from flask import Flask, request, render\_template\_string

import requests

import boto3

from googletrans import Translator

import os

from huggingface\_hub import hf\_api

app = Flask(\_name\_)

# Hugging Face API URL and API Key

API\_URL = "https://api-inference.huggingface.co/models/CompVis/stable-diffusion-v1-4" # Replace with your selected model API URL

HUGGINGFACE\_API\_KEY = os.getenv("hf\_FvMaFViSHdHxogILeeHGIkmyaTXGvOLVXN") # Store in environment variables

api = hf\_api.HfApi()

api.set\_access\_token(token)

if not HUGGINGFACE\_API\_KEY:

raise-ValueError("Hugging-Face-API-Key-not-found. .-Set-the HUGGINGFACE\_API\_KEY environment variable.")

headers = {

"Authorization": f"Bearer {HUGGINGFACE\_API\_KEY}"

}

try:

response = requests.post(

API\_URL,

headers=headers,

data = {

"inputs": "A description of what you want to generate an image of",

"parameters": {

"num\_images": 2

}

}

)

response.raise\_for\_status()

print("Response JSON:", response.json())

except requests.exceptions.HTTPError as e:

print("HTTPError:", e.response.status\_code, e.response.text)

except Exception as e:

print("Error:", str(e))

# AWS S3 Configuration

AWS\_ACCESS\_KEY = os.getenv("AKIAT4GVRKB4FAU46RMN") # Store in environment variables

AWS\_SECRET\_KEY = os.getenv("vw8SIJ5fNzfaXohRFAp+7qyrH1eIe2afYR10Vt0M") # Store in environment variables

S3\_BUCKET\_NAME = "myawsbuc-1223" # Replace with your bucket name

if not AWS\_ACCESS\_KEY or not AWS\_SECRET\_KEY:

raise ValueError("AWS credentials not found. Set the AWS\_ACCESS\_KEY\_ID and AWS\_SECRET\_ACCESS\_KEY environment variables.")

# Initialize S3 client

s3\_client = boto3.client(

"s3",

aws\_access\_key\_id=AWS\_ACCESS\_KEY,

aws\_secret\_access\_key=AWS\_SECRET\_KEY

)

# Example token error check

try:

# Example API call to Hugging Face or S3 interaction for debugging

print("Testing S3 connection...")

response = s3\_client.list\_buckets()

print("S3 Buckets:", response["Buckets"])

except Exception as e:

print("Error with token or credentials:", str(e))

# Initialize the Translator

translator = Translator()

@app.route('/generate-image', methods=['GET', 'POST'])

def home():

img\_url = None

if request.method == 'POST':

prompt = request.form['text']

language = request.form['language']

# Translate the prompt to English if necessary

try:

if language != 'english':

translated = translator.translate(prompt, src=language, dest='en')

prompt = translated.text

except Exception as e:

return f"Error in translation: {e}"

# Prepare the payload

payload = {"inputs": prompt}

# Send the API request

try:

response = requests.post(API\_URL, headers=headers, json=payload)

if response.status\_code == 200:

# Save the image locally

image\_filename = "generated\_image.png"

with open(image\_filename, "wb") as f:

f.write(response.content)

# Upload the image to S3

try:

s3\_key = f"generated\_images/{image\_filename}"

s3\_client.upload\_file(image\_filename, S3\_BUCKET\_NAME, s3\_key)

img\_url = f"https://{S3\_BUCKET\_NAME}.s3.amazonaws.com/{s3\_key}"

except Exception as e:

return f"Failed to upload to S3: {e}"

return render\_template\_string(f"""

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Text to Image Generator</title>

</head>

<body>

<h2>Image Generated Successfully!</h2>

<img src="{img\_url}" alt="Generated Image" style="max-width:100%;"><br>

<p>Image saved to S3. <a href="{img\_url}" target="\_blank">View in S3</a></p>

<a href="/">Generate Another Image</a>

</body>

</html>

""")

else:

return f"Error: {response.status\_code}, {response.text}"

except requests.exceptions.RequestException as e:

return f"Request failed: {e}"

# Render the HTML template with the form

return render\_template\_string("""

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Text to Image Generator</title>

<style>

/\* Background image and page styling \*/

body {

font-family: baguet script, sans-serif;

background: url('https://st5.depositphotos.com/2274151/65350/i/450/depositphotos\_653505068-stock-photo-abstract-pastel-colors-smoke-background.jpg') no-repeat center center fixed;

background-size: cover;

height: 100vh;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

text-align: center;

margin: 0;

}

/\* Header style \*/

h1 {

color: black;

font-size: 3em;

margin-top: 20px;

}

/\* Form container \*/

.form-container {

background-color: golden; /\* semi-transparent background \*/

padding: 20px;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

width: 400px;

text-align: center;

}

/\* Input fields and select dropdown \*/

input[type="text"], select {

padding: 10px;

width: 100%;

margin-bottom: 20px;

border-radius: 4px;

border: 1px solid #ddd;

font-size: 1em;

text-align: center; /\* Center the text in the input field \*/

}

input[type="text"] {

background-color: white; /\* Remove background color \*/

}

/\* Submit button \*/

button {

padding: 12px 30px;

background-color: white;

color: black;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 1.2em;

}

button:hover {

background-color: white;

}

/\* Generated image section \*/

.generated-image {

margin-top: 20px;

}

.generated-image img {

max-width: 100%;

border-radius: 8px;

margin-top: 20px;

}

/\* Responsive design for mobile \*/

@media (max-width: 768px) {

.form-container {

width: 90%;

}

h1 {

font-size: 2em;

}

}

</style>

</head>

<body>

<h1>Text to Image Generator</h1>

<div class="form-container">

<form action="/" method="POST">

<label for="text">Enter Prompt:</label><br>

<input type="text" id="text" name="text" required><br><br>

<label for="language">Select Language:</label><br>

<select id="language" name="language" required>

<option value="english">English</option>

<option value="kn">Kannada</option>

<option value="ta">Tamil</option>

<option value="te">Telugu</option>

</select><br><br>

<button type="submit">Generate Image</button>

</form>

</div>

{% if img\_url %}

<div class="generated-image">

<h2>Generated Image:</h2>

<img src="{{ img\_url }}" alt="Generated Image">

</div>

{% endif %}

</body>

</html>

""")

if \_name\_ == '\_main\_':

app.run(app.run(host='0.0.0.0', port=8000, debug=True)