from flask import Flask, request, render\_template, redirect, jsonify, session, render\_template\_string

import random

import string

import requests

import pyperclip

import urllib

from urllib.parse import unquote

from flask\_cors import CORS

from flask import session

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.application import MIMEApplication

app = Flask(\_\_name\_\_, static\_url\_path='/static')

CORS(app)

app.secret\_key = b'5#y2L"F4Q8z\n\xec]/'

all\_users = {}

question\_data = None

score = 0

random\_code = 0

music\_thread=None

my\_email = "15rishivejani@gmail.com"

app\_password = "wzmpddvkfhhmwhhj"

def generate\_random\_code(length):

characters = string.ascii\_letters + string.digits

random\_code = ''.join(random.choice(characters) for \_ in range(length))

return random\_code

random\_code = generate\_random\_code(5)

@app.route('/copy\_code', methods=['POST'])

def copy\_code():

pyperclip.copy(random\_code)

return '', 204

@app.route('/', methods=["GET", "POST"])

def r():

return render\_template('room.html')

@app.route('/check\_answer', methods=["POST"])

def check\_answer():

data = request.json

question\_index = int(data.get('questionIndex'))

selected\_answer = data.get('answer')

score = session.get('score', 0)

is\_correct = False

if question\_data and 0 <= question\_index < len(question\_data):

correct\_answer = question\_data[question\_index]['correct\_answer']

is\_correct = selected\_answer == correct\_answer

if is\_correct:

score += 1

session['score'] = score

return jsonify({'is\_correct': is\_correct, 'score': score})

@app.route('/data\_question', methods=["GET"])

def data\_question():

if question\_data:

return jsonify(question\_data)

else:

return jsonify({'error': 'Question data not available'})

@app.route('/register', methods=["GET", "POST"])

def reg():

if request.method == "POST":

name = request.form["userName"]

session['username'] = name

code = request.form["sc"]

if code == random\_code:

all\_users[name] = {'score': 0}

return redirect("/questions")

else:

return render\_template('register.html')

return render\_template('register.html')

@app.route('/gg', methods=["GET", "POST"])

def gg():

return redirect("/generate\_quiz")

@app.route('/generate\_quiz', methods=["GET", "POST"])

def gg1():

global question\_data

if request.method == "POST":

amount = request.form["amount"]

category = request.form["category"]

difficulty = request.form["difficulty"]

q\_type = request.form["type"]

parameters = {

"amount": amount,

"category": category,

"difficulty": difficulty,

"type": q\_type,

"encode": "url3986"

}

response = requests.get("https://opentdb.com/api.php", params=parameters)

data = response.json()

question\_data = data.get("results", [])

return redirect("/questions")

return render\_template("quiz\_generator.html")

@app.route('/admin', methods=["GET", "POST"])

def a():

global random\_code

if request.method == "POST":

name = request.form["userName"]

all\_users[name] = {'score': 0}

return redirect("/admin")

return render\_template('admin.html', all\_users=all\_users, code=random\_code)

questions = []

corr\_answers = []

filename = 'quiz\_results.txt'

def save\_questions\_to\_file():

with open(filename, 'w',encoding='utf-8') as file:

for i in range(len(questions)):

question = questions[i]

correct\_answer = corr\_answers[i]

line = f"Question: {question}\nCorrect Answer: {correct\_answer}\n\n"

file.write(line)

@app.route('/questions', methods=["GET", "POST"])

def ques():

global score

if question\_data:

decoded\_questions = []

for question in question\_data:

decoded\_question = {}

decoded\_question["question"] = unquote(question["question"])

questions.append(decoded\_question["question"])

decoded\_question["correct\_answer"] = unquote(question["correct\_answer"])

corr\_answers.append(decoded\_question["correct\_answer"])

decoded\_question["incorrect\_answers"] = [unquote(answer) for answer in question["incorrect\_answers"]]

decoded\_question["type"] = unquote(question["type"])

decoded\_question["difficulty"] = unquote(question["difficulty"])

decoded\_question["category"] = unquote(question["category"])

answers = [decoded\_question["correct\_answer"]] + decoded\_question["incorrect\_answers"]

random.shuffle(answers)

decoded\_question["shuffled\_answers"] = answers

decoded\_questions.append(decoded\_question)

return render\_template('display\_questions.html', questions=decoded\_questions, score=score)

else:

return render\_template('waiting.html')

@app.route('/share\_on\_whatsapp', methods=['POST'])

def share\_on\_whatsapp():

code = request.form.get('code')

l1 = ["Check out this code:"]

l2 = ["Here is the link to join the room: https://1rlsn44x-5000.inc1.devtunnels.ms/"]

message = f"{l1[0]} {code}\n{l2[0]}"

encoded\_message = urllib.parse.quote(message)

url = f'https://api.whatsapp.com/send?text={encoded\_message}'

return redirect(url)

def send\_email\_with\_attachment(email, subject, body, file\_path):

msg = MIMEMultipart()

msg['From'] = my\_email

msg['To'] = email

msg['Subject'] = subject

msg.attach(MIMEText(body, 'plain'))

with open(file\_path, 'rb') as file:

attachment = MIMEApplication(file.read(), Name=file\_path)

attachment.add\_header('Content-Disposition', f'attachment; filename="{file\_path}"')

msg.attach(attachment)

try:

with smtplib.SMTP("smtp.gmail.com", port=587) as connection:

connection.starttls()

connection.login(user=my\_email, password=app\_password)

connection.sendmail(from\_addr=my\_email, to\_addrs=email, msg=msg.as\_string())

print(f"Email with attachment sent successfully to {email}")

except Exception as e:

print(f"Failed to send email: {e}")

@app.route('/leaderboard', methods=['GET','POST'])

def leaderboard():

save\_questions\_to\_file()

if request.method == "POST":

email = request.form["mail"]

file\_path = "quiz\_results.txt"

subject = "Quizify"

body = "Here is the correct answers of the quiz!"

send\_email\_with\_attachment(email, subject, body, file\_path)

sorted\_users = sorted(all\_users.items(), key=lambda x: x[1]['score'], reverse=True)

print("Sorted Users:", sorted\_users)

return render\_template('leaderboard.html', users=sorted\_users)

@app.route('/byebye', methods=['POST'])

def byebye():

if request.method == "POST":

data = request.json

username = session.get('username')

final\_score = data.get('final\_score')

if username:

all\_users.setdefault(username, {'score': 0})['score'] = final\_score

print("Final score for", username, ":", final\_score)

print("All users and their scores:")

for user, info in all\_users.items():

print(user, ":", info['score'])

return jsonify({'success': True}), 200

else:

return jsonify({'success': False, 'error': 'Username not found in session'}), 404

return jsonify({'success': False}), 405

def clear\_all\_users():

global all\_users

all\_users.clear()

session.pop('username', None)

session.pop('score', None)

def restart\_game():

global question\_data, random\_code

question\_data = None

random\_code = generate\_random\_code(5)

clear\_all\_users()

@app.route('/restart-game', methods=['POST'])

def restart\_game\_endpoint():

clear\_all\_users()

restart\_game()

return jsonify({'statu s': 'success'}), 200

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)