

# **Citizen AI Documentation**

**Project Title: Citizen AI: Intelligent Citizen Engagement platform**

**Team Member: Pavithra P**

**Team Member: Pavithra R**

**Team Member: Pooja I**

**Team Member: Pooja M**

---

## **1. Introduction**

Citizen AI with IBM is an **intelligent citizen engagement platform** designed to help people easily access information about government services, civic issues, and public programs.

Instead of manually searching across different websites, citizens can ask questions and get **quick, AI-driven responses** powered by IBM Granite models.

One of the key features of Citizen AI is its ability to **track public sentiment** by analyzing user queries and feedback. The system provides **dashboards** that allow government officials to view the concerns and opinions of citizens in a structured manner.

This platform is built on **Generative AI** technology, ensuring that the responses are not only fact-based but also conversational and easy to understand.

## **Objective**

The main objective of Citizen AI is to:

- Improve citizen interaction with government services
- Reduce response times with AI-generated answers

- Provide officials with tools to monitor public feedback
  - Use **open-source AI models (IBM Granite)** for cost efficiency
  - Deploy on **Google Colab** for accessibility without expensive infrastructure
- 

## 2. Pre-requisites

To successfully build and run the Citizen AI project, certain **skills and tools** are required:

1. **Gradio Framework Knowledge**
    - Gradio is a Python library that allows easy creation of interactive AI applications.
    - Used to build the chatbot interface for Citizen AI.
  2. **IBM Granite Models (Hugging Face)**
    - IBM Granite models are powerful, lightweight AI models hosted on Hugging Face.
    - Example: granite-3.2-2b-instruct, suitable for running in limited compute environments.
  3. **Python Programming Proficiency**
    - Necessary for writing and executing code in Colab.
    - Includes knowledge of libraries like transformers and torch.
  4. **Version Control with Git**
    - Git and GitHub are essential for storing, updating, and collaborating on the project.
  5. **Google Colab T4 GPU**
    - The project is deployed in Colab using T4 GPU for free/affordable computation.
    - Users must configure runtime settings correctly.
- 

## 3. Project Workflow

The project development is divided into **four main activities**:

- **Activity 1:** Exploring the Naan Mudhalavan Smart Interz Portal
- **Activity 2:** Choosing an IBM Granite Model from Hugging Face
- **Activity 3:** Running the Application in Google Colab
- **Activity 4:** Uploading the Project in GitHub

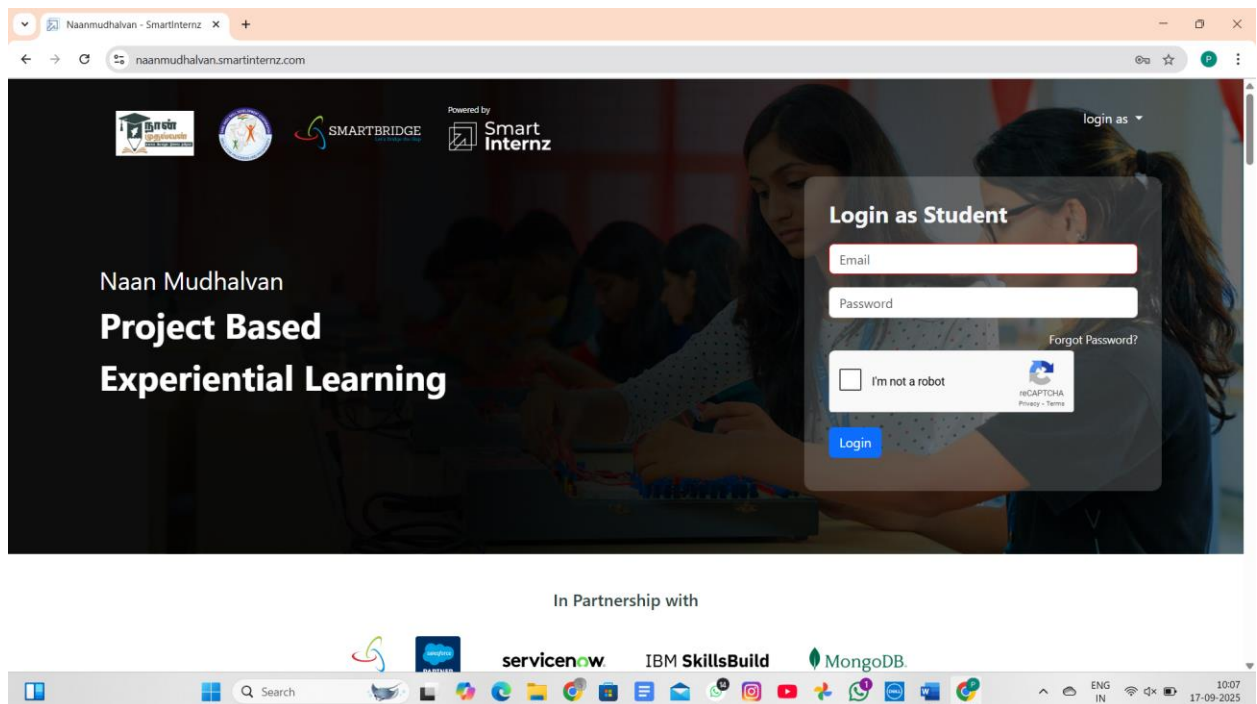
This workflow ensures a smooth process, starting from **enrollment in the project portal**, to **building the AI system**, and finally, to **deployment and version control**.

---

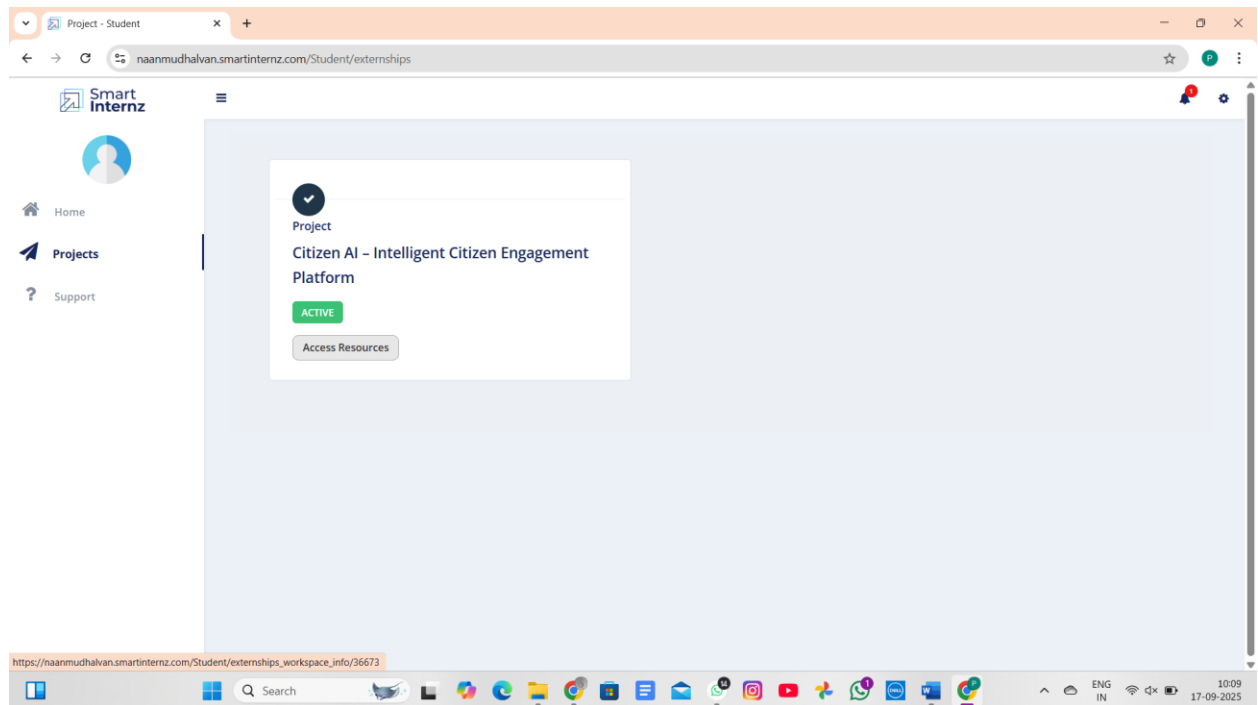
## 4. Activities in Detail

### Activity 1: Exploring Naan Mudhalavan Smart Interz Portal

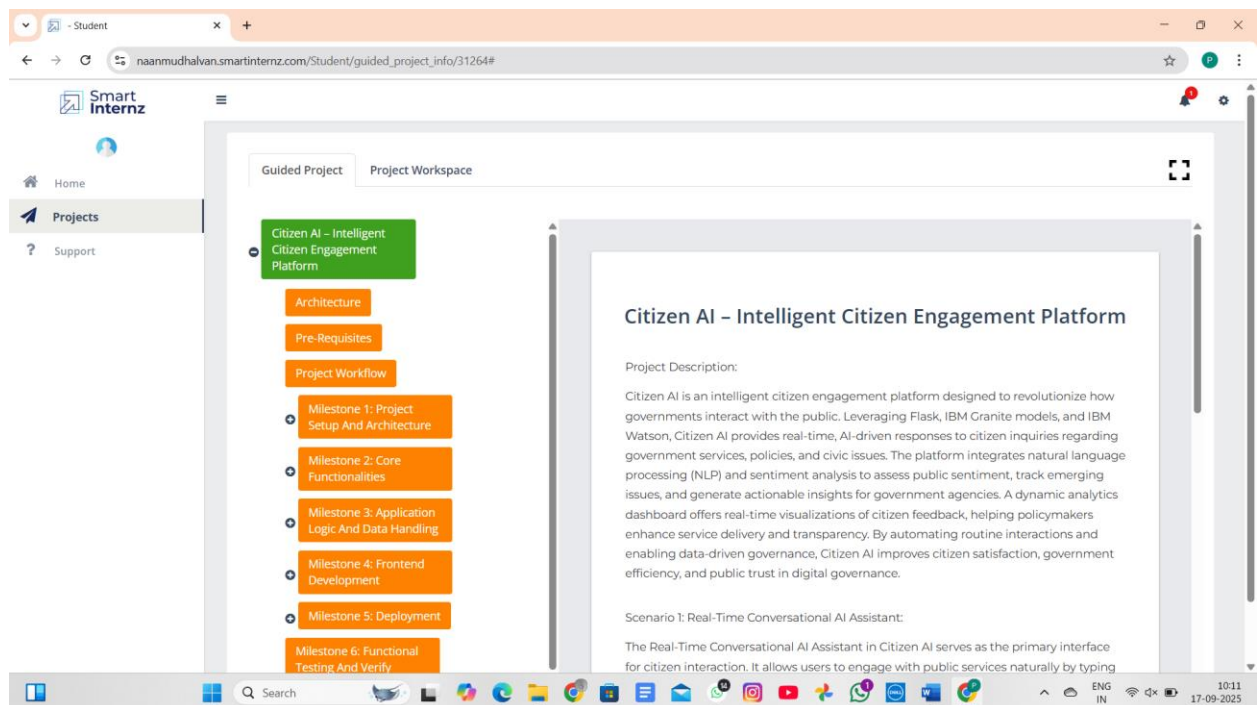
- Login to the **Naan Mudhalavan Smart Interz Portal**.



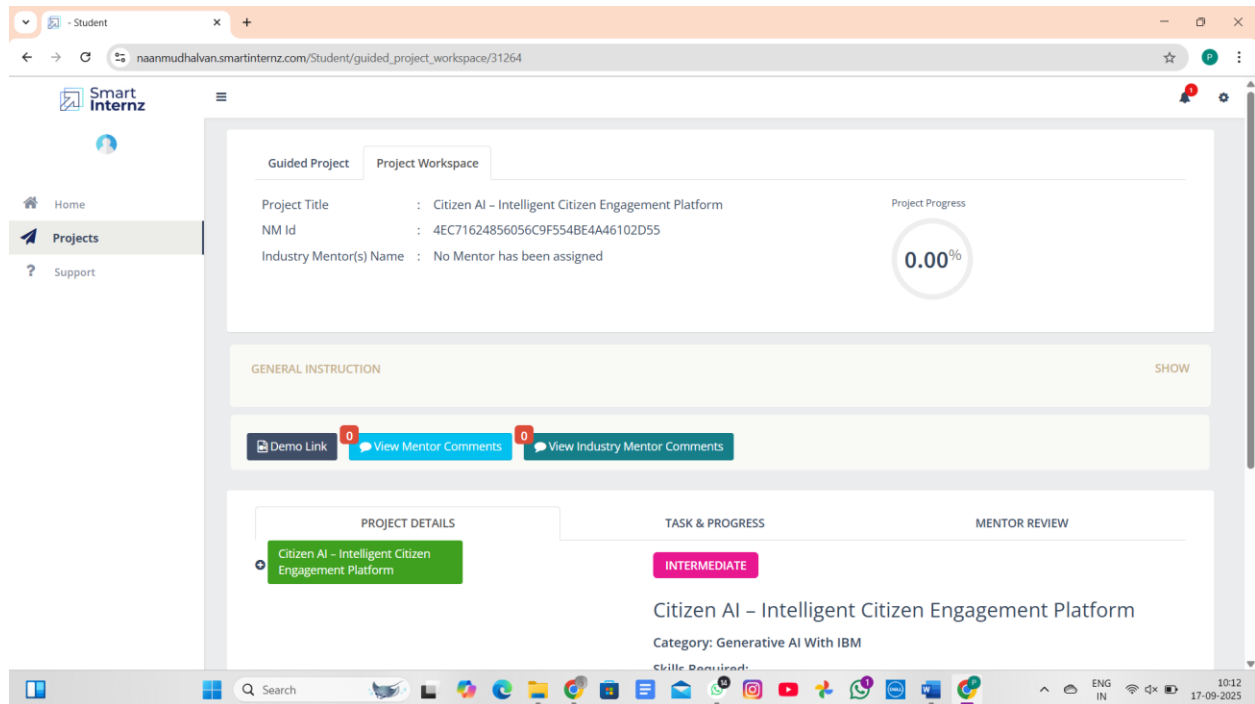
- Navigate to the **Projects Section** and choose **Citizen AI**.



- Access project resources under the **Guided Project Section**.

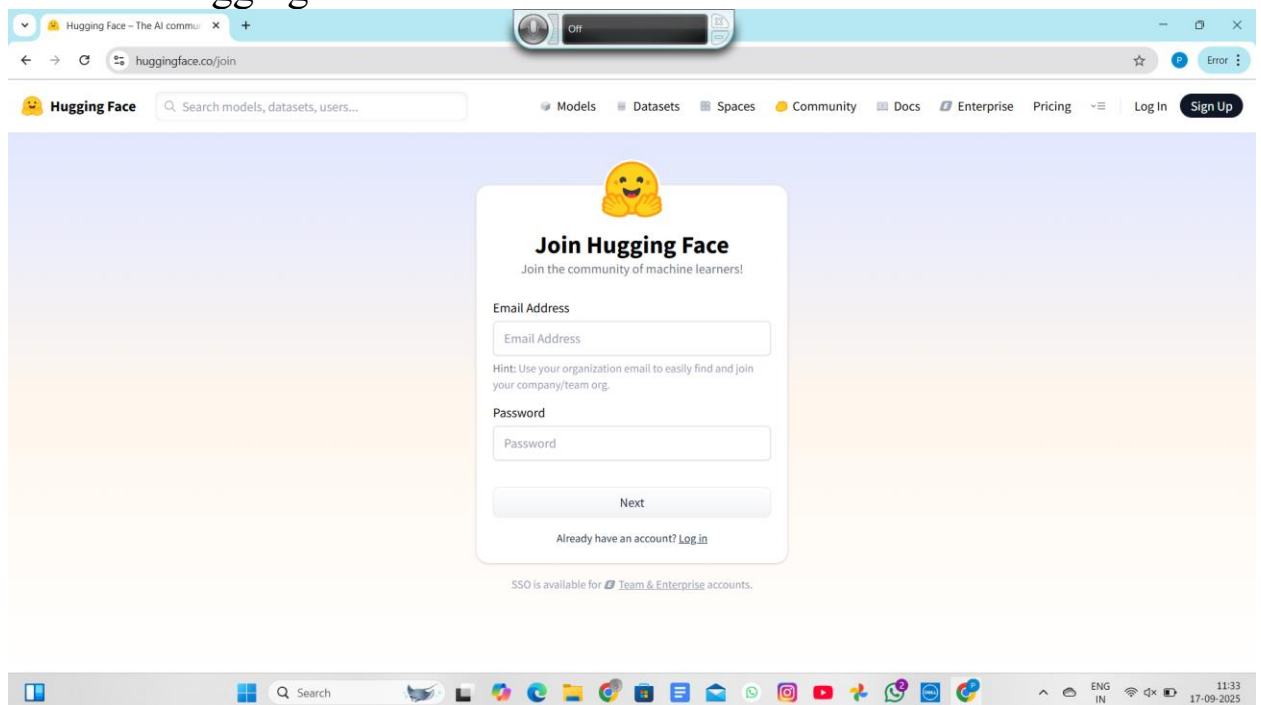


- Open the **Workspace** to check progress and upload project demo links.

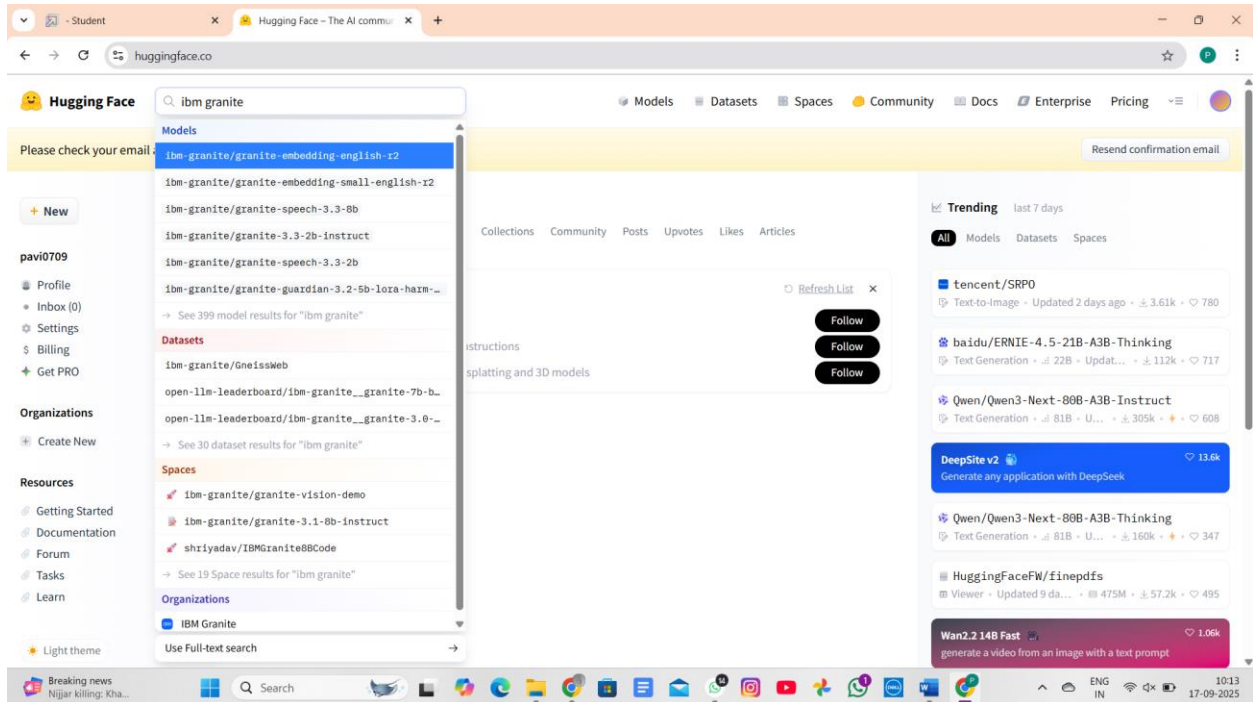


## Activity 2: Choosing an IBM Granite Model

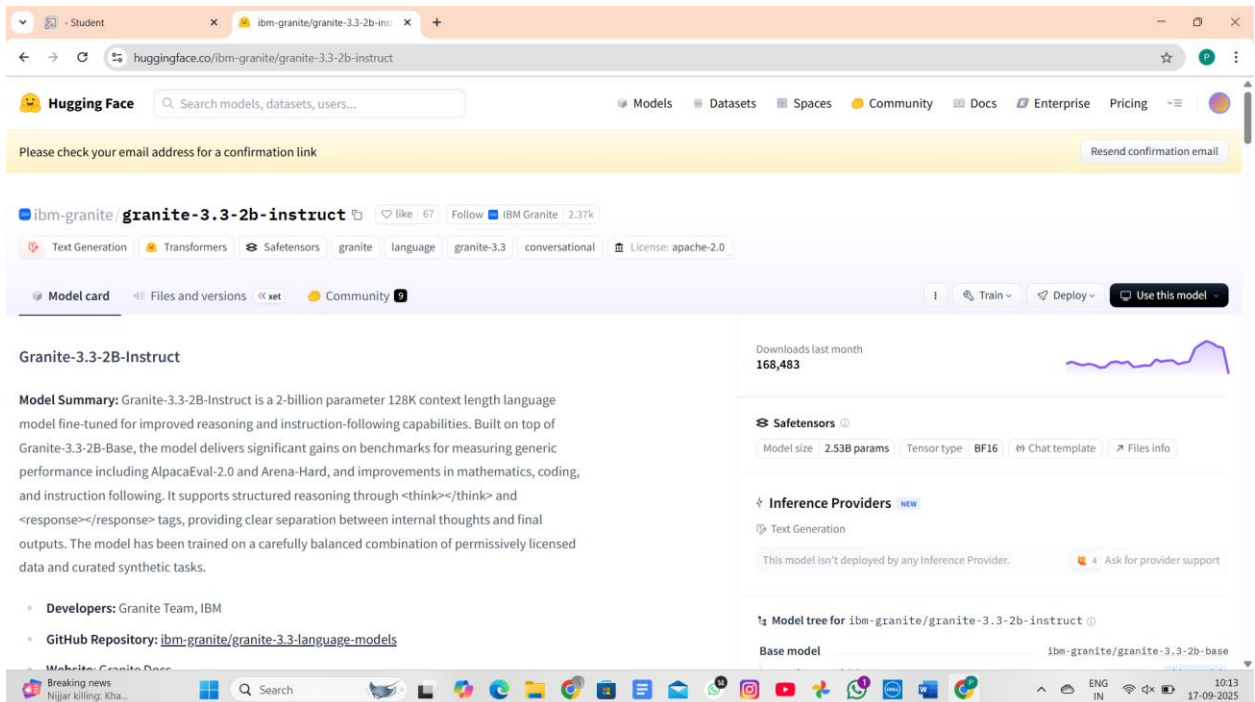
- Create a Hugging Face account.



- Search for **IBM Granite Models**.



- Select a suitable model (recommended: **granite-3.2-2b-instruct**).

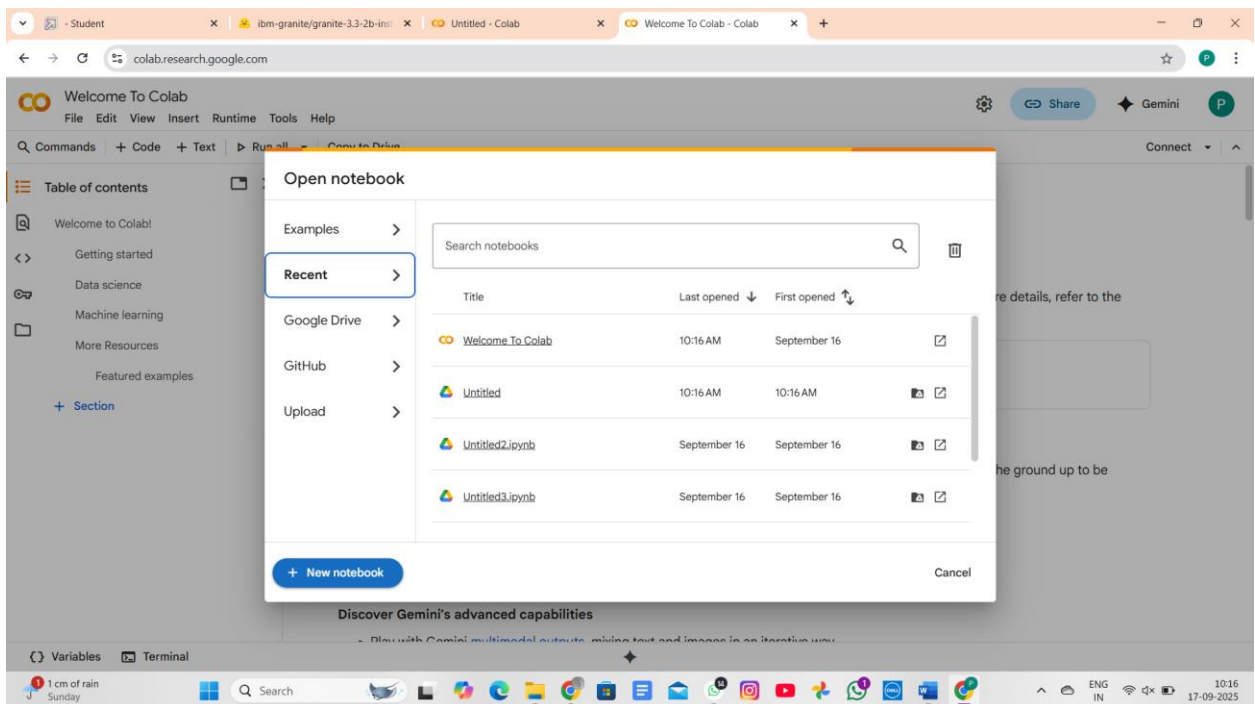


This model is **fast, lightweight, and Colab-compatible**, making it ideal for citizen engagement use cases

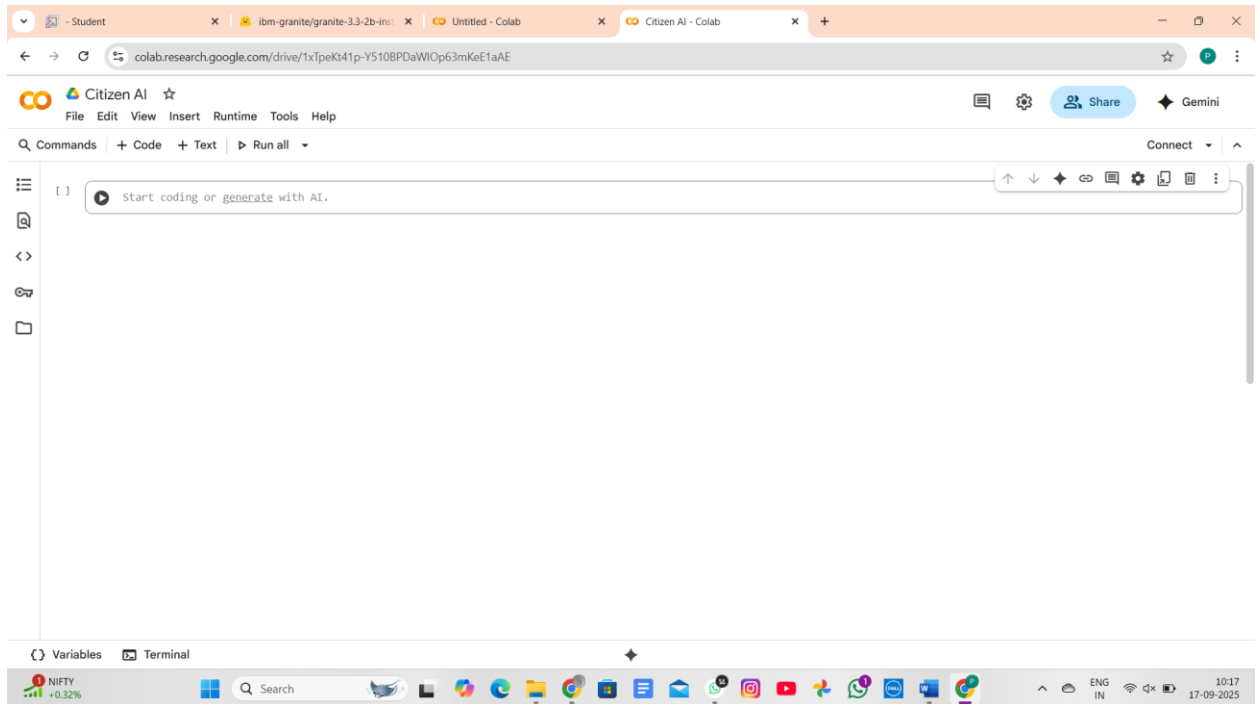
## 5. Implementation and Deployment

### Activity 3: Running the Application in Google Colab

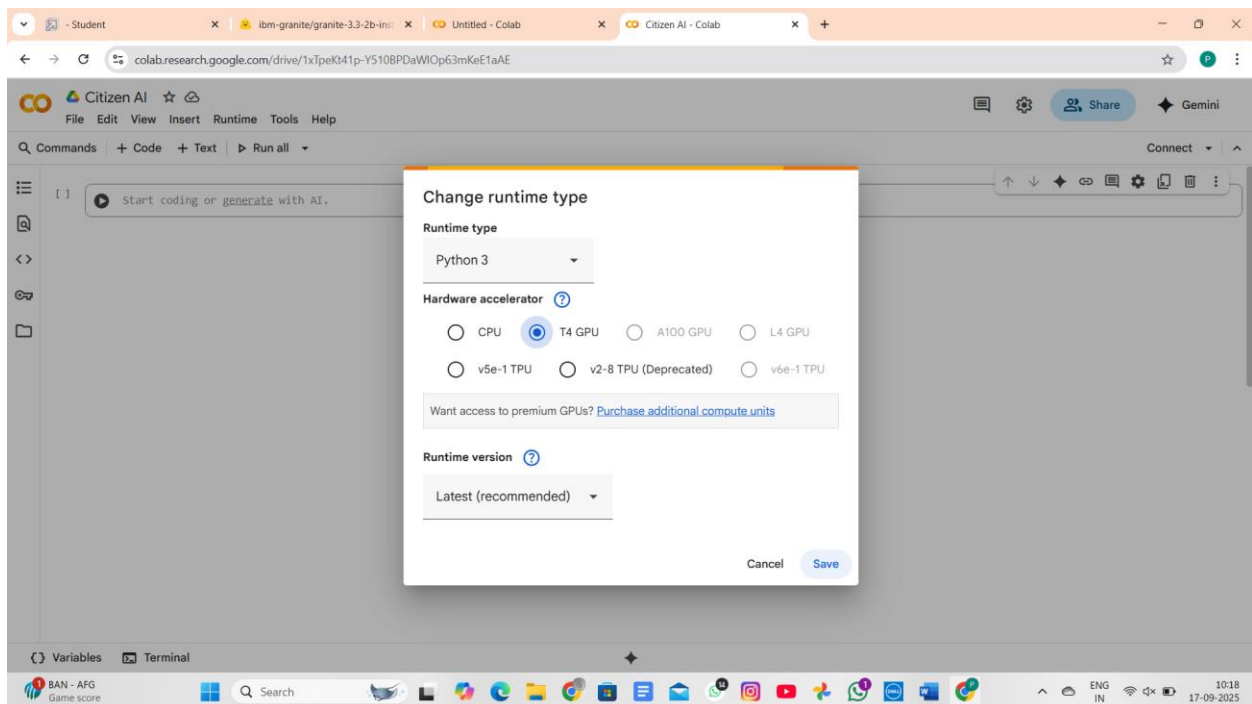
1. Open **Google Colab** and create a new notebook.



## 2. Rename it as **Citizen AI**.



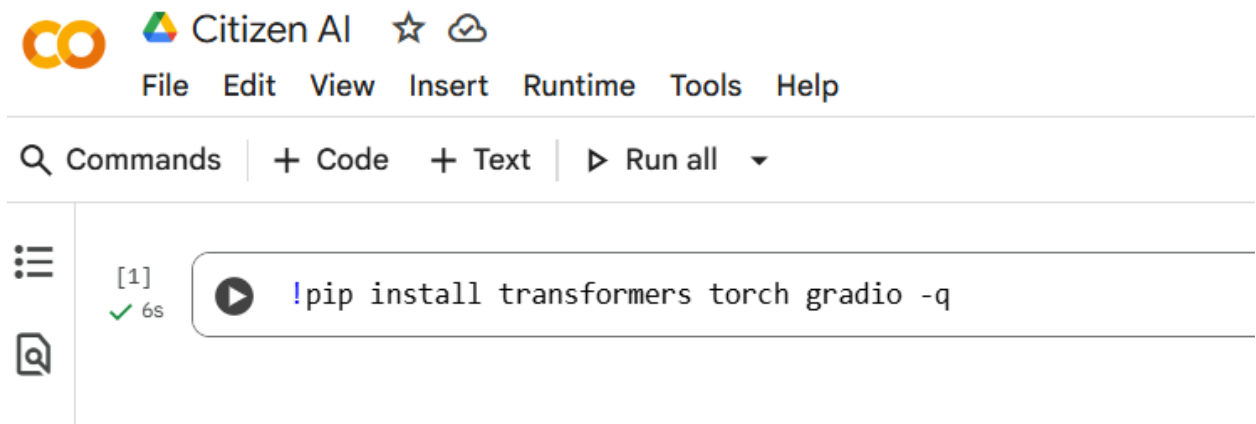
## 3. Change runtime to **T4 GPU** (Runtime → Change Runtime Type → GPU).





4. Install required libraries:

5. `!pip install transformers torch gradio -q`



6. Paste and run the project code.



```
[2] ✓ 3m
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 statistics\n2. Accident rates and traffic safety information\n3. Overall safety as\nreturn generate_response(prompt, max_length=1000)

def citizen_interaction(query):
    prompt = f"As a government assistant, provide accurate and helpful information about the following citizen query related to public services, government policies, or civic iss\nreturn 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)", lines=15)

        analyze_btn.click(analyze_btn, 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)

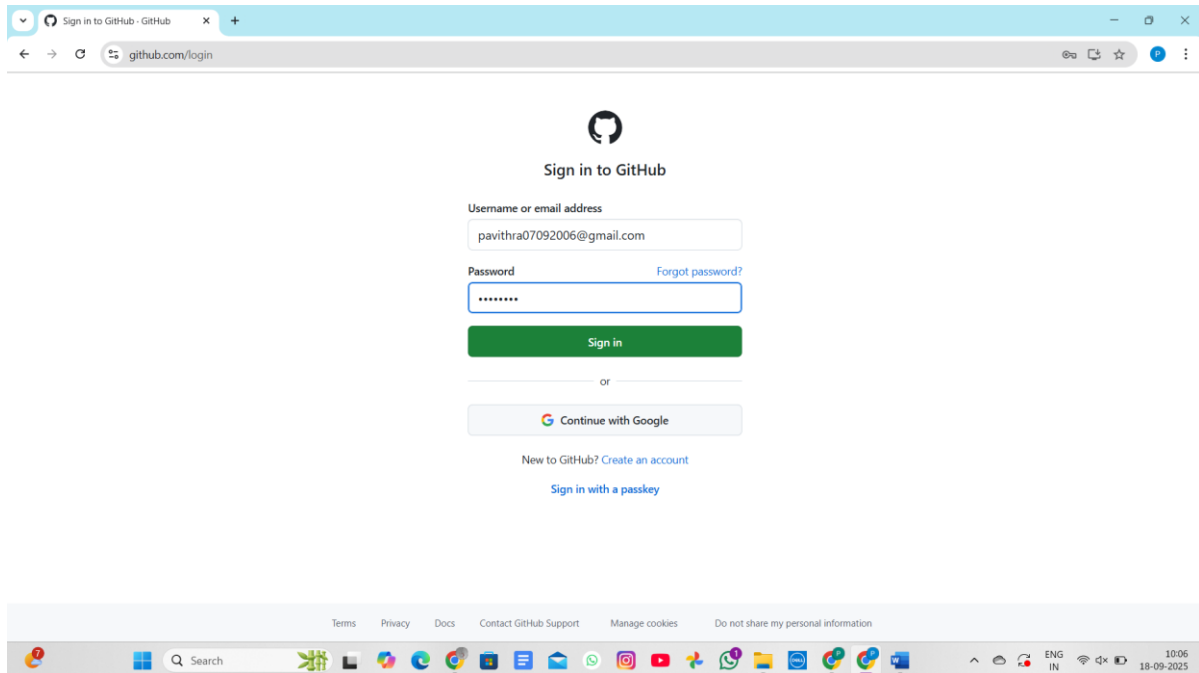
app.launch(share=True)
```

## 7. Click on the generated **Gradio** link to access the Citizen AI chatbot.

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()  
\* Running on public URL: <https://561392f96ac8213b97.gradio.live>

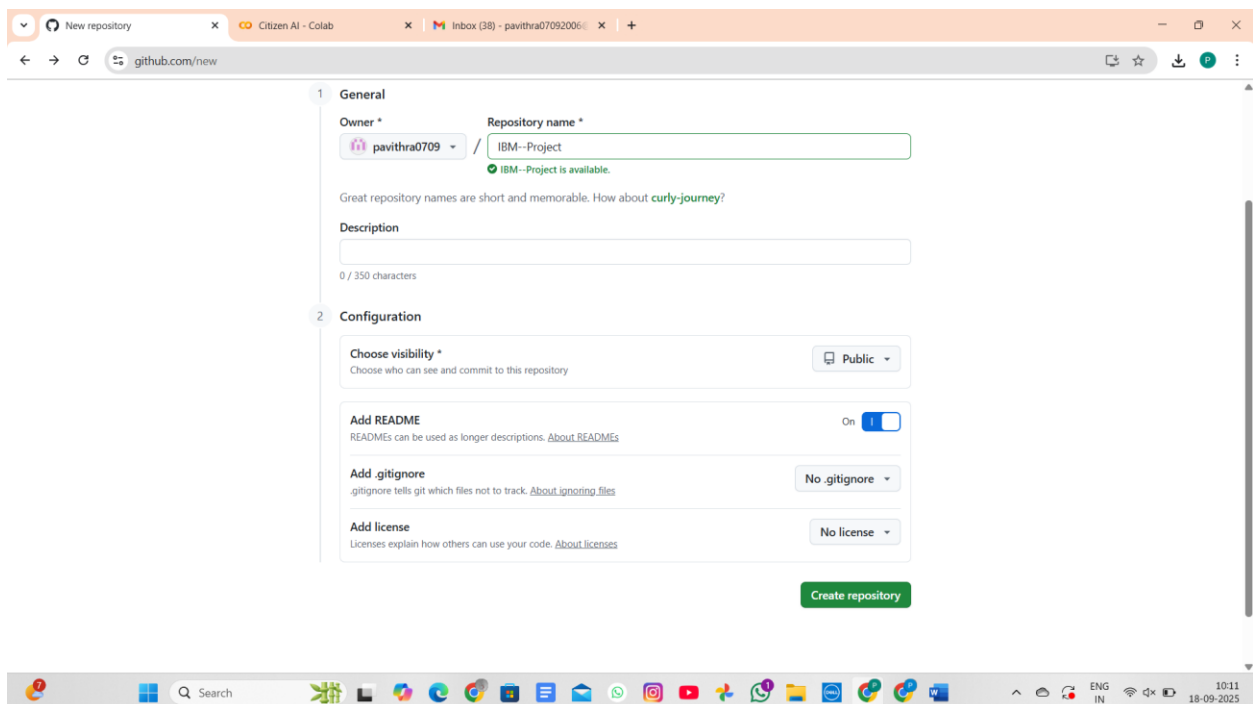
## Activity 4: Uploading to GitHub

1. Create a **GitHub account** (or log in if already created).



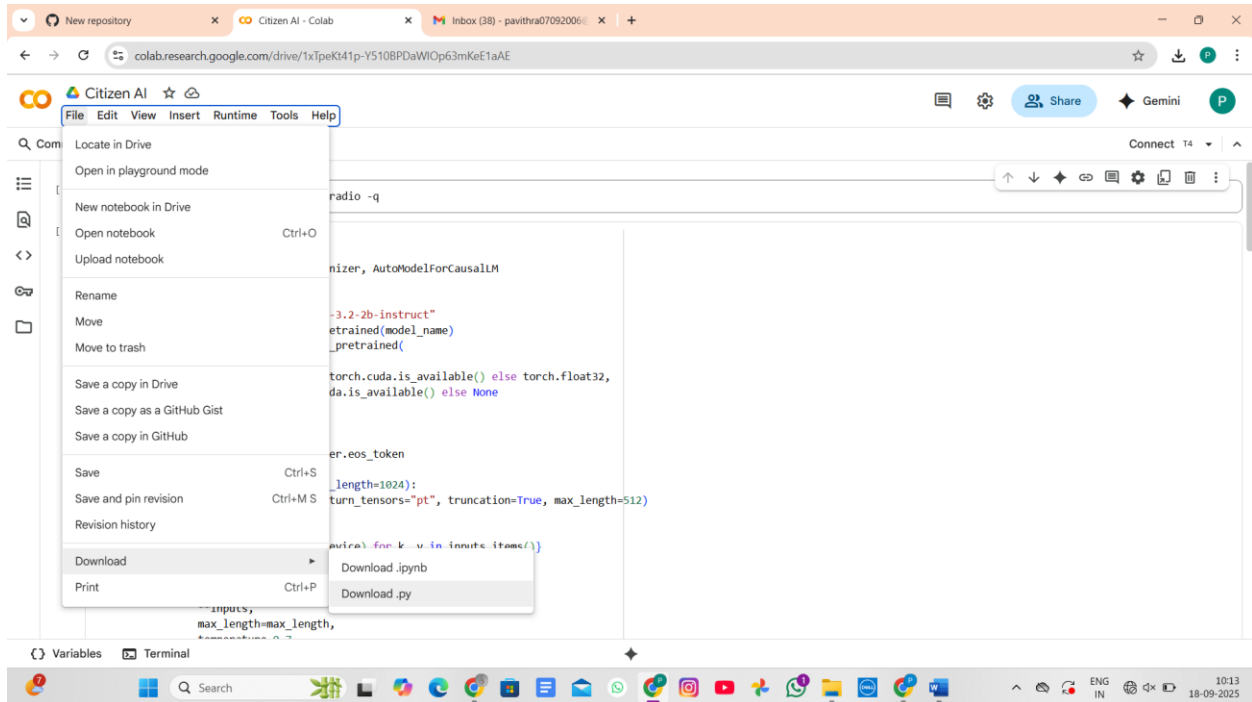
The screenshot shows the GitHub login page in a web browser. The browser's address bar displays 'github.com/login'. The page features the GitHub logo at the top, followed by the heading 'Sign in to GitHub'. Below this, there are two input fields: 'Username or email address' with the value 'pavithra07092006@gmail.com' and 'Password' with masked characters. A 'Forgot password?' link is positioned to the right of the password field. A green 'Sign in' button is located below the password field. Underneath the button is the text 'or' and a 'Continue with Google' button. At the bottom of the login section, there are two links: 'New to GitHub? Create an account' and 'Sign in with a passkey'. The browser's taskbar at the bottom shows various application icons and the system clock indicating 10:06 on 18-09-2025.

2. Create a **new repository** (e.g., IBM-Project).
3. Enable the option to **Add README**.

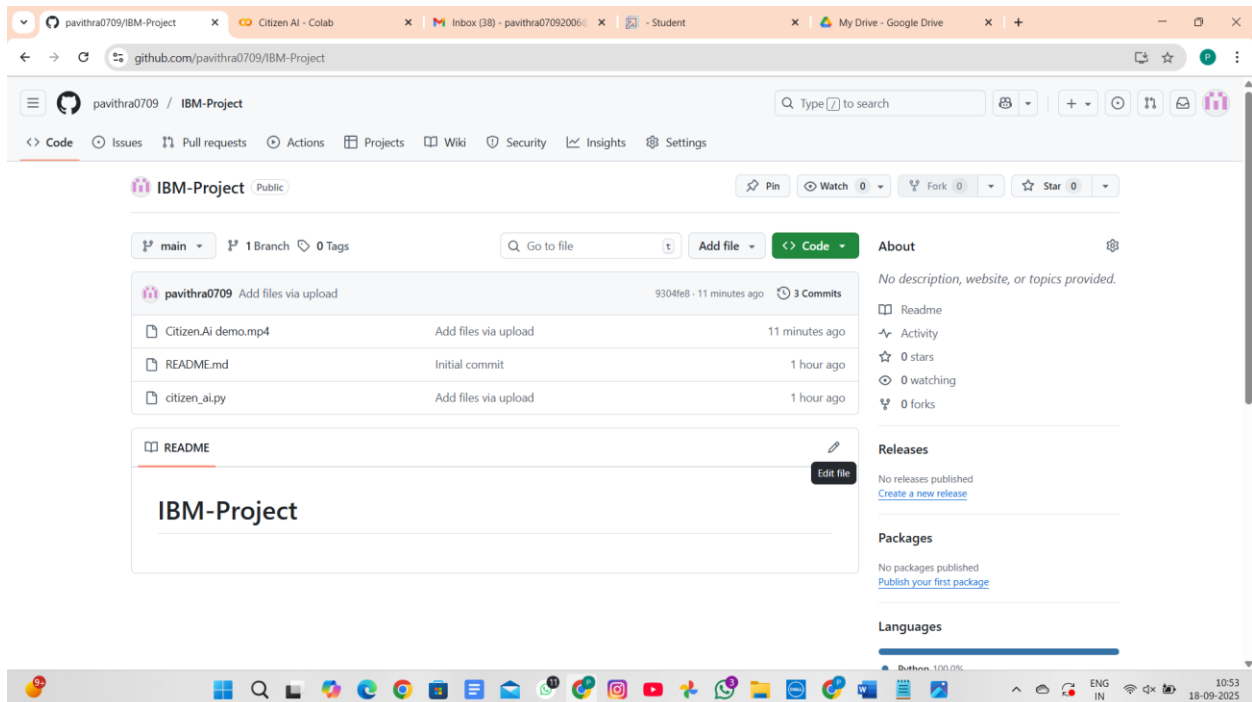


The screenshot displays the 'New repository' page on GitHub. The browser's address bar shows 'github.com/new'. The page is divided into two main sections: 'General' and 'Configuration'. In the 'General' section, the 'Owner' is set to 'pavithra0709' and the 'Repository name' is 'IBM--Project', with a confirmation message 'IBM--Project is available.' Below this is a 'Description' field. The 'Configuration' section includes a 'Choose visibility' dropdown set to 'Public', an 'Add README' section with a toggle switch turned 'On', an 'Add .gitignore' dropdown set to 'No .gitignore', and an 'Add license' dropdown set to 'No license'. A green 'Create repository' button is at the bottom right. The browser's taskbar at the bottom shows the system clock indicating 10:11 on 18-09-2025.

#### 4. From Google Colab, download your project as .py.



#### 5. Upload the file to your repository using the Upload Files option.



6. Commit changes and maintain version history.

## **Conclusion**

Citizen AI is a **practical example of AI in governance**.

It makes public services more accessible, reduces response time, and helps governments gain insights into citizens' concerns.