Project Title

Project Documentation

1.Introduction

• **Project title: Citizen AI-Intelligent Citizen Engagement Platform.**

 • **Team member: Akshaya V.**

• **Team member: Anusiya M.**

• **Team member: Devi Sri T.**

 • **Team member: Archana D.**

2.Project overview:

The **Citizen AI** project is focused on developing an intelligent system that encourages active citizen participation in governance and community development. The project leverages Artificial Intelligence (AI), Natural Language Processing (NLP), and Machine Learning (ML) to create an interactive platform where citizens can express opinions, report issues, and collaborate on solutions.

Key Objectives

1. Citizen Engagement – Provide a platform for people to share feedback, complaints, and suggestions about civic issues.

2. AI-Powered Analysis – Use NLP to analyze citizen inputs, categorize concerns, and prioritize them.

3. Transparency & Collaboration – Help governments and organizations understand public needs and make data-driven decisions.

4. Accessibility – Ensure the platform is easy to use with multilingual and voice-enabled features.

5. Impact Measurement – Track progress of issues and display real-time updates for citizens.

Core Features

- Grievance reporting system – Citizens can submit complaints with text, images, or voice.

- Sentiment & priority analysis – AI evaluates urgency, tone, and impact of issues.

- Dashboard for authorities – Provides structured insights for decision-making.

- Community interaction – Citizens can upvote, comment, and collaborate on issues.

- Mobile & web support – Accessible across devices with user-friendly interfaces.

Technology Stack

- Frontend – Gradio / Web UI

- Backend – Python, Flask/Django

- AI Models – Transformers (for NLP), Torch (for ML tasks)

- Database – SQL/NoSQL for storing citizen data and issue logs

- Deployment – Cloud integration with scalability support

3. Architecture:

1. Frontend (User Interface)

- Tool: Gradio

- Users:

- Citizens – Ask questions, submit feedback, interact with chatbot
- Officials – View dashboards and sentiment reports
- Role: Provides a simple and accessible interface for both citizens and administrators.

2. Backend (AI Models & Processing)

- IBM Granite Models via Hugging Face
  - Natural Language Understanding (citizen queries)
  - Response Generation (chatbot answers)
  - Sentiment Analysis (classifies public opinion: positive/negative/neutral)

3. Deployment Layer

- Google Colab
  - Provides GPU-enabled environment (T4 GPU)
  - Low-cost and accessible setup for running the AI + Gradio app
  - Temporary hosting of the application

4. Version Control & Collaboration

- GitHub
  - Stores source code, Colab notebooks, and configuration
  - Enables collaboration among team members
  - Maintains version history

4. API Documentation

Citizen AI does not define its own custom REST APIs. Instead, it integrates existing APIs from Hugging Face (IBM Granite models) and exposes endpoints automatically via Gradio.

This documentation explains how the APIs are used, their endpoints, request/response formats, and future plans for API development.

5. Authentication:

1. Current State

- The demo application (Google Colab + Gradio) runs in open mode.

- Anyone with the Gradio link can access the chatbot and dashboard.

- No authentication or authorization layers are currently implemented.

2. Hugging Face API Authentication

Since Citizen AI depends on IBM Granite models via Hugging Face, API access requires an authentication token.

6.Testing:

- Unit Testing – Verifies individual functions like model calls and sentiment analysis.
- Integration Testing – Ensures smooth interaction between Gradio, models, and dashboard.
- Manual Testing – Checks chatbot responses and dashboard outputs by human testers.
- Deployment Testing – Confirms the app runs correctly on Google Colab with GPU.
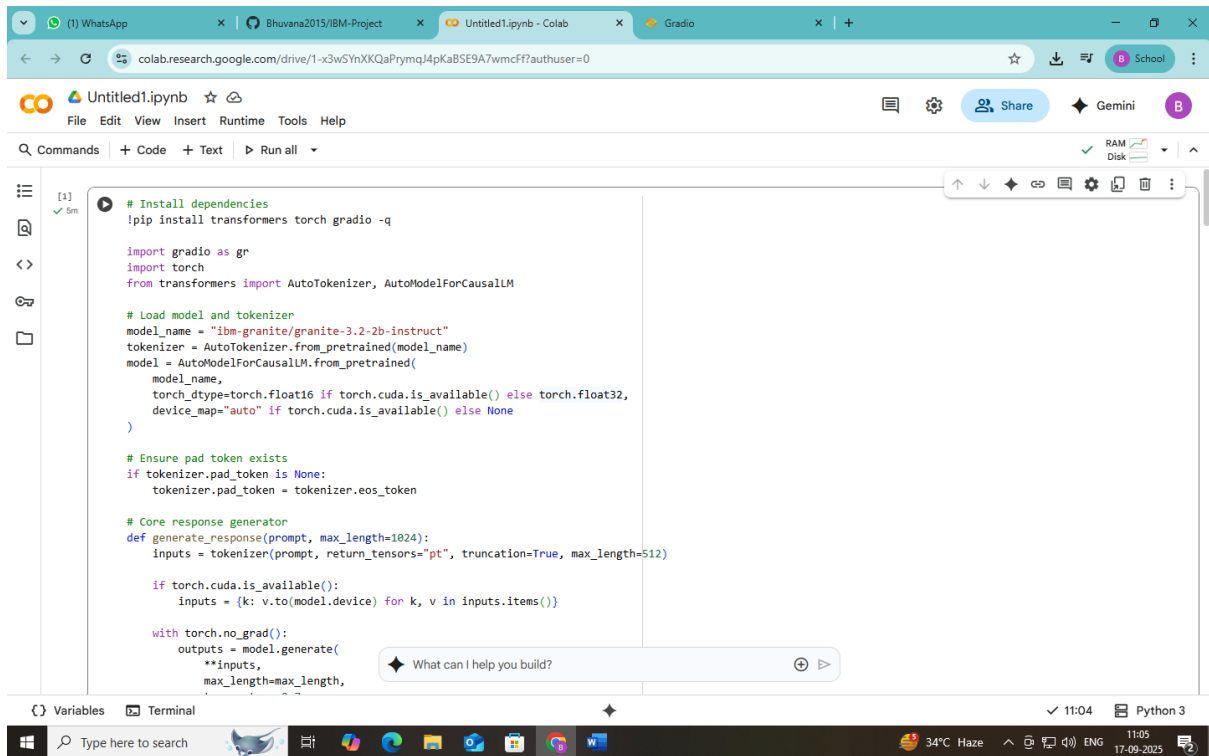
Performance Testing – Measures response time and system efficiency.

7.Screenshots:

Hugging Face:
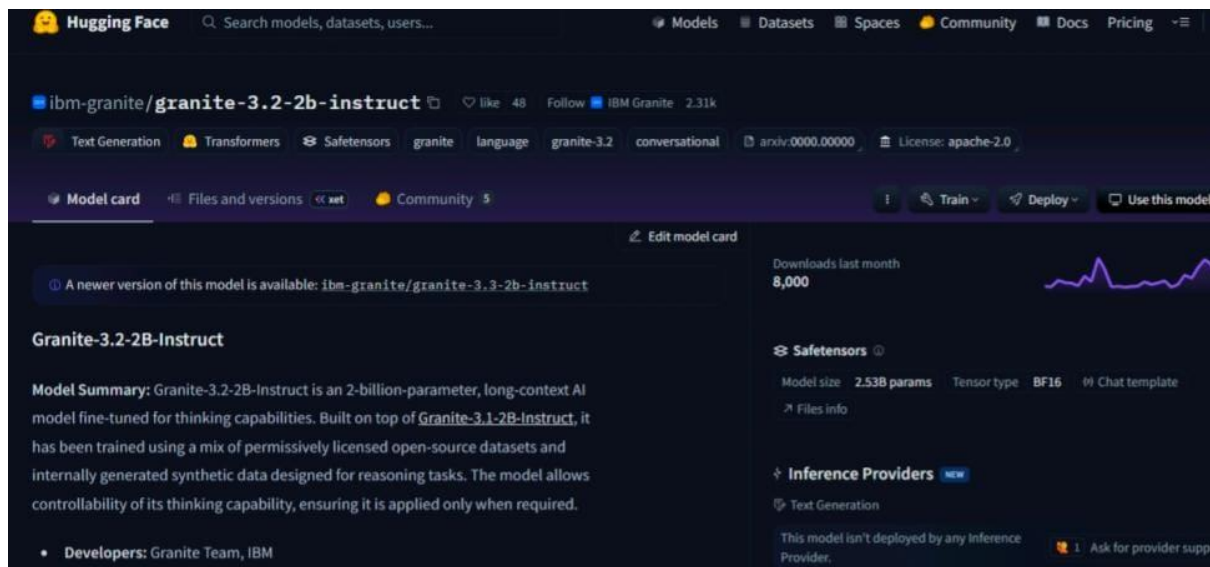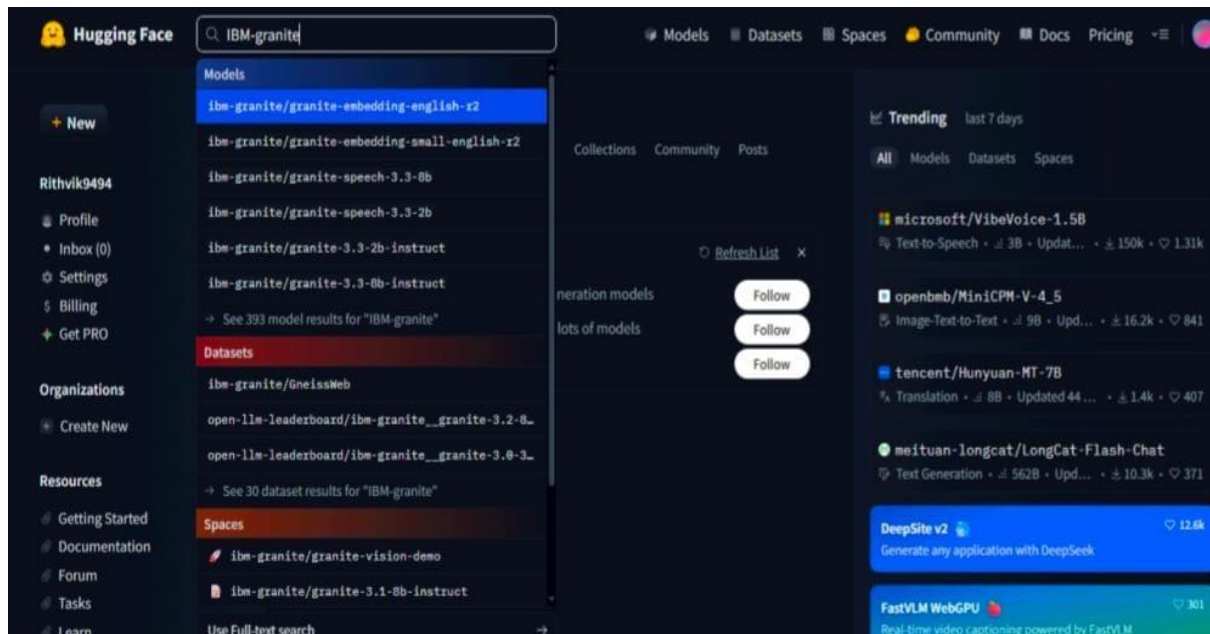
Google Colab:


```python
# Install dependencies
!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
)

# Ensure pad token exists
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

# Core response generator
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,
```
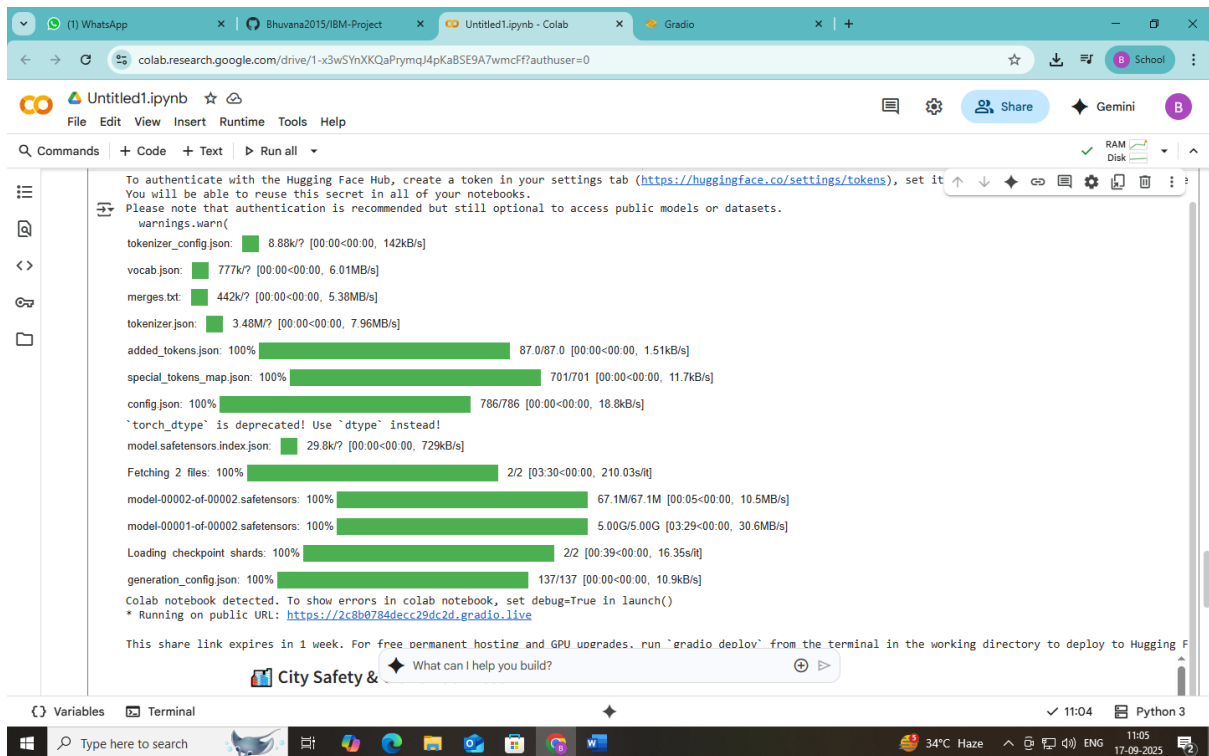


```
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
    warnings.warn(
tokenizer_config.json:     8.88k/? [00:00<00:00, 142kB/s]
vocab.json:     777k/? [00:00<00:00, 6.01MB/s]
merges.txt:     442k/? [00:00<00:00, 5.38MB/s]
tokenizer.json:     3.48M/? [00:00<00:00, 7.96MB/s]
added_tokens.json: 100%     87.0/87.0 [00:00<00:00, 1.51kB/s]
special_tokens_map.json: 100%     701/701 [00:00<00:00, 11.7kB/s]
config.json: 100%     786/786 [00:00<00:00, 18.8kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:     29.8k/? [00:00<00:00, 729kB/s]
Fetching 2 files: 100%     2/2 [03:30<00:00, 210.03s/it]
model-00002-of-00002.safetensors: 100%     67.1M/67.1M [00:05<00:00, 10.5MB/s]
model-00001-of-00002.safetensors: 100%     5.00G/5.00G [03:29<00:00, 30.6MB/s]
Loading checkpoint shards: 100%     2/2 [00:39<00:00, 16.35s/it]
generation_config.json: 100%     137/137 [00:00<00:00, 10.9kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://2c8b0784decc29dc2d.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging F
```

Output:
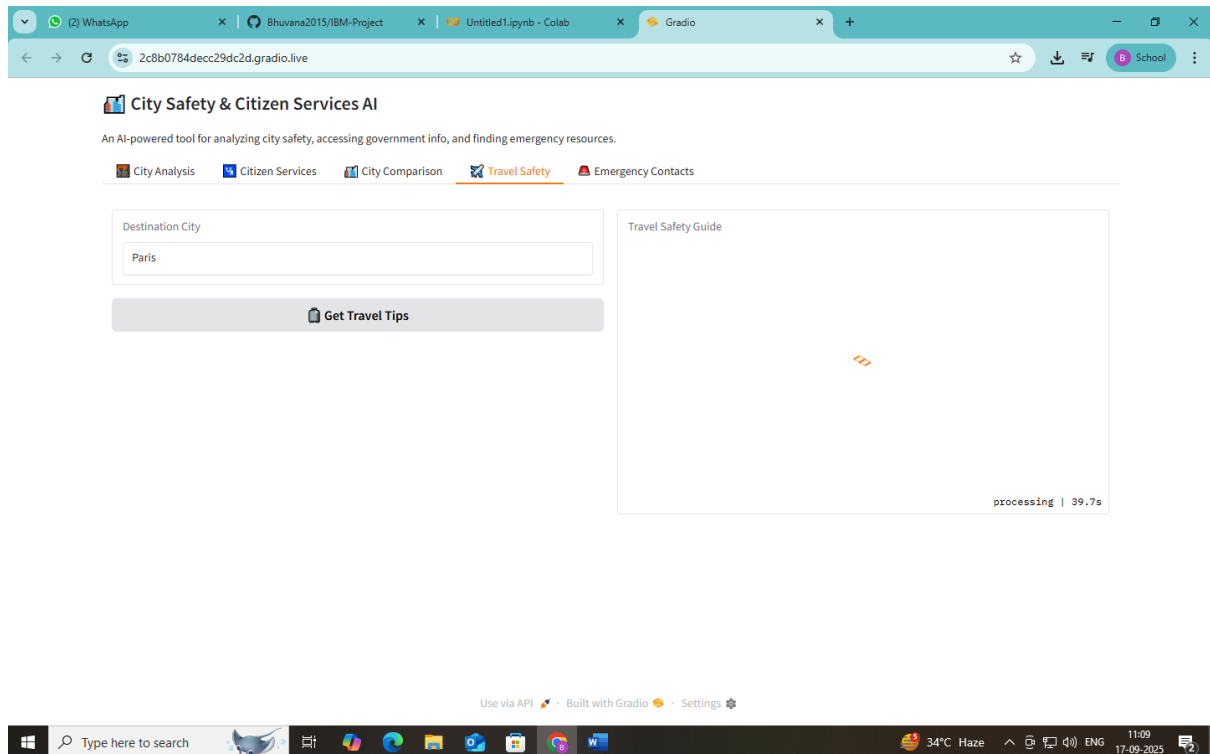
8.Known Issues:

Colab runtime sessions are temporary (session resets after timeout)

- Dependent on stable internet connection
- Relies on Hugging Face Granite model availability

9.Future enhancement:

- **Dedicated Cloud Deployment** – Host the system on IBM Cloud, AWS, or Azure for stability.

- **Advanced Dashboards** – Add analytics, charts, and deeper insights for officials.
- **Multi-language Support** – Enable chatbot interaction in regional languages.
- **Real Data Integration** – Connect with actual government databases for live updates.