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

import sqlite3 as sql

from functools import wraps

import re

import ibm\_db

import os

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail

from datetime import datetime, timedelta

conn = ibm\_db.connect("DATABASE=bludb;HOSTNAME=815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30367;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gkx49901;PWD=kvWCsySl7vApfsy2", '', '')

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

app.secret\_key = 'jackiechan'

def rewrite(url):

view\_func, view\_args = app.create\_url\_adapter(request).match(url)

return app.view\_functions[view\_func](\*\*view\_args)

def login\_required(f):

@wraps(f)

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

if "id" not in session:

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

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

return decorated\_function

@app.route('/')

def root():

return render\_template('login.html')

@app.route('/user/<id>')

@login\_required

def user\_info(id):

with sql.connect('inventorymanagement.db') as con:

con.row\_factory = sql.Row

cur = con.cursor()

cur.execute(f'SELECT \* FROM users WHERE email="{id}"')

user = cur.fetchall()

return render\_template("user\_info.html", user=user[0])

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

def login():

global userid

msg = ''

if request.method == 'POST':

un = request.form['username']

pd = request.form['password\_1']

print(un, pd)

sql = "SELECT \* FROM users WHERE email =? AND password=?"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, un)

ibm\_db.bind\_param(stmt, 2, pd)

ibm\_db.execute(stmt)

account = ibm\_db.fetch\_assoc(stmt)

print(account)

if account:

session['loggedin'] = True

session['id'] = account['EMAIL']

userid = account['EMAIL']

session['username'] = account['USERNAME']

msg = 'Logged in successfully !'

return rewrite('/dashboard')

else:

msg = 'Incorrect username / password !'

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

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

def signup():

mg = ''

if request.method == "POST":

username = request.form['username']

email = request.form['email']

pw = request.form['password']

sql = 'SELECT \* FROM users WHERE email =?'

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt, 1, email)

ibm\_db.execute(stmt)

acnt = ibm\_db.fetch\_assoc(stmt)

print(acnt)

if acnt:

mg = 'Account already exits!!'

elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

mg = 'Please enter the avalid email address'

elif not re.match(r'[A-Za-z0-9]+', username):

ms = 'name must contain only character and number'

else:

insert\_sql = 'INSERT INTO users (USERNAME,FIRSTNAME,LASTNAME,EMAIL,PASSWORD) VALUES (?,?,?,?,?)'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, username)

ibm\_db.bind\_param(pstmt, 2, "firstname")

ibm\_db.bind\_param(pstmt, 3, "lastname")

# ibm\_db.bind\_param(pstmt,4,"123456789")

ibm\_db.bind\_param(pstmt, 4, email)

ibm\_db.bind\_param(pstmt, 5, pw)

print(pstmt)

ibm\_db.execute(pstmt)

mg = 'You have successfully registered click login!'

message = Mail(

from\_email=os.environ.get('MAIL\_DEFAULT\_SENDER'),

to\_emails=email,

subject='New SignUp',

html\_content='<p>Hello, Your Registration was successfull. <br><br> Thank you for choosing us.</p>')

sg = SendGridAPIClient(

api\_key=os.environ.get('SENDGRID\_API\_KEY'))

response = sg.send(message)

print(response.status\_code, response.body)

return render\_template("login.html", meg=mg)

elif request.method == 'POST':

msg = "fill out the form first!"

return render\_template("signup.html", meg=mg)

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

@login\_required

def dashBoard():

sql = "SELECT \* FROM stocks"

stmt = ibm\_db.exec\_immediate(conn, sql)

dictionary = ibm\_db.fetch\_assoc(stmt)

stocks = []

headings = [\*dictionary]

while dictionary != False:

stocks.append(dictionary)

# print(f"The ID is : ", dictionary["NAME"])

# print(f"The name is : ", dictionary["QUANTITY"])

dictionary = ibm\_db.fetch\_assoc(stmt)

return render\_template("dashboard.html", headings=headings, data=stocks)

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

@login\_required

def addStocks():

if request.method == "POST":

print(request.form['item'])

try:

item = request.form['item']

quantity = request.form['quantity']

price = request.form['price']

total = int(price) \* int(quantity)

insert\_sql = 'INSERT INTO stocks (NAME,QUANTITY,PRICE\_PER\_QUANTITY,TOTAL\_PRICE) VALUES (?,?,?,?)'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, item)

ibm\_db.bind\_param(pstmt, 2, quantity)

ibm\_db.bind\_param(pstmt, 3, price)

ibm\_db.bind\_param(pstmt, 4, total)

ibm\_db.execute(pstmt)

except Exception as e:

msg = e

finally:

# print(msg)

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

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

@login\_required

def UpdateStocks():

if request.method == "POST":

try:

item = request.form['item']

print("hello")

field = request.form['input-field']

value = request.form['input-value']

print(item, field, value)

insert\_sql = 'UPDATE stocks SET ' + field + "= ?" + " WHERE NAME=?"

print(insert\_sql)

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, value)

ibm\_db.bind\_param(pstmt, 2, item)

ibm\_db.execute(pstmt)

if field == 'PRICE\_PER\_QUANTITY' or field == 'QUANTITY':

insert\_sql = 'SELECT \* FROM stocks WHERE NAME= ?'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, item)

ibm\_db.execute(pstmt)

dictonary = ibm\_db.fetch\_assoc(pstmt)

print(dictonary)

total = dictonary['QUANTITY'] \* dictonary['PRICE\_PER\_QUANTITY']

insert\_sql = 'UPDATE stocks SET TOTAL\_PRICE=? WHERE NAME=?'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, total)

ibm\_db.bind\_param(pstmt, 2, item)

ibm\_db.execute(pstmt)

except Exception as e:

msg = e

finally:

# print(msg)

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

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

@login\_required

def deleteStocks():

if request.method == "POST":

print(request.form['item'])

try:

item = request.form['item']

insert\_sql = 'DELETE FROM stocks WHERE NAME=?'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, item)

ibm\_db.execute(pstmt)

except Exception as e:

msg = e

finally:

# print(msg)

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

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

@login\_required

def updateUser():

if request.method == "POST":

try:

email = session['id']

field = request.form['input-field']

value = request.form['input-value']

insert\_sql = 'UPDATE users SET ' + field + '= ? WHERE EMAIL=?'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, value)

ibm\_db.bind\_param(pstmt, 2, email)

ibm\_db.execute(pstmt)

except Exception as e:

msg = e

finally:

# print(msg)

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

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

@login\_required

def updatePassword():

if request.method == "POST":

try:

email = session['id']

password = request.form['prev-password']

curPassword = request.form['cur-password']

confirmPassword = request.form['confirm-password']

insert\_sql = 'SELECT \* FROM users WHERE EMAIL=? AND PASSWORD=?'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, email)

ibm\_db.bind\_param(pstmt, 2, password)

ibm\_db.execute(pstmt)

dictionary = ibm\_db.fetch\_assoc(pstmt)

print(dictionary)

if curPassword == confirmPassword:

insert\_sql = 'UPDATE users SET PASSWORD=? WHERE EMAIL=?'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, confirmPassword)

ibm\_db.bind\_param(pstmt, 2, email)

ibm\_db.execute(pstmt)

except Exception as e:

msg = e

finally:

# print(msg)

return render\_template('result.html')

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

@login\_required

def orders():

query = "SELECT \* FROM orders"

stmt = ibm\_db.exec\_immediate(conn, query)

dictionary = ibm\_db.fetch\_assoc(stmt)

orders = []

headings = [\*dictionary]

while dictionary != False:

orders.append(dictionary)

dictionary = ibm\_db.fetch\_assoc(stmt)

return render\_template("orders.html", headings=headings, data=orders)

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

@login\_required

def createOrder():

if request.method == "POST":

try:

stock\_id = request.form['stock\_id']

query = 'SELECT PRICE\_PER\_QUANTITY FROM stocks WHERE ID= ?'

stmt = ibm\_db.prepare(conn, query)

ibm\_db.bind\_param(stmt, 1, stock\_id)

ibm\_db.execute(stmt)

dictionary = ibm\_db.fetch\_assoc(stmt)

if dictionary:

quantity = request.form['quantity']

date = str(datetime.now().year) + "-" + str(

datetime.now().month) + "-" + str(datetime.now().day)

delivery = datetime.now() + timedelta(days=7)

delivery\_date = str(delivery.year) + "-" + str(

delivery.month) + "-" + str(delivery.day)

price = float(quantity) \* \

float(dictionary['PRICE\_PER\_QUANTITY'])

query = 'INSERT INTO orders (STOCKS\_ID,QUANTITY,DATE,DELIVERY\_DATE,PRICE) VALUES (?,?,?,?,?)'

pstmt = ibm\_db.prepare(conn, query)

ibm\_db.bind\_param(pstmt, 1, stock\_id)

ibm\_db.bind\_param(pstmt, 2, quantity)

ibm\_db.bind\_param(pstmt, 3, date)

ibm\_db.bind\_param(pstmt, 4, delivery\_date)

ibm\_db.bind\_param(pstmt, 5, price)

ibm\_db.execute(pstmt)

except Exception as e:

print(e)

finally:

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

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

@login\_required

def updateOrder():

if request.method == "POST":

try:

item = request.form['item']

field = request.form['input-field']

value = request.form['input-value']

query = 'UPDATE orders SET ' + field + "= ?" + " WHERE ID=?"

pstmt = ibm\_db.prepare(conn, query)

ibm\_db.bind\_param(pstmt, 1, value)

ibm\_db.bind\_param(pstmt, 2, item)

ibm\_db.execute(pstmt)

except Exception as e:

print(e)

finally:

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

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

@login\_required

def cancelOrder():

if request.method == "POST":

try:

order\_id = request.form['order\_id']

query = 'DELETE FROM orders WHERE ID=?'

pstmt = ibm\_db.prepare(conn, query)

ibm\_db.bind\_param(pstmt, 1, order\_id)

ibm\_db.execute(pstmt)

except Exception as e:

print(e)

finally:

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

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

@login\_required

def suppliers():

sql = "SELECT \* FROM suppliers"

stmt = ibm\_db.exec\_immediate(conn, sql)

dictionary = ibm\_db.fetch\_assoc(stmt)

suppliers = []

orders\_assigned = []

headings = [\*dictionary]

while dictionary != False:

suppliers.append(dictionary)

orders\_assigned.append(dictionary['ORDER\_ID'])

dictionary = ibm\_db.fetch\_assoc(stmt)

# get order ids from orders table and identify unassigned order ids

sql = "SELECT ID FROM orders"

stmt = ibm\_db.exec\_immediate(conn, sql)

dictionary = ibm\_db.fetch\_assoc(stmt)

order\_ids = []

while dictionary != False:

order\_ids.append(dictionary['ID'])

dictionary = ibm\_db.fetch\_assoc(stmt)

unassigned\_order\_ids = set(order\_ids) - set(orders\_assigned)

return render\_template("suppliers.html",headings=headings,data=suppliers,order\_ids=unassigned\_order\_ids)

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

@login\_required

def UpdateSupplier():

if request.method == "POST":

try:

item = request.form['name']

field = request.form['input-field']

value = request.form['input-value']

print(item, field, value)

insert\_sql = 'UPDATE suppliers SET ' + field + "= ?" + " WHERE NAME=?"

print(insert\_sql)

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, value)

ibm\_db.bind\_param(pstmt, 2, item)

ibm\_db.execute(pstmt)

except Exception as e:

msg = e

finally:

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

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

@login\_required

def addSupplier():

if request.method == "POST":

try:

name = request.form['name']

order\_id = request.form.get('order-id-select')

print(order\_id)

print("Hello world")

location = request.form['location']

insert\_sql = 'INSERT INTO suppliers (NAME,ORDER\_ID,LOCATION) VALUES (?,?,?)'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, name)

ibm\_db.bind\_param(pstmt, 2, order\_id)

ibm\_db.bind\_param(pstmt, 3, location)

ibm\_db.execute(pstmt)

except Exception as e:

msg = e

finally:

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

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

@login\_required

def deleteSupplier():

if request.method == "POST":

try:

item = request.form['name']

insert\_sql = 'DELETE FROM suppliers WHERE NAME=?'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, item)

ibm\_db.execute(pstmt)

except Exception as e:

msg = e

finally:

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

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

@login\_required

def profile():

if request.method == "GET":

try:

email = session['id']

insert\_sql = 'SELECT \* FROM users WHERE EMAIL=?'

pstmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(pstmt, 1, email)

ibm\_db.execute(pstmt)

dictionary = ibm\_db.fetch\_assoc(pstmt)

print(dictionary)

except Exception as e:

msg = e

finally:

# print(msg)

return render\_template("profile.html", data=dictionary)

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

@login\_required

def logout():

print(request)

resp = make\_response(render\_template("login.html"))

session.clear()

return resp

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

app.run(debug=True)

# ALTER TABLE stocks ALTER COLUMN ID SET GENERATED BY DEFAULT AS IDENTITY