the frontend with Streamlit.**
- **Open the Streamlit dashboard in a browser to use AI features.**

Frontend (Streamlit):

- **“Streamlit provides a simple, interactive web interface for users to access AI features like chat, policy summarization, forecasts, and eco-tips.”**

Backend (Fast API):

- **“FastAPI handles requests between the frontend and AI models, manages data securely, and ensures fast, reliable performance.”**

7.API Documentation:

Backend APIs Available:

- **/chat – Handles conversational queries from citizens.**
- **/summarize – Summarizes policy documents or text inputs.**
- **/forecast – Provides resource and KPI forecasts.**
- **/anomaly – Detects anomalies in data.**
- **/eco-tip – Generates personalized environmental tips.**
- **/feedback – Collects and manages citizen feedback.**

8.Authentication:

- **“Authentication ensures that only authorized users can access the platform, using API keys, tokens, or login credentials for secure interactions.”**

9.User Interface:

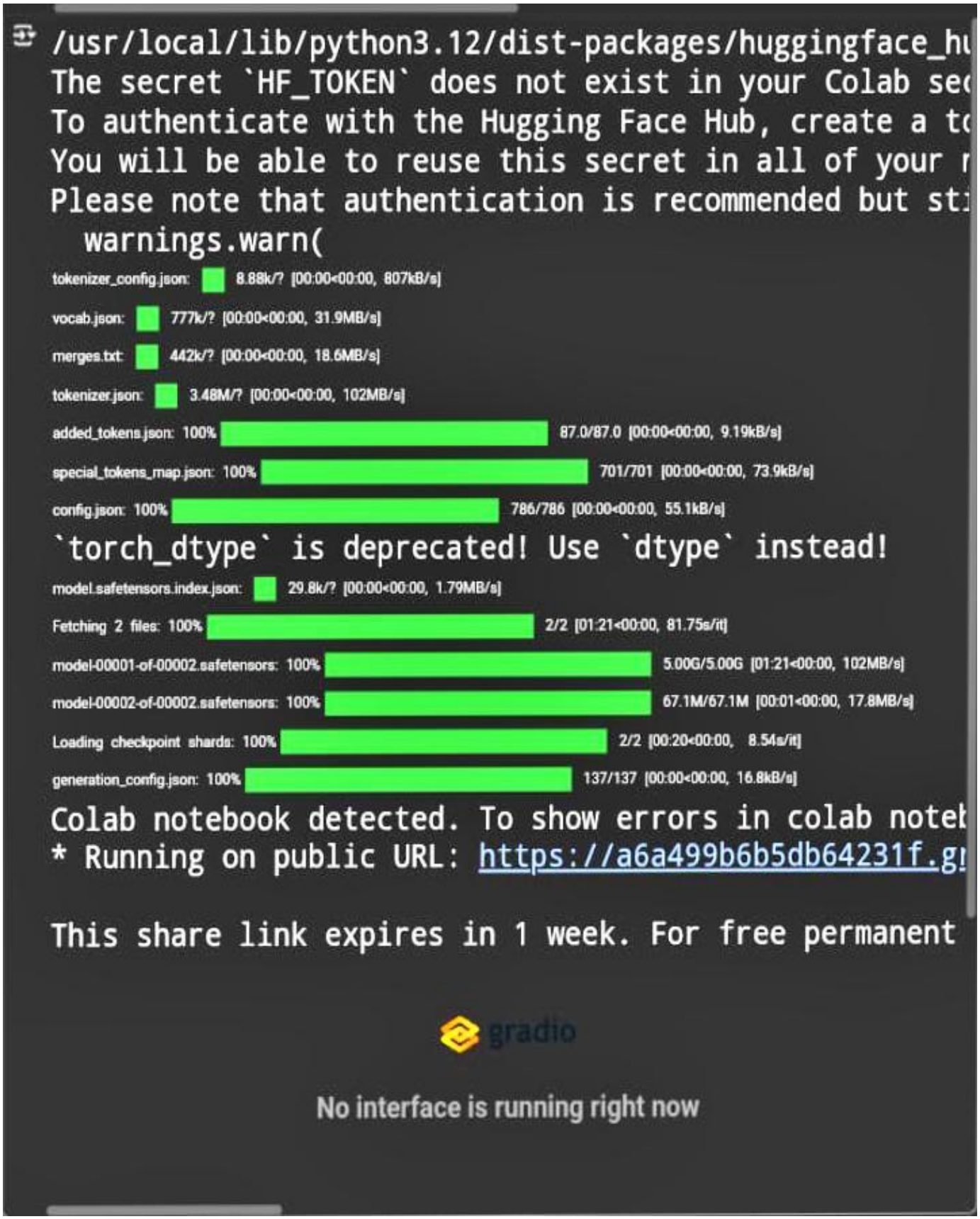
- **“The user interface, built with Streamlit, is intuitive and interactive, allowing citizens to chat, submit feedback, view summaries, forecasts, and eco-tips easily.”**

10. Testing:

- **“Testing ensures that all features of the Citizen AI platform—chat, policy summarization, forecasting, anomaly detection, eco-tips, and feedback—work correctly, reliably, and efficiently.”**

11.Screen shots:

```
# run this project file in google collab by changing run type to T4 GPU

!pip install transformers torch gradio -q

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
        )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.replace(prompt, "").strip()
    return response

def city_analysis(city_name):
    prompt = f"Provide a detailed analysis of {city_name} including:\n1. Crime Index and safety s
    return generate_response(prompt, max_length=1000)

def citizen_interaction(query):
    prompt = f"As a government assistant, provide accurate and helpful information about the foll
    return generate_response(prompt, max_length=1000)

# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# City Analysis & Citizen Services AI")

    with gr.Tabs():
        with gr.TabItem("City Analysis"):
            with gr.Row():
                with gr.Column():
                    city_input = gr.Textbox(
                        label="Enter City Name",
                        placeholder="e.g., New York, London, Mumbai...",
                        lines=1
                    )
                    analyze_btn = gr.Button("Analyze City")

                with gr.Column():
                    city_output = gr.Textbox(label="City Analysis (Crime Index & Accidents)", lin

            analyze_btn.click(city_analysis, inputs=city_input, outputs=city_output)

        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)

app.launch(share=True)
```


/usr/local/lib/python3.12/dist-packages/huggingface_hu
The secret `HF_TOKEN` does not exist in your Colab sec
To authenticate with the Hugging Face Hub, create a to
You will be able to reuse this secret in all of your n
Please note that authentication is recommended but sti
warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 807kB/s]
vocab.json: 777k/? [00:00<00:00, 31.9MB/s]
merges.txt: 442k/? [00:00<00:00, 18.6MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 102MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 9.19kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 73.9kB/s]
config.json: 100% 786/786 [00:00<00:00, 55.1kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.79MB/s]
Fetching 2 files: 100% 2/2 [01:21<00:00, 81.75s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:21<00:00, 102MB/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:01<00:00, 17.8MB/s]
Loading checkpoint shards: 100% 2/2 [00:20<00:00, 8.54s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 16.8kB/s]
Colab notebook detected. To show errors in colab noteb
* Running on public URL: https://a6a499b6b5db64231f.gr
This share link expires in 1 week. For free permanent
gradio
No interface is running right now

City Analysis & Citizen Services AI

City Analysis | Citizen Services

Enter City Name

Chennai

Analyze City

City Analysis (Crime Index & Accidents)

1. Crime Index and Safety Statistics:

Chennai, the capital city of Tamil Nadu, India, has seen a steady decline in its crime rate over the past decade. According to the National Crime Records Bureau (NCRB), the city's Crime Index (a composite index of crimes against persons, property, and government) stood at 276.1 in 2020, indicating a significant improvement from the previous years. This index is ranked 81st among Indian cities, positioning Chennai in the category of cities with lower crime rates compared to other metropolises.

Key areas of concern, however, include:

- Robbery with violence: Chennai has a moderately high rate, with 68 incidents per 100,000 population in 2020.
- Motor vehicle-related offenses: The city faces challenges with speeding and motorcycle

12. Known Issues:

- **“Current known issues include occasional delayed responses from the AI models, minor UI glitches on some devices, and dependency version conflicts during setup.”**

13.Future Enhancement:

- **Add multilingual support for wider citizen access.**
- **Integrate a mobile app for on-the-go interaction.**
- **Develop a real-time analytics and reporting dashboard.**
- **Improve AI accuracy for conversations and summaries.**
- **Expand coverage to include more citizen services and departments.**

Thank you!