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

import mysql.connector

import bcrypt

from functools import wraps

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key\_here'

app.config['SESSION\_TYPE'] = 'filesystem'

# Database configuration (same as before)

db\_config = {

'host': 'localhost',

'user': 'root',

'password': '1234',

'database': 'user\_transaction\_db'

}

# Login required decorator

def login\_required(f):

@wraps(f)

def decorated\_function(\*args, \*\*kwargs):

if 'admin\_logged\_in' not in session:

return redirect(url\_for('login'))

return f(\*args, \*\*kwargs)

return decorated\_function

# Routes

@app.route('/')

def home():

return redirect(url\_for('login'))

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

def login():

if request.method == 'POST':

try:

username = request.form['username']

password = request.form['password'].encode('utf-8')

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM admins WHERE username = %s", (username,))

admin = cursor.fetchone()

cursor.close()

conn.close()

if admin and bcrypt.checkpw(password, admin['password'].encode('utf-8')):

session['admin\_logged\_in'] = True

session['admin\_username'] = username

return redirect(url\_for('dashboard'))

else:

return render\_template('login.html', error='Invalid credentials')

except mysql.connector.Error as err:

error\_message = f"Database error: {err}"

return render\_template('login.html', error=error\_message)

except Exception as e:

return render\_template('login.html', error=str(e))

return render\_template('login.html')

@app.route('/dashboard')

@login\_required

def dashboard():

try:

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor(dictionary=True)

# Get stats

cursor.execute("SELECT COUNT(\*) as total\_users FROM users")

total\_users = cursor.fetchone()['total\_users']

cursor.execute("""

SELECT

COUNT(\*) as total\_orders,

SUM(status = 'success') as success\_orders,

SUM(status = 'failed') as failed\_orders

FROM transaction\_table

""")

stats = cursor.fetchone()

cursor.close()

conn.close()

return render\_template('dashboard.html',

total\_users=total\_users,

total\_orders=stats['total\_orders'],

success\_orders=stats['success\_orders'],

failed\_orders=stats['failed\_orders'])

except Exception as e:

return str(e)

# Users route

@app.route('/users')

@login\_required

def users():

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM users ORDER BY date DESC")

users = cursor.fetchall()

cursor.close()

conn.close()

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

# Transactions route

@app.route('/transactions')

@login\_required

def transactions():

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM transaction\_table ORDER BY date DESC")

transactions = cursor.fetchall()

cursor.close()

conn.close()

return render\_template('transactions.html', transactions=transactions)

# User Routes

@app.route('/get\_user/<int:sno>')

@login\_required

def get\_user(sno):

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM users WHERE sno = %s", (sno,))

user = cursor.fetchone()

cursor.close()

conn.close()

return jsonify(user)

# Update User Route

@app.route('/update\_user', methods=['PUT'])

@login\_required

def update\_user():

try:

data = request.get\_json()

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor()

cursor.execute("""

UPDATE users SET

username = %s,

email = %s,

mobile\_no = %s

WHERE sno = %s

""", (data['username'], data['email'], data['mobile\_no'], data['sno']))

conn.commit()

return '', 204

except Exception as e:

return jsonify({'error': str(e)}), 400

finally:

cursor.close()

conn.close()

@app.route('/delete\_user/<int:sno>', methods=['DELETE'])

@login\_required

def delete\_user(sno):

try:

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor()

cursor.execute("DELETE FROM users WHERE sno = %s", (sno,))

conn.commit()

return '', 204

except Exception as e:

return str(e), 400

finally:

cursor.close()

conn.close()

# Transaction Routes

@app.route('/get\_transaction/<int:sno>')

@login\_required

def get\_transaction(sno):

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor(dictionary=True)

cursor.execute("SELECT \* FROM transaction\_table WHERE sno = %s", (sno,))

transaction = cursor.fetchone()

cursor.close()

conn.close()

return jsonify(transaction)

# Update Transaction Route

@app.route('/update\_transaction', methods=['PUT'])

@login\_required

def update\_transaction():

try:

data = request.get\_json()

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor()

cursor.execute("""

UPDATE transaction\_table SET

username = %s,

transaction\_id = %s,

email = %s,

amount = %s,

commission = %s,

status = %s

WHERE sno = %s

""", (

data['username'],

data['transaction\_id'],

data['email'],

data['amount'],

data['commission'],

data['status'],

data['sno']

))

conn.commit()

return '', 204

except Exception as e:

return jsonify({'error': str(e)}), 400

finally:

cursor.close()

conn.close()

@app.route('/delete\_transaction/<int:sno>', methods=['DELETE'])

@login\_required

def delete\_transaction(sno):

try:

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor()

cursor.execute("DELETE FROM transaction\_table WHERE sno = %s", (sno,))

conn.commit()

return '', 204

except Exception as e:

return str(e), 400

finally:

cursor.close()

conn.close()

@app.route('/logout')

def logout():

session.clear()

return redirect(url\_for('login'))

# Keep your existing API endpoints here

# ...

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

#app.run(debug=True)

app.run(host="0.0.0.0", port=5000)