

CITIZEN AI: INTELLIGENT CITIZEN ENGAGEMENT PLATFORM

Project Description:

Citizen AI uses the Granite model from Hugging Face to give quick, helpful answers about government services and civic issues. It tracks public sentiment and shows simple dashboards for officials to see feedback. This project will be deployed in Google Colab using Granite for easy, low-cost setup and reliable performance.

My team has successfully enrolled for the project. Find the team details below.

Team ID: NM2025TMID06280

Team Size: 4

Team Leader: ILAIYARAJA C

Team member: CHANDRA PRASANTH M

Team member: HALITH K

Team member: JEEVANANTHAM M

Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

- Search for “Naan Mudhalavan Smart Interz” Portal in any Browser.
- Then Click on the first link.
- ([Naanmudhalvan Smartinternz](#)) Then login with your details.
- Then you will be redirected to your account then click on “Projects”
- Section.
- There you can see which project you have enrolled in here it is “Citizen AI”.
- Then click on “Access Resources” and go to the “Guided Project” Section.
- Click on the “Go to workspace” section.
- Then you can find the detailed explanation of Generative AI Project using IBM API key.
- Click on “Project Workspace”, there you can find your project progress and Place to upload “Demo link”.
- Now we have gone through portal understanding, now lets find a IBM granite model from hugging face to integrate in our project.

Activity-2: Choose a IBM Granite model From Hugging Face.

Search for “Hugging face” in any browser.

Then click on the first link ([Hugging Face](#)), then click on signup and create your own account in Hugging Face.

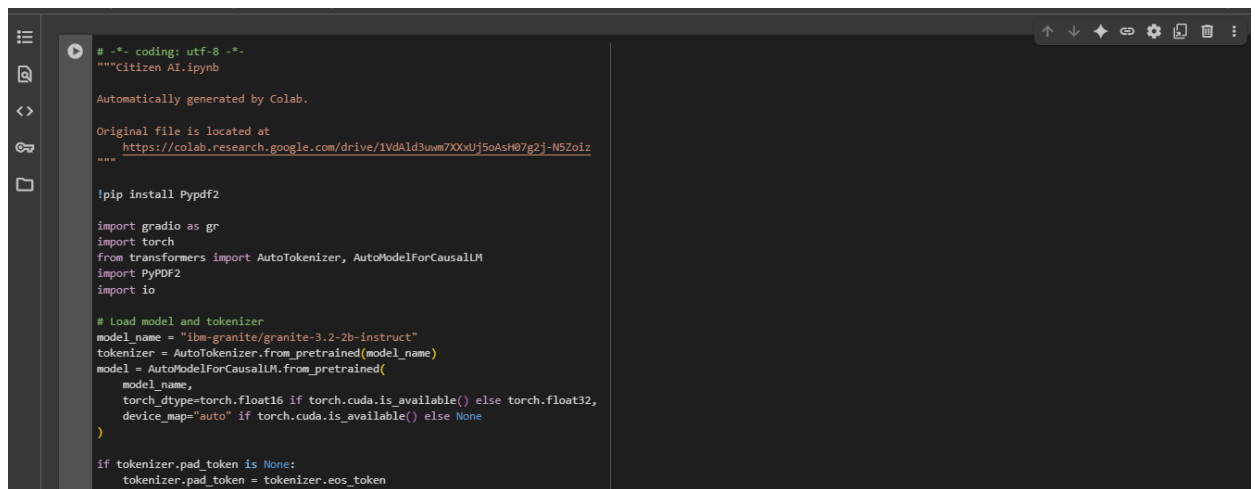
Then search for “IBM-Granite models” and choose any model.

Here for this project we are using “granite-3.2-2b-instruct” which is compatible fast and light weight.

Now we will start building our project in Google collab.

Activity-3: Running Application in Google Collab.

- ✓ Search for “Google collab” in any browser.
- ✓ Click on the first link ([Google Colab](https://colab.research.google.com/)), then click on “Files” and then “Open Notebook”.
- ✓ Click on “New Notebook”.
- ✓ Change the title of the notebook “Untitled” to “Citizen AI”. Then click on “Runtime”, then go to “Change Runtime Type”.
- ✓ Choose “T4 GPU” and click on “Save”.
- ✓ Run this command in first cell “!pip install transformers torch gradio -q”.
- ✓ To install the required libraries to run our application.
- ✓ Then run the rest of the code in the next cell.



```
# -*- coding: utf-8 -*-
"""Citizen AI.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1VdAld3uwm7XXxUj5oAsH07g2j-N5Zoiz

!pip install Pypdf2

import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
import PyPDF2
import io

# 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
        )

    # slice off input tokens to avoid duplication
    gen_tokens = outputs[0][inputs["input_ids"].shape[-1]:]
    response = tokenizer.decode(gen_tokens, skip_special_tokens=True).strip()
    return response

def extract_text_from_pdf(pdf_file):
    if pdf_file is None:
        return ""

    try:
        pdf_reader = PyPDF2.PdfReader(pdf_file)
        text = ""
        for page in pdf_reader.pages:
            text += page.extract_text() + "\n"
        return text

```

```

except Exception as e:
    return f"Error reading PDF: {str(e)}"

# --- Citizen Complaint Draft Generator ---
def citizen_complaint_generator(issue_description):
    prompt = f"Draft a clear and polite citizen complaint for the issue: {issue_description}. " \
            f"Include location, impact on residents, and request timely action."
    return generate_response(prompt, max_length=1000)

# --- Policy Summarization ---
def policy_summarization(pdf_file, policy_text):
    # Get text from PDF or direct input
    if pdf_file is not None:
        content = extract_text_from_pdf(pdf_file)
        summary_prompt = f"Summarize the following government policy and extract the most important points, " \
            f"benefits to citizens, and steps for application:\n\n{content}"
    else:
        summary_prompt = f"Summarize the following government policy and extract the most important points, " \
            f"benefits to citizens, and steps for application:\n\n{policy_text}"

    return generate_response(summary_prompt, max_length=1200)

# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# CitizenAI 🤖 Intelligent Citizen Engagement Platform")

    with gr.Tabs():
        # Complaint Generator
        with gr.TabItem("Citizen Complaint Generator"):
            with gr.Row():
                with gr.Column():
                    issue_input = gr.Textbox(
                        label="Describe Your Civic Issue",
                        placeholder="e.g., open pothole near bus stop, garbage not cleared, water leakage...",
                        lines=3
                    )
                    generate_complaint_btn = gr.Button("Generate Complaint Draft")

                with gr.Column():
                    complaint_output = gr.Textbox(
                        label="AI Generated Complaint Draft",
                        lines=15,
                        show_copy_button=True
                    )

            generate_complaint_btn.click(citizen_complaint_generator, inputs=issue_input, outputs=complaint_output)

        # Policy Summarization
        with gr.TabItem("Policy Summarization"):
            with gr.Row():
                with gr.Column():
                    pdf_upload = gr.File(label="Upload Policy PDF", file_types=[".pdf"])
                    policy_text_input = gr.Textbox(
                        label="Or paste policy text here",
                        placeholder="Paste policy document text...",
                        lines=5
                    )
                    summarize_btn = gr.Button("Summarize Policy")

```

```

            with gr.Column():
                issue_input = gr.Textbox(
                    label="Describe Your Civic Issue",
                    placeholder="e.g., open pothole near bus stop, garbage not cleared, water leakage...",
                    lines=3
                )
                generate_complaint_btn = gr.Button("Generate Complaint Draft")

            with gr.Column():
                complaint_output = gr.Textbox(
                    label="AI Generated Complaint Draft",
                    lines=15,
                    show_copy_button=True
                )

            generate_complaint_btn.click(citizen_complaint_generator, inputs=issue_input, outputs=complaint_output)

        # Policy Summarization
        with gr.TabItem("Policy Summarization"):
            with gr.Row():
                with gr.Column():
                    pdf_upload = gr.File(label="Upload Policy PDF", file_types=[".pdf"])
                    policy_text_input = gr.Textbox(
                        label="Or paste policy text here",
                        placeholder="Paste policy document text...",
                        lines=5
                    )
                    summarize_btn = gr.Button("Summarize Policy")

```

```
with gr.TabItem("Policy Summarization"):
    with gr.Row():
        with gr.Column():
            pdf_upload = gr.File(label="Upload Policy PDF", file_types=[".pdf"])
            policy_text_input = gr.Textbox(
                label="Or paste policy text here",
                placeholder="Paste policy document text...",
                lines=5
            )
            summarize_btn = gr.Button("Summarize Policy")

        with gr.Column():
            summary_output = gr.Textbox(
                label="Policy Summary & Key Points",
                lines=20,
                show_copy_button=True
            )

    summarize_btn.click(policy_summarization, inputs=[pdf_upload, policy_text_input], outputs=summary_output)

app.launch(share=True)
```

- You can find the code here in this link: [CitizenAI Code](#)

Output:


- Now you can see our model is being Downloaded and application is running.

```
warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 333kB/s]
vocab.json: 777k/? [00:00<00:00, 8.47MB/s]
merges.txt: 442k/? [00:00<00:00, 13.0MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 56.2MB/s]
added_tokens.json: 100% ██████████ 87.0/87.0 [00:00<00:00, 4.28kB/s]
special_tokens_map.json: 100% ██████████ 701/701 [00:00<00:00, 37.2kB/s]
config.json: 100% ██████████ 786/786 [00:00<00:00, 65.1kB/s]
model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.98MB/s]
Fetching 2 files: 100% ██████████ 2/2 [02:16<00:00, 136.40s/it]
model-00001-of-00002.safetensors: 100% ██████████ 5.00G/5.00G [02:16<00:00, 49.6MB/s]
model-00002-of-00002.safetensors: 0% ██████████ 0.00/67.1M [00:00<?, ?B/s]
Loading checkpoint shards: 100% ██████████ 2/2 [00:32<00:00, 13.39s/it]
generation_config.json: 100% ██████████ 137/137 [00:00<00:00, 11.6kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://165837bd9b57952524.gradio.live
```

- Click on the URL to open the Gradio Application
<https://colab.research.google.com/drive/1QpuP1c0wXz1c-U97O82RvJVdKcZFInIf?usp=sharing> click on the link.

Activity-4: Upload Your Project in GitHub.

- ❖ Search for “GitHub” in any browser, then click on the first link ([GitHub](#)).
- ❖ Then click on “Signup” and create your own account in GitHub.
- ❖ If you already have an account click on “Sign in”.
- ❖ Click on “Create repository”.
- ❖ In “General” Name your repo. (Here I have given “<https://github.com/ilaiyaraja106886/NM2025TMID06280-PROJECT.git>” as my repo name and it is available)
- ❖ In “Configurations” Turn On “Add readme” file Option.
- ❖ Now Download your code from Google collab by Clicking on “File”, then Goto “Download” then download as “.py”.
- ❖ Then your repository is created, then Click on “Add file” Option.
- ❖ Then Click “Upload files” to upload your files.
- ❖ Click on “choose your file”.
- ❖ Choose your project file and click on “Open”.
- ❖ After your file has Uploaded Click on “Commit changes”.

 THANKS TO SMART BRIDGE TEAM FOR GIVEN THIS OPPORTUNITY BY
NM2025TMID06280 - PROJECT
TEAM.