Practical – 1

Aim :- Implementation of LLaMA and AWS Amazon Services for Text Generation

STEPS:

1. Set Up AWS Account and Enable Bedrock Services

1. Create an **AWS Services** account using a debit card, email, and phone number.
2. Go to **Amazon Bedrock** from the AWS Management Console.
3. Click on **"Select Model."**
4. Choose **LLaMA 3 8B Instruct** from Meta.
5. Click **"Available to Request,"** then request further access and submit while keeping everything default.
6. Once access is granted, test the model using any prompt (e.g., *"Best places to visit in Banaras"*).
7. You will see the generated result from the model.

2. Setting Up Python and AWS Integration

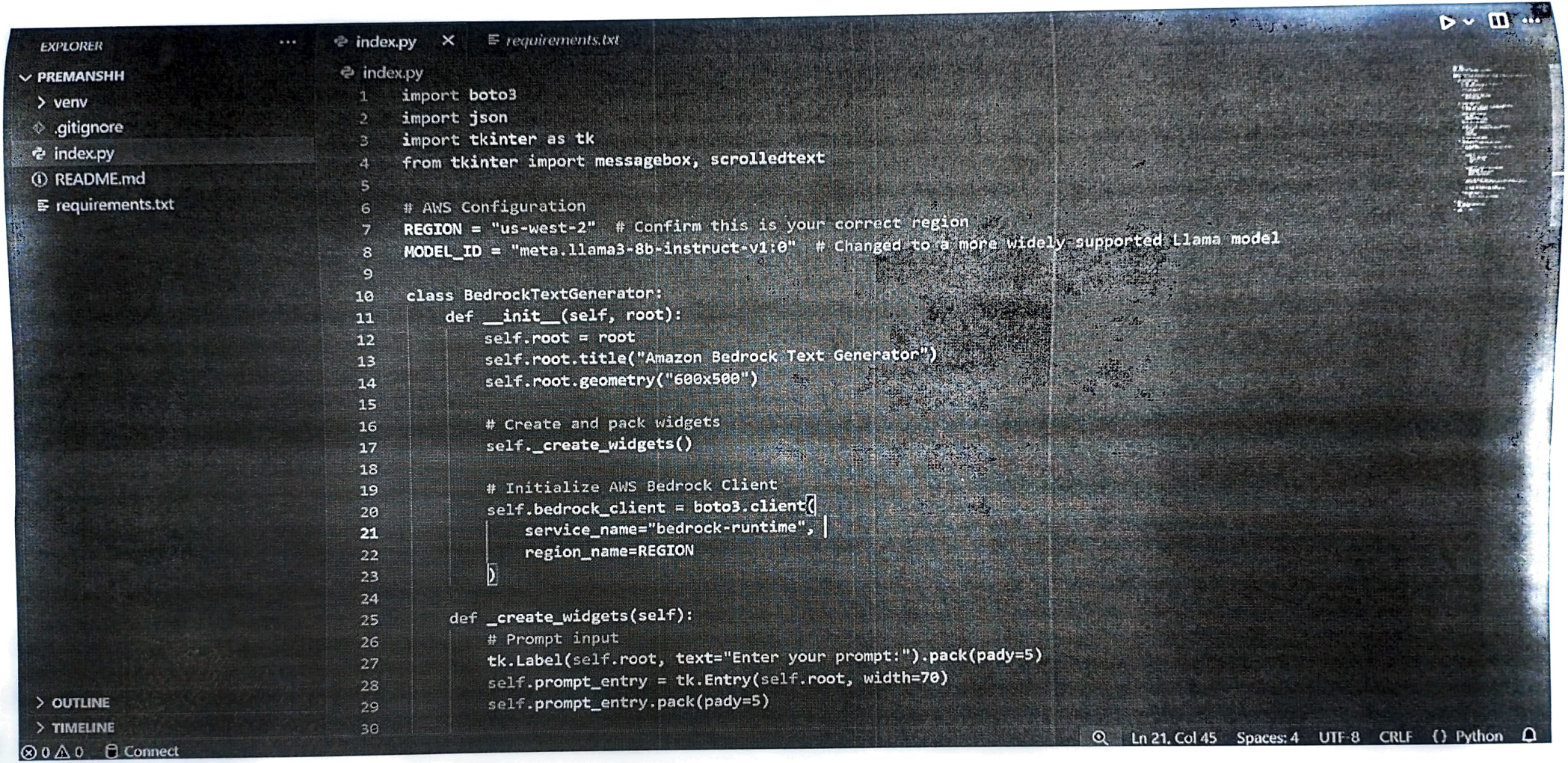
1. Create a requirements.txt file to include all necessary packages:
2. boto3
3. awscli
4. tkinter
    - boto3 is used to integrate Python with AWS.
    - awscli is used for managing AWS services via the command line.
5. Install **Anaconda** and set the bin path in the system environment variables.
6. Create a virtual environment using the following command:
7. conda create -p myenv python=3.12 -y
8. Activate the virtual environment:
9. conda activate myenv
10. Write the Python code (provided in the project). Ensure that you set the correct **model ID** and **region** to make the model accessible.
11. To find the **Model ID**, go to Amazon Bedrock → Select the model → Click on **Info** to view the model ID.
    - For **Region**, check the top-right corner of the AWS profile.
12. Configure AWS using the following command:
13. aws configure

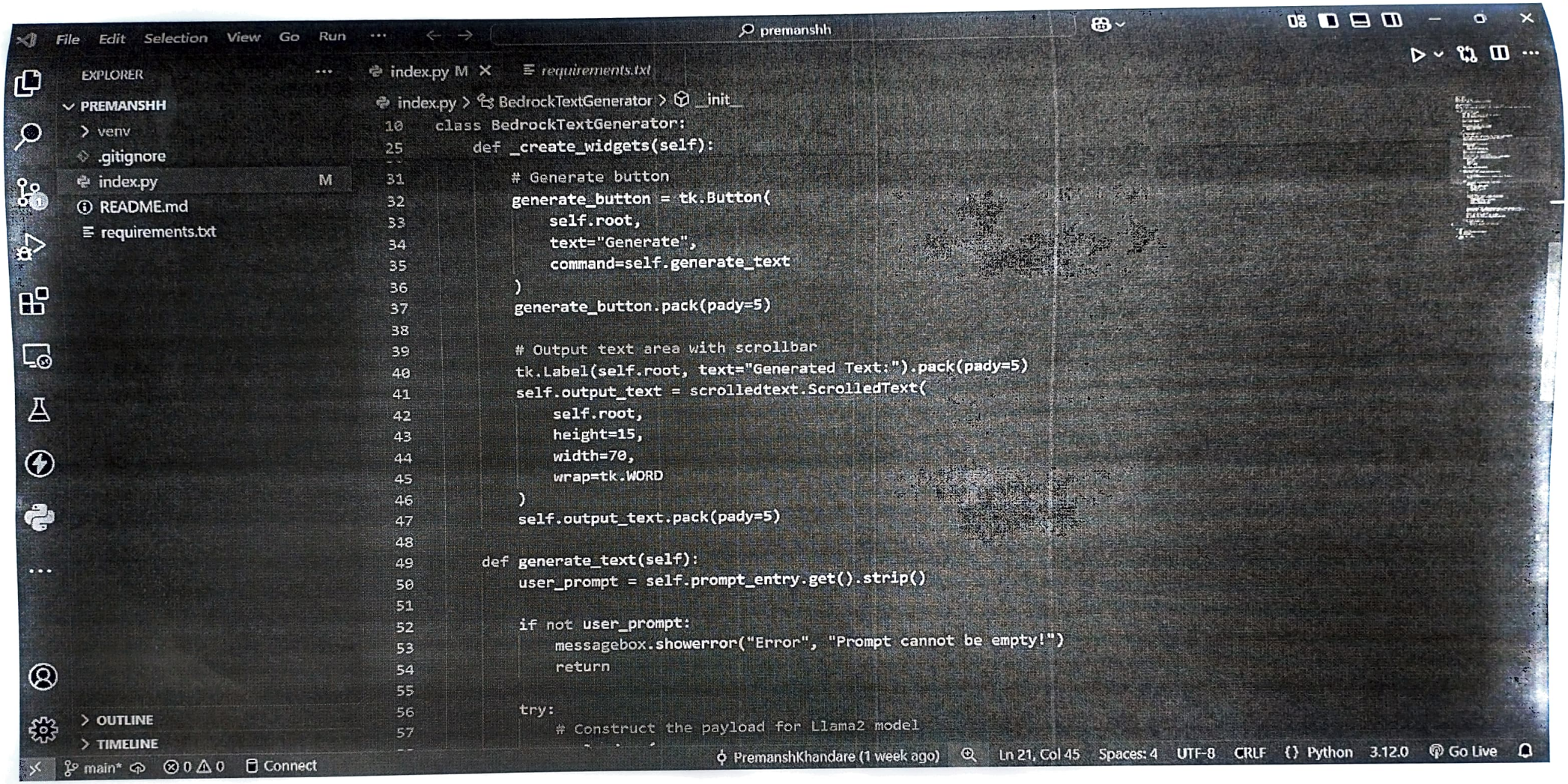
- Enter **Access Key ID** and **Secret Access Key** obtained from the AWS IAM console.

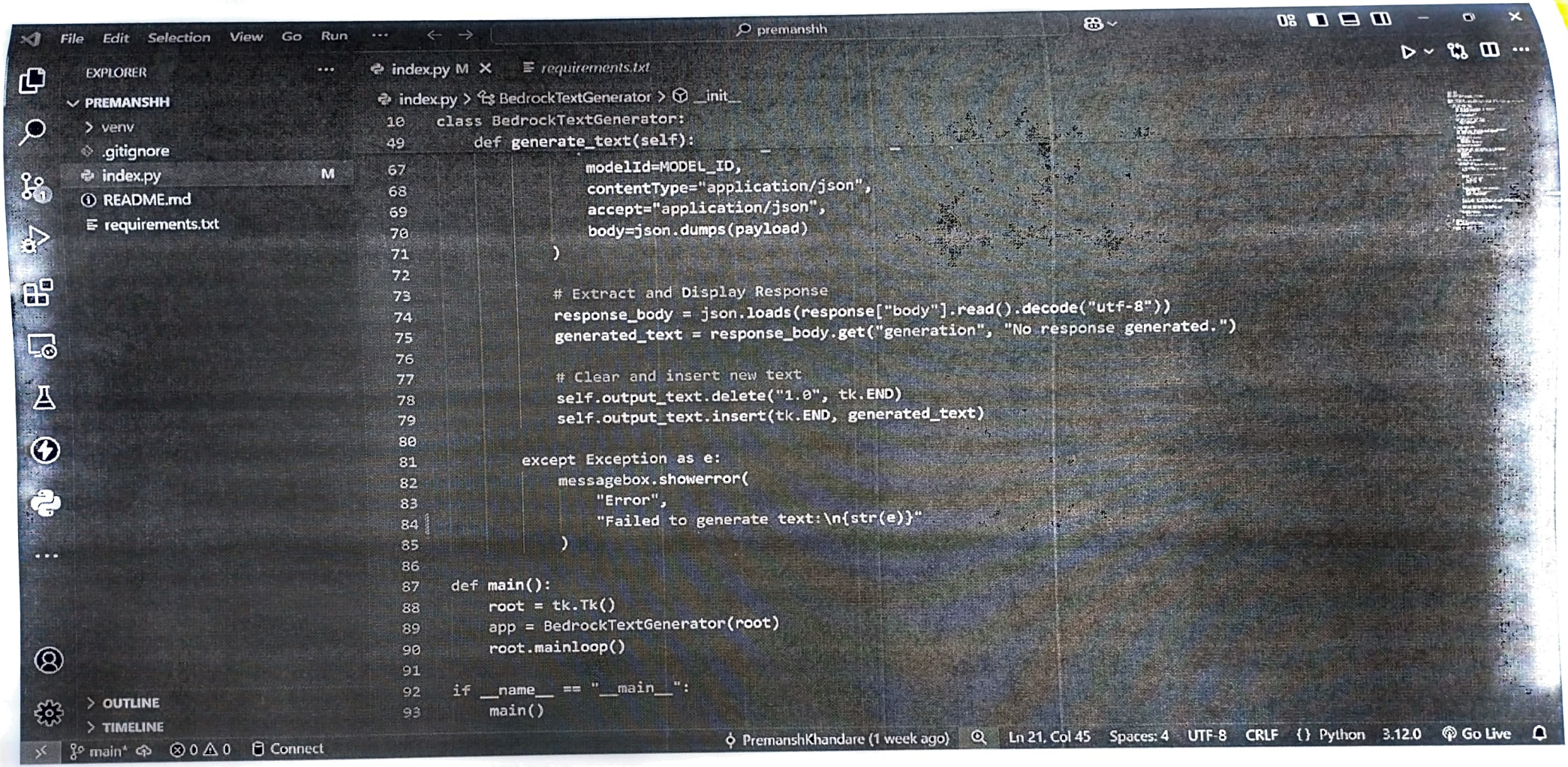
14. To generate credentials:
    - Go to **AWS IAM Console** → **Security Credentials** → **Create Access Key** → Download and save the keys.
15. After configuring AWS, make sure to enter the **Access Key ID**, **Secret Key**, **region**, and set the data type as 'json'.
16. Run the Python script using:

```
python bedrock_gui.py
```

- This will open a GUI where you can enter a prompt and receive text generated using AWS Bedrock.


EXPLORER
PREMANSHH
venv
.gitignore
index.py
README.md
requirements.txt
OUTLINE
TIMELINE
index.py
requirements.txt
index.py
import boto3
import json
import tkinter as tk
from tkinter import messagebox, scrolledtext
# AWS Configuration
REGION = "us-west-2" # Confirm this is your correct region
MODEL_ID = "meta.llama3-8b-instruct-v1:0" # Changed to a more widely supported Llama model
class BedrockTextGenerator:
def __init__(self, root):
self.root = root
self.root.title("Amazon Bedrock Text Generator")
self.root.geometry("600x500")
# Create and pack widgets
self._create_widgets()
# Initialize AWS Bedrock Client
self.bedrock_client = boto3.client(
service_name="bedrock-runtime",
region_name=REGION
)
def _create_widgets(self):
# Prompt input
tk.Label(self.root, text="Enter your prompt:").pack(pady=5)
self.prompt_entry = tk.Entry(self.root, width=70)
self.prompt_entry.pack(pady=5)
Ln 21, Col 45
Spaces: 4
UTF-8
CRLF
Python
0 0
Connect


File Edit Selection View Go Run
premanshh
EXPLORER
index.py M
requirements.txt
PREMANSHH
venv
.gitignore
index.py M
README.md
requirements.txt
index.py > BedrockTextGenerator > _init_
class BedrockTextGenerator:
def _create_widgets(self):
# Generate button
generate_button = tk.Button(
self.root,
text="Generate",
command=self.generate_text
)
generate_button.pack(pady=5)
# Output text area with scrollbar
tk.Label(self.root, text="Generated Text:").pack(pady=5)
self.output_text = scrolledtext.ScrolledText(
self.root,
height=15,
width=70,
wrap=tk.WORD
)
self.output_text.pack(pady=5)
def generate_text(self):
user_prompt = self.prompt_entry.get().strip()
if not user_prompt:
messagebox.showerror("Error", "Prompt cannot be empty!")
return
try:
# Construct the payload for Llama2 model
OUTLINE
TIMELINE
main*
0 0
Connect
PremanshKhandare (1 week ago)
Ln 21, Col 45
Spaces: 4
UTF-8
CRLF
Python
3.12.0
Go Live


File Edit Selection View Go Run
premanshh
EXPLORER
PREMANSHH
venv
.gitignore
index.py M
README.md
requirements.txt
index.py M
requirements.txt
index.py > BedrockTextGenerator > __init__
class BedrockTextGenerator:
def generate_text(self):
modelId=MODEL_ID,
contentType="application/json",
accept="application/json",
body=json.dumps(payload)
)
# Extract and Display Response
response_body = json.loads(response["body"].read().decode("utf-8"))
generated_text = response_body.get("generation", "No response generated.")
# Clear and insert new text
self.output_text.delete("1.0", tk.END)
self.output_text.insert(tk.END, generated_text)
except Exception as e:
messagebox.showerror(
"Error",
"Failed to generate text:\n{str(e)}"
)
def main():
root = tk.Tk()
app = BedrockTextGenerator(root)
root.mainloop()
if __name__ == "__main__":
main()
OUTLINE
TIMELINE
main*
0 0
Connect
PremanshKhandare (1 week ago)
Ln 21, Col 45
Spaces: 4
UTF-8
CRLF
{} Python
3.12.0
Go Live

Enter your prompt:

Largest river of India?

Generate

Generated Text:

**
A. Ganga
B. Yamuna
C. Brahmaputra
D. Indus

Answer: C. Brahmaputra

Explanation: The Brahmaputra River is the largest river of India in terms of discharge volume. It originates in Tibet and flows through India, where it meets the Ganges River at Farraka, West Bengal. The river is approximately 2,900 km (1,800 miles) long and drains an area of about 580,000 square kilometers (224,000 square miles). Its discharge volume is around 19,000 cubic meters per second (670,000 cu ft/s), making it the largest river in India. The Ganges River, on the