```python
from flask import Flask, render_template, request, redirect, url_for, session,
flash
from flask_mysqldb import MySQL
from flask_mail import Mail, Message
from werkzeug.security import generate_password_hash, check_password_hash
from config import Config
import os
from werkzeug.utils import secure_filename

app = Flask(__name__)
app.config.from_object(Config)

mysql = MySQL(app)
mail = Mail(app)

app.config['UPLOAD_FOLDER'] = 'uploads'  # Define the upload folder
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024  # Limit file size to 16MB


os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/services')
def services():
    return render_template('services.html')

@app.route('/contact_us')
def contact_us():
    return render_template('contact-us.html')

@app.route('/our_gallery')
def our_gallery():
    return render_template('our-gallery.html')

@app.route('/select_role')
def select_role():
    return render_template('select_role.html')








disorder_to_specialist = {
    "Stuttering": ["SLP"],  # Updated to match the database
    "Voice Disorder": ["ENT Specialist"],  # Updated to match the database
    "Neurological Speech Issue": ["Neurologist"],
    "Pediatric Speech Delay": ["Pediatrician"],
    "Autism Communication Issue": ["Psychologist"],
    "Post-Surgery Recovery": ["Rehabilitation Specialist"],
    "Hearing and Speech Issue": ["Audiologist"],
}


# Patient Signup
```

```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        try:
            # ▫ Debugging: Print Form Data
            print("➡ Signup Form Submitted")
            print(request.form)

            patient_name = request.form['patient-name']
            age = request.form['age']
            gender = request.form['gender']
            parents_name = request.form['parents-name']
            contact_no = request.form['contact-no']
            email = request.form['email']
            password = request.form['password']
            address = request.form['address']
            disorder = request.form['disorder']

            # ▫ Debugging: Check Values
            print(f"📌 Name: {patient_name}, Email: {email}, Disorder:
{disorder}")

            # Hash the password before storing
            hashed_password = generate_password_hash(password)

            cur = mysql.connection.cursor()
            cur.execute("""
                INSERT INTO patients (patient_name, age, gender, parents_name,
contact_no, email, password, address, disorder)
                VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
            """, (patient_name, age, gender, parents_name, contact_no, email,
hashed_password, address, disorder))

            mysql.connection.commit()
            cur.close()

            flash("✅ Signup successful! Please log in.", "success")
            print("✅ User registered successfully!")  # Debugging
            return redirect(url_for('login'))

        except Exception as e:
            print(f"❌ Error: {str(e)}")  # Debugging
            flash(f"⚠ Error: {str(e)}", "danger")
            return redirect(url_for('signup'))

    return render_template('signup.html')


# Doctor Signup
@app.route('/doctor_signup', methods=['GET', 'POST'])
def doctor_signup():
    if request.method == 'POST':
        clinic_name = request.form['clinic_name']
        doctor_name = request.form['doctor_name']
        age = request.form['age']
        gender = request.form['gender']
        specialty = request.form['specialty']
        experience = request.form['experience']
        license_certificate_no = request.form['license_certificate_no']
        address = request.form['address']
        phone = request.form['phone']
        email = request.form['email']
        password = generate_password_hash(request.form['password'])  # Hashing
Password
```

```
        try:
            cur = mysql.connection.cursor()
            cur.execute("""
                INSERT INTO doctors (clinic_name, doctor_name, age, gender,
specialty, experience, license_certificate_no, address, phone, email, password)
                VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
            """, (clinic_name, doctor_name, age, gender, specialty, experience,
license_certificate_no, address, phone, email, password))
            mysql.connection.commit()
            cur.close()

            flash("✅ Signup Successful! Redirecting to Select Doctor...",
"success")
            print("✅ Doctor registered successfully! Redirecting to login...")
# Debugging
            return redirect(url_for('doctor_login'))  # Redirect to Doctor Login
Page

        except Exception as e:
            print(f"❌ Error during doctor signup: {e}")  # Debugging
            flash(f"❌ Error: {e}", "danger")
            return redirect(url_for('doctor_signup'))  # Redirect back to signup
page on error

    return render_template('doctor_signup.html')




@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        cur = mysql.connection.cursor()
        cur.execute("SELECT * FROM patients WHERE email = %s", (email,))
        user = cur.fetchone()
        cur.close()

        if user:
            stored_password = user[7]  # Assuming the password is in the 7th
column (index 6)

            if check_password_hash(stored_password, password):  # Validate
password
                session['loggedin'] = True  # Set session variable
                session['role'] = 'patient'  # Set role
                session['user_id'] = user[0]  # Store user ID in session
                session['user_name'] = user[1]  # Store username for display
                session['email'] = email  # Store email in session

                flash("✅ Login successful!", "success")
                print("✅ Login successful! Redirecting to select_doctor...")  #
Debugging
                return redirect(url_for('patient_dashboard'))  # Redirect to
Select Doctor Page

            else:
                flash("⚠️ Incorrect password. Try again!", "danger")
        else:
            flash("⚠️ Email not found. Please sign up!", "danger")
    return render_template('login.html')  # Reload login page if login fails



@app.route('/doctor_login', methods=['GET', 'POST'])
def doctor_login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')

        try:
            cur = mysql.connection.cursor()
            cur.execute("SELECT id, doctor_name, password FROM doctors WHERE
email = %s", (email,))
            doctor = cur.fetchone()
            cur.close()
        except Exception as e:
            print(f"❌ Database Error: {e}")
            flash('An error occurred. Please try again later.', 'danger')
            return render_template('doctor_login.html')

        if doctor:
            doctor_id, doctor_name, stored_password = doctor

            print(f"✅ Doctor Found: {doctor}")
            print(f"🔍 Stored Password: {stored_password}")
            print(f"🔍 Entered Password: {password}")

            if stored_password and check_password_hash(stored_password,
password):
                session['loggedin'] = True
                session['email'] = email
                session['role'] = 'doctor'
                session['doctor_id'] = doctor_id
                session['doctor_name'] = doctor_name
                print("✅ Login Successful!")
                return redirect(url_for('doctor_dashboard'))
            else:
                print("❌ Invalid Password")
                flash('Invalid password. Please try again.', 'danger')
        else:
            print("❌ Doctor Not Found")
            flash('Doctor not found. Please sign up.', 'danger')

    return render_template('doctor_login.html')


# Select Doctor Based on Disease
@app.route('/select_doctor', methods=['GET'])
def select_doctor():
    if 'loggedin' not in session or session['role'] != 'patient':
        return redirect(url_for('login'))

    cur = mysql.connection.cursor()

    # Fetch the patient's disorder
    cur.execute("SELECT disorder FROM patients WHERE email = %s",
(session['email'],))
    patient_disorder = cur.fetchone()

    if patient_disorder:
        disorder = patient_disorder[0]
        print(f"DEBUG: Patient's Disorder = {disorder}")  # Debugging
        # Get the relevant specializations for the patient's disorder
        specializations = disorder_to_specialist.get(disorder, [])
        print(f"DEBUG: Mapped Specializations = {specializations}")  # Debugging

        if specializations:
            # Fetch doctors whose specialty matches the relevant specializations
            query = "SELECT id, doctor_name, specialty, experience, degree,fees 
FROM doctors WHERE specialty IN %s"
            print(f"DEBUG: Executing Query = {query % 
(tuple(specializations),)}")  # Debugging
            cur.execute(query, (tuple(specializations),))
            doctors = cur.fetchall()
            print(f"DEBUG: Doctors Found = {doctors}")  # Debugging
        else:
            doctors = []
            flash("No specializations found for your disorder.", "warning")
    else:
        doctors = []
        flash("No disorder found for the patient.", "danger")

    cur.close()

    return render_template('select_doctor.html', doctors=doctors)

@app.route('/upload_plan/<int:patient_id>', methods=['POST'])
def upload_plan(patient_id):
    if 'loggedin' not in session or session['role'] != 'doctor':
        return redirect(url_for('login'))

    if 'therapy_plan' not in request.files:
        flash("No file uploaded!", "danger")
        return redirect(url_for('doctor_dashboard'))

    file = request.files['therapy_plan']
    if file.filename == '':
        flash("No file selected!", "danger")
        return redirect(url_for('doctor_dashboard'))

    filename = secure_filename(file.filename)
    file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)

    cur = mysql.connection.cursor()
    try:
        file.save(file_path)

        # ✅ Insert Therapy Plan into the Database
        cur.execute("""
            INSERT INTO therapy_plans (patient_id, doctor_id, file_path, 
upload_date)
            VALUES (%s, %s, %s, NOW())
        """, (patient_id, session['doctor_id'], file_path))
        mysql.connection.commit()

        # ✅ Update Appointment Status
        cur.execute("""
            UPDATE appointments
            SET status = 'Completed'
            WHERE patient_id = %s AND doctor_id = %s
        """, (patient_id, session['doctor_id']))
        mysql.connection.commit()

        flash("Therapy plan uploaded successfully!", "success")
    except Exception as e:
        flash(f"Error uploading file: {e}", "danger")
    finally:
        cur.close()

    return redirect(url_for('doctor_dashboard'))


from flask import send_from_directory

@app.route('/download_plan/<string:plan_name>')
def download_plan(plan_name):
    if 'loggedin' not in session or session['role'] != 'patient':
        return redirect(url_for('login'))

    # Fetch the file path from the database
    cur = mysql.connection.cursor()
    try:
        cur.execute("""
            SELECT file_path
            FROM therapy_plans
            WHERE plan_name = %s AND patient_id = %s
        """, (plan_name, session['user_id']))
        file_path = cur.fetchone()[0]
    except Exception as e:
        print(f"DEBUG: Error fetching file path: {e}")
        flash("Error downloading file.", "danger")
        return redirect(url_for('patient_dashboard'))
    finally:
        cur.close()

    # Send the file for download
    return send_from_directory(app.config['UPLOAD_FOLDER'],
os.path.basename(file_path), as_attachment=True)

# Route to start a session
@app.route('/start_session/<int:appointment_id>', methods=['POST'])
def start_session(appointment_id):
    if 'loggedin' not in session or session['role'] != 'doctor':
        return redirect(url_for('login'))

    meeting_link = request.form['meeting_link']
    time_slot = request.form['time_slot']

    print(f"DEBUG: Meeting Link = {meeting_link}")
    print(f"DEBUG: Time Slot = {time_slot}")

    cur = mysql.connection.cursor()
    cur.execute("UPDATE appointments SET session_link = %s, time_slot = %s,
status = 'Ongoing' WHERE id = %s", (meeting_link, time_slot, appointment_id))
    mysql.connection.commit()
    cur.close()

    flash("Session started successfully!", "success")
    return redirect(url_for('doctor_dashboard'))

# Route to get today's sessions
@app.route('/todays_sessions')
def todays_sessions():
    if 'doctor_id' not in session:
        return redirect(url_for('login'))

    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM appointments WHERE doctor_id=%s AND
DATE(appointment_date) = CURDATE()", (session['doctor_id'],))
    sessions = cur.fetchall()
    cur.close()

    return render_template('todays_sessions.html', sessions=sessions)

# Route to submit a meeting link
@app.route('/submit_meeting', methods=['POST'])
def submit_meeting():
    if 'doctor_id' not in session:
        return redirect(url_for('login'))

    appointment_id = request.form['appointment_id']
    meeting_link = request.form['meeting_link']

    cur = mysql.connection.cursor()
    cur.execute("UPDATE appointments SET session_link=%s WHERE id=%s",
(meeting_link, appointment_id))
    mysql.connection.commit()
    cur.close()

    flash('Meeting link submitted successfully!', 'success')
    return redirect(url_for('doctor_dashboard'))

# Request Appointment
@app.route('/request_appointment/<int:doctor_id>', methods=['POST'])
def request_appointment(doctor_id):
    if 'loggedin' not in session or session['role'] != 'patient':
        return redirect(url_for('login'))

    cur = mysql.connection.cursor()

    # 🔹 Fetch patient details
    cur.execute("SELECT id, patient_name FROM patients WHERE email = %s",
(session['email'],))
    patient = cur.fetchone()

    # 🔹 Fetch doctor details (including email)
    cur.execute("SELECT doctor_name, email FROM doctors WHERE id = %s",
(doctor_id,))
    doctor = cur.fetchone()

    if patient and doctor:
        patient_id = patient[0]
        patient_name = patient[1]
        doctor_name = doctor[0]
        doctor_email = doctor[1]  # 🔹 Fetch email dynamically from DB

        # 🔹 Insert the appointment with a "Pending" status
        cur.execute("INSERT INTO appointments (patient_id, doctor_id, status)
VALUES (%s, %s, 'Pending')",
                    (patient_id, doctor_id))
        mysql.connection.commit()

        # 🔹 Send email to the doctor
        msg = Message("New Appointment Request",
                      sender="mrshreyash08@gmail.com",
                      recipients=[doctor_email])  # 🔹 Automatically fetched
email
        msg.body = f"Dear Dr. {doctor_name},\n\nYou have received a new
appointment request from {patient_name}.\n\nPlease log in to your dashboard to
respond.\n\nBest Regards,\nTherapyTalk Team"

        try:
            mail.send(msg)
            flash('Appointment request sent! The doctor has been notified via
email.', 'success')
        except Exception as e:
            flash(f'Appointment requested, but email could not be sent:
{str(e)}', 'warning')

    cur.close()
    return redirect(url_for('patient_dashboard'))


# Patient Dashboard
@app.route('/patient_dashboard')
def patient_dashboard():
    print("DEBUG: Entering patient_dashboard route")
    if 'loggedin' not in session or session['role'] != 'patient':
        print("DEBUG: Not logged in or incorrect role. Redirecting to login.")
        return redirect(url_for('login'))

    print(f"DEBUG: Session Data - {session}")
    cur = mysql.connection.cursor()

    try:
        # Fetch patient ID and name
        cur.execute("SELECT id, patient_name, disorder FROM patients WHERE email
= %s", (session['email'],))
        patient = cur.fetchone()

        if not patient:
            print("DEBUG: Patient not found. Redirecting to login.")
            flash("Patient not found!", "danger")
            return redirect(url_for('login'))

        patient_id = patient[0]  # Index 0: id
        patient_name = patient[1]  # Index 1: patient_name
        patient_disorder = patient[2]  # Index 2: disorder

        print(f"DEBUG: Logged-in Patient ID = {patient_id}")
        print(f"DEBUG: Patient's Disorder = {patient_disorder}")

        # Fetch therapist name (selected therapist)
        cur.execute("""
            SELECT d.doctor_name
            FROM appointments a
            JOIN doctors d ON a.doctor_id = d.id
            WHERE a.patient_id = %s AND a.status = 'Ongoing'
            LIMIT 1
        """, (patient_id,))
        therapist = cur.fetchone()
        therapist_name = therapist[0] if therapist else "No Therapist Selected"
# Index 0: doctor_name

        # Fetch appointment statistics
        cur.execute("SELECT COUNT(*) FROM appointments WHERE patient_id = %s",
(patient_id,))
        total_requests = cur.fetchone()[0]  # Index 0: count

        cur.execute("SELECT COUNT(*) FROM appointments WHERE patient_id = %s AND
status = 'Approved'", (patient_id,))
        approved_requests = cur.fetchone()[0]  # Index 0: count

        cur.execute("SELECT COUNT(*) FROM appointments WHERE patient_id = %s AND
status = 'Completed'", (patient_id,))
        completed_sessions = cur.fetchone()[0]  # Index 0: count

        # Fetch appointments
        cur.execute("""
            SELECT a.id, d.doctor_name, d.specialty, a.status, a.payment_status,
a.admin_message
            FROM appointments a
            JOIN doctors d ON a.doctor_id = d.id
            WHERE a.patient_id = %s
        """, (patient_id,))
        appointments = cur.fetchall()

        # Fetch doctors for the Select Doctor tab
        specializations = disorder_to_specialist.get(patient_disorder, [])
        print(f"DEBUG: Mapped Specializations = {specializations}")

        if specializations:
            query = "SELECT id, doctor_name, specialty, experience, degree, fees
FROM doctors WHERE specialty IN %s"
            print(f"DEBUG: Executing Query = {query %
(tuple(specializations),)}")
            cur.execute(query, (tuple(specializations),))
            doctors = cur.fetchall()
            print(f"DEBUG: Doctors Found = {doctors}")
        else:
            doctors = []
            flash("No specializations found for your disorder.", "warning")

        # Fetch session details
        cur.execute("""
            SELECT d.doctor_name, a.appointment_date, a.time_slot,
a.session_link
            FROM appointments a
            JOIN doctors d ON a.doctor_id = d.id
            WHERE a.patient_id = %s AND a.status = 'Ongoing'
        """, (patient_id,))
        session_details = cur.fetchall()

        # Fetch approved doctors for the Book Appointment section
        cur.execute("""
            SELECT d.id, d.doctor_name, d.specialty, d.experience, d.degree,
d.fees
            FROM appointments a
            JOIN doctors d ON a.doctor_id = d.id
            WHERE a.patient_id = %s AND a.status = 'Approved'
        """, (patient_id,))
        approved_doctors = cur.fetchall()
        print(f"DEBUG: Approved doctors: {approved_doctors}")
        print(f"DEBUG: Number of approved doctors: {len(approved_doctors)}")

        # Fetch therapy plans for the logged-in patient
        cur.execute("""
            SELECT tp.file_path, d.doctor_name, tp.upload_date
            FROM therapy_plans tp
            JOIN doctors d ON tp.doctor_id = d.id
            WHERE tp.patient_id = %s
        """, (patient_id,))
        therapy_plans = cur.fetchall()
        print(f"DEBUG: Therapy Plans: {therapy_plans}")

        # Fetch payment history
        cur.execute("""
            SELECT transaction_id, amount, payment_status, admin_message,
created_at
            FROM payment_history
            WHERE patient_id = %s
        """, (patient_id,))
        payment_history = cur.fetchall()
        print(f"DEBUG: Payment History: {payment_history}")

        # Prepare patient dashboard data
        patient_dashboard_data = {
            "selected_therapist": therapist_name,
            "total_requests_sent": total_requests,
            "approved_appointments": approved_requests,
            "completed_sessions": completed_sessions,
        }

        print("DEBUG: Rendering patient_dashboard.html")
        return render_template(
            'patient_dashboard.html',
            patient_dashboard_data=patient_dashboard_data,
            patient_name=patient_name,
            appointments=appointments,
            doctors=doctors,
            approved_doctors=approved_doctors,
            session_details=session_details,
            therapy_plans=therapy_plans,
            payment_history=payment_history
        )

    except Exception as e:
        print(f"DEBUG: Error in patient_dashboard: {e}")
        flash("An error occurred. Please try again.", "danger")
        return redirect(url_for('login'))  # Redirect to login instead of
patient_dashboard to avoid loops

    finally:
        cur.close()

@app.route('/activity_recommendation', methods=['POST'])
def activity_recommendation():
    if request.method == 'POST':
        disorder = request.form.get('disorder')
        duration = request.form.get('duration')

        recommendations = get_activity_recommendations(disorder)

        if recommendations:
            patient_id = session.get('user_id')  # Get patient ID from session
            patient_dashboard_data = fetch_patient_dashboard_data(patient_id)
            patient_name = session.get('user_name')
            appointments = fetch_appointments(patient_id)
            doctors = fetch_doctors(disorder)
            print("DEBUG: Reached render_template('activity_results.html')")
#add this line.
            return render_template('activity_results.html',
                                   recommendations=recommendations,
                                   disorder=disorder,
                                   duration=duration,

patient_dashboard_data=patient_dashboard_data,
                                   patient_name=patient_name,
                                   appointments=appointments,
                                   doctors=doctors)
        else:
            flash("No activity recommendations found for the selected
disorder.", 'warning')
            return redirect(url_for('patient_dashboard'))
    return redirect(url_for('patient_dashboard'))
```

```
def get_activity_recommendations(disorder):
    activity_map = {
        'Stuttering': ['Speech therapy exercises', 'Reading aloud', 'Slow
breathing exercises', 'Mindfulness meditation'],
        'Voice Disorder': ['Vocal warm-ups', 'Humidifier use', 'Hydration',
'Vocal rest'],
        'Neurological Speech Issue': ['Cognitive exercises', 'Speech therapy',
'Physical therapy', 'Occupational therapy'],
        'Pediatric Speech Delay': ['Play-based therapy', 'Interactive reading',
'Social interaction games', 'Early intervention programs'],
        'Autism Communication Issue': ['Picture Exchange Communication System
(PECS)', 'Social stories', 'Speech therapy', 'Sensory integration activities'],
        'Post-Surgery Recovery': ['Breathing exercises', 'Gentle vocal
exercises', 'Physical therapy', 'Rest'],
        'Hearing and Speech Issue': ['Sign language training', 'Lip reading',
'Auditory training', 'Speech therapy'],
        'Other': ['Consult with a specialist', 'General wellness activities']
    }
    return activity_map.get(disorder, activity_map['Other'])

def fetch_patient_dashboard_data(patient_id):
    cur = mysql.connection.cursor()
    cur.execute("SELECT COUNT(*) FROM appointments WHERE patient_id = %s",
(patient_id,))
    total_requests = cur.fetchone()[0]

    cur.execute("SELECT COUNT(*) FROM appointments WHERE patient_id = %s AND
status = 'Approved'", (patient_id,))
    approved_requests = cur.fetchone()[0]

    cur.execute("SELECT COUNT(*) FROM appointments WHERE patient_id = %s AND
status = 'Completed'", (patient_id,))
    completed_sessions = cur.fetchone()[0]
    cur.close()

    return {
        "total_requests_sent": total_requests,
        "approved_appointments": approved_requests,
        "completed_sessions": completed_sessions,
    }

def fetch_appointments(patient_id):
    cur = mysql.connection.cursor()
    cur.execute("""
        SELECT a.id, d.doctor_name, d.specialty, a.status
        FROM appointments a
        JOIN doctors d ON a.doctor_id = d.id
        WHERE a.patient_id = %s
    """, (patient_id,))
    appointments = cur.fetchall()
    cur.close()
    return appointments

def fetch_doctors(disorder):
    specializations = disorder_to_specialist.get(disorder, [])
    if specializations:
        cur = mysql.connection.cursor()
        query = "SELECT id, doctor_name, specialty, experience, degree, fees
FROM doctors WHERE specialty IN %s"
        cur.execute(query, (tuple(specializations),))
        doctors = cur.fetchall()
        cur.close()
        return doctors
    else:
        return []

from flask_mail import Message

@app.route('/accept_appointment/<int:appointment_id>', methods=['POST'])
def accept_appointment(appointment_id):
    if 'loggedin' not in session or session['role'] != 'doctor':
        return redirect(url_for('login'))

    cur = mysql.connection.cursor()

    try:
        # Debug: Print the appointment ID being processed
        print(f"DEBUG: Approving appointment ID = {appointment_id}")

        # Update the appointment status to "Approved"
        cur.execute("UPDATE appointments SET status = 'Approved' WHERE id = %s", 
(appointment_id,))
        mysql.connection.commit()

        # Debug: Verify the update
        cur.execute("SELECT status FROM appointments WHERE id = %s", 
(appointment_id,))
        updated_status = cur.fetchone()[0]
        print(f"DEBUG: Appointment {appointment_id} status updated to: 
{updated_status}")

        # Fetch patient details to send an email notification
        cur.execute("""
            SELECT p.patient_name, p.email, d.doctor_name, d.specialty
            FROM appointments a
            JOIN patients p ON a.patient_id = p.id
            JOIN doctors d ON a.doctor_id = d.id
            WHERE a.id = %s
        """, (appointment_id,))
        appointment_details = cur.fetchone()

        if appointment_details:
            patient_name, patient_email, doctor_name, specialty = 
appointment_details

            # Send email to the patient
            msg = Message(
                subject="Appointment Approved - TherapyTalk",
                sender="mrshreyash08@gmail.com",  # Replace with your email
                recipients=[patient_email]
            )
            msg.body = f"""
            Dear {patient_name},

            Your appointment request with Dr. {doctor_name} ({specialty}) has 
been approved and is now ongoing.

            Appointment Details:
            - Doctor: Dr. {doctor_name}
            - Specialty: {specialty}
            - Status: Ongoing

            Please log in to your TherapyTalk account for further details.

            Best Regards,
            TherapyTalk Team
            """
            mail.send(msg)
            print(f"DEBUG: Email sent to {patient_email} for appointment
{appointment_id}")
            flash("Appointment approved and patient notified via email.",
"success")
        else:
            flash("Appointment not found.", "danger")

    except Exception as e:
        print(f"DEBUG: Error updating appointment status: {e}")
        flash("An error occurred while approving the appointment.", "danger")
    finally:
        cur.close()

    return redirect(url_for('doctor_dashboard'))

@app.route('/reject_appointment/<int:appointment_id>', methods=['POST'])
def reject_appointment(appointment_id):
    if 'loggedin' not in session or session['role'] != 'doctor':
        return redirect(url_for('login'))

    cur = mysql.connection.cursor()

    try:
        # Update the appointment status to "Rejected"
        cur.execute("UPDATE appointments SET status = 'Rejected' WHERE id = %s",
(appointment_id,))
        mysql.connection.commit()

        # Fetch appointment details
        cur.execute("""
            SELECT p.patient_name, p.email, d.doctor_name, d.specialty
            FROM appointments a
            JOIN patients p ON a.patient_id = p.id
            JOIN doctors d ON a.doctor_id = d.id
            WHERE a.id = %s
        """, (appointment_id,))
        appointment_details = cur.fetchone()

        if appointment_details:
            patient_name, patient_email, doctor_name, specialty =
appointment_details

            # Send email to the patient
            msg = Message(
                subject="Appointment Rejected - TherapyTalk",
                sender="mrshreyash08@gmail.com",  # Replace with your email
                recipients=[patient_email]
            )
            msg.body = f"""
            Dear {patient_name},

            We regret to inform you that your appointment request with Dr.
{doctor_name} ({specialty}) has been rejected.

            Appointment Details:
            - Doctor: Dr. {doctor_name}
            - Specialty: {specialty}
            - Status: Rejected

            Please log in to your TherapyTalk account to request another
appointment.

            Best Regards,
            TherapyTalk Team
            """
            mail.send(msg)

            flash("Appointment rejected and patient notified via email.",
"success")
        else:
            flash("Appointment not found.", "danger")

    except Exception as e:
        print(f"Error: {e}")
        flash("An error occurred while processing your request.", "danger")

    finally:
        cur.close()

    return redirect(url_for('doctor_dashboard'))

@app.route('/complete_appointment/<int:appointment_id>', methods=['POST'])
def complete_appointment(appointment_id):
    if 'loggedin' not in session or session['role'] != 'doctor':
        return redirect(url_for('login'))

    cur = mysql.connection.cursor()

    try:
        # Update the appointment status to "Completed"
        cur.execute("UPDATE appointments SET status = 'Completed' WHERE id =
%s", (appointment_id,))
        mysql.connection.commit()

        flash("Appointment marked as completed!", "success")
    except Exception as e:
        print(f"Error: {e}")
        flash("An error occurred while completing the appointment.", "danger")
    finally:
        cur.close()

    return redirect(url_for('doctor_dashboard'))

@app.route('/admin/approve_appointment/<int:request_id>', methods=['POST'])
def approve_appointment(request_id):
    if 'loggedin' not in session or session['role'] != 'admin':
        return redirect(url_for('login'))

    action = request.form.get('action')  # 'approve' or 'decline'
    admin_message = request.form.get('admin_message', '')

    # 🚀 Debugging Print Statements
    print(f"DEBUG: Received request_id={request_id}, action={action}")

    try:
        cur = mysql.connection.cursor()

        # 🚀 Verify if request_id exists in DB
        cur.execute("SELECT id FROM appointment_requests WHERE id = %s",
(request_id,))
        row = cur.fetchone()
        if not row:
            print(f"ERROR: request_id {request_id} not found in DB!")
            flash("Invalid appointment request.", "danger")
            return redirect(url_for('admin_dashboard'))

        # ✅ Update appointment status based on action
        if action == 'approve':
            print(f"DEBUG: Approving appointment ID {request_id}")
            cur.execute("""
                UPDATE appointment_requests
                SET payment_status = 'Completed', appointment_status =
'Approved', admin_message = %s
                WHERE id = %s
            """, (admin_message, request_id))
        elif action == 'decline':
            print(f"DEBUG: Declining appointment ID {request_id}")
            cur.execute("""
                UPDATE appointment_requests
                SET payment_status = 'Failed', appointment_status = 'Declined',
admin_message = %s
                WHERE id = %s
            """, (admin_message, request_id))

        # 🚀 Verify if rows were updated
        mysql.connection.commit()
        rows_updated = cur.rowcount
        print(f"DEBUG: Rows updated: {rows_updated}")

        cur.close()

        if rows_updated > 0:
            flash(f"Appointment request {action}d successfully.", "success")
        else:
            flash("No changes were made. Possible issue with request ID.",
"warning")

        return redirect(url_for('admin_dashboard'))

    except Exception as e:
        print(f"ERROR: {e}")
        flash("An error occurred while updating the appointment.", "danger")
        return redirect(url_for('admin_dashboard'))


    except Exception as e:
        print(f"DEBUG: Error approving appointment: {e}")  # Debugging error
        flash("An error occurred while processing the request.", "danger")
        return redirect(url_for('admin_dashboard'))





@app.route('/doctor/accept_appointment/<int:request_id>', methods=['POST'])
def doctor_accept_appointment(request_id):
    if 'loggedin' not in session or session['role'] != 'doctor':
        return redirect(url_for('login'))

    action = request.form.get('action')  # 'accept' or 'reschedule'
    new_date = request.form.get('new_date')
    new_time = request.form.get('new_time')

    try:
        cur = mysql.connection.cursor()

        # 🚀 Debugging - Print values received
        print(f"DEBUG: Action = {action}, Request ID = {request_id}, New Date =
{new_date}, New Time = {new_time}")

        # ✅ Check if request_id exists
        cur.execute("SELECT id FROM appointment_requests WHERE id = %s",
(request_id,))
        row = cur.fetchone()
        if not row:
            print(f"ERROR: request_id {request_id} not found in DB!")
            flash("Invalid appointment request.", "danger")
            return redirect(url_for('doctor_dashboard'))

        if action == 'accept':
            print(f"DEBUG: Accepting appointment ID {request_id}")
            cur.execute("""
                UPDATE appointment_requests
                SET appointment_status = 'Accepted'
                WHERE id = %s
            """, (request_id,))

        elif action == 'reschedule':
            print(f"DEBUG: Rescheduling appointment ID {request_id} to
{new_date} {new_time}")
            cur.execute("""
                UPDATE appointment_requests
                SET appointment_date = %s, time_slot = %s, appointment_status =
'Rescheduled'
                WHERE id = %s
            """, (new_date, new_time, request_id))

        mysql.connection.commit()  # Ensure changes are saved
        rows_updated = cur.rowcount  # Check how many rows were updated
        cur.close()

        print(f"DEBUG: Rows updated = {rows_updated}")

        if rows_updated > 0:
            flash(f"Appointment {action}ed successfully.", "success")
        else:
            flash("No changes were made. Possible issue with request ID.",
"warning")

        return redirect(url_for('doctor_dashboard'))

    except Exception as e:
        print(f"ERROR: {e}")
        flash("An error occurred while updating the appointment request.",
"danger")
        return redirect(url_for('doctor_dashboard'))


@app.route('/reschedule_appointment/<int:request_id>', methods=['POST'])
def reschedule_appointment(request_id):
    if 'loggedin' not in session or session['role'] != 'doctor':
        return redirect(url_for('login'))

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

    try:
        cur = mysql.connection.cursor()
        cur.execute("""
            UPDATE appointment_requests
            SET appointment_status = 'Rescheduled', appointment_date = %s
            WHERE id = %s
        """, (new_time, request_id))
        mysql.connection.commit()
        cur.close()

        flash("Appointment rescheduled successfully.", "success")
        return redirect(url_for('doctor_dashboard'))

    except Exception as e:
        print(f"ERROR: {e}")
        flash("An error occurred while rescheduling the appointment.", "danger")
        return redirect(url_for('doctor_dashboard'))



@app.route('/book_appointment', methods=['POST'])
def book_appointment():
    if 'loggedin' not in session or session['role'] != 'patient':
        return redirect(url_for('login'))

    patient_id = session.get('user_id')
    doctor_id = request.form.get('doctor_id')
    appointment_date = request.form.get('appointmentDate')
    time_slot = request.form.get('timeSlot')
    account_holder = request.form.get('account_holder')
    transaction_id = request.form.get('transaction_id')

    try:
        cur = mysql.connection.cursor()
        cur.execute("""
            INSERT INTO appointment_requests (patient_id, doctor_id,
appointment_date, time_slot, account_holder, transaction_id, payment_status,
appointment_status)
            VALUES (%s, %s, %s, %s, %s, %s, 'Pending', 'Pending')
        """, (patient_id, doctor_id, appointment_date, time_slot,
account_holder, transaction_id))
        mysql.connection.commit()
        cur.close()

        flash("Appointment request submitted. Payment verification is pending.",
"info")
        return redirect(url_for('patient_dashboard'))

    except Exception as e:
        print(f"DEBUG: Error booking appointment: {e}")
        flash("An error occurred while booking appointment.", "danger")
        return redirect(url_for('book_appointment'))

@app.route('/doctor_dashboard')
def doctor_dashboard():
    if 'loggedin' not in session or session['role'] != 'doctor' or 'doctor_id'
not in session:
        return redirect(url_for('login'))

    doctor_id = session['doctor_id']
    print(f"DEBUG: Doctor ID = {doctor_id}")

    doctor_id = session.get('doctor_id')
    if not doctor_id:
        flash("Doctor ID not found in session. Please log in again.", "danger")
        return redirect(url_for('login'))

    try:
        cur = mysql.connection.cursor()

        # Fetch Patient Activity (Dashboard Stats)
        cur.execute("""
            SELECT
                COUNT(DISTINCT patient_id) AS total_requests,
                SUM(CASE WHEN status IN ('Ongoing', 'Completed') THEN 1 ELSE 0
END) AS approved_requests,
                SUM(CASE WHEN status = 'Ongoing' THEN 1 ELSE 0 END) AS
active_patients,
                SUM(CASE WHEN status = 'Completed' THEN 1 ELSE 0 END) AS
completed_therapy,
                SUM(CASE WHEN status = 'Pending' THEN 1 ELSE 0 END) AS
pending_requests
            FROM appointments
            WHERE doctor_id = %s
        """, (doctor_id,))
        patient_activity = cur.fetchone()
        print(f"DEBUG: Patient Activity = {patient_activity}")

        # Fetch Pending Requests (Appointments Table)
        cur.execute("""
            SELECT a.id, p.patient_name, p.disorder, p.email, p.contact_no,
p.address, a.request_date
            FROM appointments a
            JOIN patients p ON a.patient_id = p.id
            WHERE a.doctor_id = %s AND a.status = 'Pending'
        """, (doctor_id,))
        pending_requests_list = cur.fetchall()
        print(f"DEBUG: Pending Requests = {pending_requests_list}")

        # Fetch Active Patients
        cur.execute("""
            SELECT a.id, p.patient_name, p.disorder, p.email, p.contact_no
            FROM appointments a
            JOIN patients p ON a.patient_id = p.id
            WHERE a.doctor_id = %s AND a.status = 'Ongoing'
        """, (doctor_id,))
        active_patients_list = cur.fetchall()
        print(f"DEBUG: Active Patients = {active_patients_list}")

        # Fetch Today's Sessions
        cur.execute("""
            SELECT a.id, p.patient_name, p.disorder, p.email, p.contact_no,
a.time_slot, a.appointment_status
            FROM appointment_requests a
            JOIN patients p ON a.patient_id = p.id
            WHERE a.doctor_id = %s
            AND DATE(a.appointment_date) = CURDATE()
            AND a.appointment_status IN ('Accepted', 'Rescheduled', 'Session
Scheduled')
        """, (doctor_id,))
        todays_sessions = cur.fetchall()
        print(f"DEBUG: Today's Sessions = {todays_sessions}")

        # Fetch Completed Therapy Patients
        cur.execute("""
            SELECT p.patient_name, p.disorder, p.email, p.contact_no,
COUNT(a.id) AS sessions_completed
            FROM appointments a
            JOIN patients p ON a.patient_id = p.id
            WHERE a.doctor_id = %s AND a.status = 'Completed'
            GROUP BY p.id
        """, (doctor_id,))
        completed_therapy_list = cur.fetchall()
        print(f"DEBUG: Completed Therapy = {completed_therapy_list}")

        # Fetch Patients for Uploading Therapy Plan
        cur.execute("""
            SELECT a.id, p.patient_name, p.disorder, p.email, p.contact_no,
a.status
```

```
            FROM appointments a
            JOIN patients p ON a.patient_id = p.id
            WHERE a.doctor_id = %s AND a.status IN ('Approved', 'Ongoing')
        """, (doctor_id,))
        upload_plan_appointments = cur.fetchall()
        print(f"DEBUG: Upload Plan Appointments = {upload_plan_appointments}")

        # Fetch pending payment request(appointment_requests table)
        cur.execute("""
    SELECT ar.id, p.patient_name, ar.appointment_date, ar.time_slot,
ar.appointment_status
    FROM appointment_requests ar
    JOIN patients p ON ar.patient_id = p.id
    WHERE ar.appointment_status = 'Approved' AND ar.doctor_id = %s
""", (doctor_id,))
        pending_payment_requests = cur.fetchall()
        print(f"DEBUG: Pending Payment Requests = {pending_payment_requests}")

        # Fetch Payment history
        cur.execute("""
    SELECT p.patient_name, ar.transaction_id, ar.payment_status,
ar.appointment_date
    FROM appointment_requests ar
    JOIN patients p ON ar.patient_id = p.id
    WHERE ar.doctor_id = %s AND ar.payment_status IN ('Approved', 'Denied')
""", (doctor_id,))
        payment_history = cur.fetchall()

        cur.close()

        if patient_activity:
            total_requests = int(patient_activity[0]) if patient_activity[0] is
not None else 0
            approved_requests = int(patient_activity[1]) if patient_activity[1]
is not None else 0
            active_patients = int(patient_activity[2]) if patient_activity[2] is
not None else 0
            completed_therapy = int(patient_activity[3]) if patient_activity[3]
is not None else 0
            pending_requests = int(patient_activity[4]) if patient_activity[4]
is not None else 0
        else:
            total_requests = 0
            approved_requests = 0
            active_patients = 0
            completed_therapy = 0
            pending_requests = 0

        return render_template(
            'doctor_dashboard.html',
            total_requests=total_requests,
            approved_requests=approved_requests,
            active_patients=active_patients,
            completed_therapy=completed_therapy,
            pending_requests=pending_requests,
            pending_requests_list=pending_requests_list,
            active_patients_list=active_patients_list,
            todays_sessions=todays_sessions,
            completed_therapy_list=completed_therapy_list,
            upload_plan_appointments=upload_plan_appointments,
            pending_payment_requests=pending_payment_requests,
            payment_history=payment_history
        )
    except Exception as general_err:
        print(f"DEBUG: General Error in doctor_dashboard: {general_err}")
        flash("An unexpected error occurred. Please contact support.", "danger")
        return redirect(url_for('doctor_login'))

    finally:
        if 'cur' in locals() and cur:
            cur.close()

@app.route('/admin_login', methods=['GET', 'POST'])
def admin_login():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')

        print(f"Received username: {username}, password: {password}")  #
Debugging

        cur = mysql.connection.cursor()
        cur.execute("SELECT * FROM admins WHERE username = %s", (username,))
        admin = cur.fetchone()
        cur.close()

        if admin:
            print(f"Stored password in DB: {admin[2]}")  # Debugging

            if admin[2] == password:  # Direct comparison
                session['loggedin'] = True
                session['role'] = 'admin'
                session['user_id'] = admin[0]
                session['username'] = username

                print("Login successful!")  # Debugging
                flash("Login successful!", "success")
                return redirect(url_for('admin_dashboard'))
            else:
                print("Password mismatch")  # Debugging
                flash("Invalid username or password.", "danger")
                return render_template('admin_login.html')

        flash("Invalid username or password.", "danger")
        return render_template('admin_login.html')

    return render_template('admin_login.html')



@app.route('/admin_dashboard')
def admin_dashboard():
    if 'loggedin' not in session or session['role'] != 'admin':
        return redirect(url_for('admin_login'))

    cur = mysql.connection.cursor()

    # Fetch dashboard statistics
    cur.execute("SELECT COUNT(*) FROM patients")
    patient_count = cur.fetchone()[0]

    cur.execute("SELECT COUNT(*) FROM doctors")
    therapist_count = cur.fetchone()[0]

    cur.execute("SELECT COUNT(*) FROM appointments WHERE status = 'Ongoing'")
    active_cases = cur.fetchone()[0]
    cur.execute("SELECT COUNT(*) FROM appointments WHERE status = 'Completed'")
    completed_cases = cur.fetchone()[0]

    cur.execute("SELECT SUM(amount) FROM payments")
    total_revenue = cur.fetchone()[0] or 0

    # Fetch therapist data
    cur.execute("SELECT doctor_name, email, specialty, patients_allocated, 
patients_completed FROM doctors")
    therapists = cur.fetchall()

    # Fetch patient data
    cur.execute("SELECT patient_name, email, disorder, therapist_allocated, 
status FROM patients")
    patients = cur.fetchall()

    # Fetch payment history
    cur.execute("SELECT patient_name, transaction_id, amount, payment_date FROM 
payments")
    payments = cur.fetchall()

    # ✅ Fetch pending approval appointments
    cur.execute("""
    SELECT p.patient_name, p.email, p.disorder, ar.amount, ar.account_holder,
           ar.transaction_id, ar.appointment_date, ar.time_slot, 
ar.request_time, ar.id
    FROM appointment_requests ar
    JOIN patients p ON ar.patient_id = p.id
    WHERE ar.appointment_status = 'Pending'
""")

    pending_appointments = cur.fetchall()

    cur.close()

    return render_template(
        'admin_dashboard.html',
        patient_count=patient_count,
        therapist_count=therapist_count,
        active_cases=active_cases,
        completed_cases=completed_cases,
        total_revenue=total_revenue,
        therapists=therapists,
        patients=patients,
        payments=payments,
        pending_appointments=pending_appointments  # ✅ Add pending approvals
    )


if __name__ == '__main__':
    app.run(debug=True)
```