```
from flask import Flask,render_template,request,redirect,session,make_response,url_for
import sqlite3 as sql
from functools import wraps
import datetime
import re
import os
from datetime import timedelta
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

import ibm_db
app=Flask(__name__)
app.secret_key = 'jackiechan'
SENDGRID_API_KEY="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"
MAIL_DEFAULT_SENDER="jagadeep.j.2019.cse@ritchennai.edu.in"

hostname = "815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
uid = "tkx67682"
pwd = "o7CLe1RKTEB89jC8"
driver = "{IBM DB2 ODBC DRIVER}"
db = "bludb"
port = "30367"
protocol = "TCPIP"
cert = "DigiCertGlobalRootCA.crt"

dsn = (
   "DATABASE={0};"
   "HOSTNAME={1};"
   "PORT={2};"
   "UID={3};"
```

```
    "SECURITY=SSL;"
    "SSLServerCertificate={4};"
    "PWD={5};"
).format(db, hostname, port, uid, cert, pwd)

print(dsn)

conn = ibm_db.connect(dsn, "", "")
email = 'harish19gmail.com'

sql = "SELECT * FROM USERS WHERE email = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, email)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)

print(account)
@app.route('/')
def root():
    return render_template("login.html")

@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):
           mg = 'name must contain only character and number'
        else:
           insert_sql = 'INSERT INTO USERS (USERNAME,EMAIL,PASSWORD) VALUES (?,?,?)'
           pstmt = ibm_db.prepare(conn, insert_sql)
           ibm_db.bind_param(pstmt, 1, username)
           # ibm_db.bind_param(pstmt,4,"123456789")
           ibm_db.bind_param(pstmt, 2, email)
           ibm_db.bind_param(pstmt, 3, pw)
           print(pstmt)
           ibm_db.execute(pstmt)
           mg = 'You have successfully registered click login!'
           message = Mail( from_email="jagadeep.j.2019.cse@ritchennai.edu.in",
                       to_emails=email,
                       subject='New SignUp',
                       html_content='<strong> signup sucessfull!!</strong>')


           try:
              sg = SendGridAPIClient(SENDGRID_API_KEY)
              response = sg.send(message)
              print(response.status_code)
              print(response.body)
              return render_template("login.html", meg=mg)
```

```
            except Exception as e:
                print(e )

        return render_template("signup.html", meg=mg)
    else:
        return render_template("signup.html", meg=mg)



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('/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 = []

    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", data=stocks)



@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 = []
    while dictionary != False:
        orders.append(dictionary)
        dictionary = ibm_db.fetch_assoc(stmt)
    return render_template("orders.html", data=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 = []
    while dictionary != False:
        suppliers.append(dictionary)
        orders_assigned.append(dictionary['NAME'])
        dictionary = ibm_db.fetch_assoc(stmt)

# get order ids from orders table and identify unassigned order ids
    sql = "SELECT STOCKS_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['STOCKS_ID'])
        dictionary = ibm_db.fetch_assoc(stmt)

    unassigned_order_ids = set(order_ids) - set(orders_assigned)
    return render_template("suppliers.html",data=suppliers,order_ids=unassigned_order_ids)


@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




@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)
    else:
        return render_template('login.html')



@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)
            id =request.form['item_id']
            insert_sql = 'INSERT INTO STOCKS
(NAME,QUANTITY,PRICE_PER_QUANTITY,TOTAL_PRICE,STOCK_ID) 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.bind_param(pstmt,5,id)
            ibm_db.execute(pstmt)

        except Exception as e:
            msg = e
            print(msg)

        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)

                print('helloi')

                total = int(dictonary['QUANTITY']) * int(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

            print(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('/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('/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('/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']
            print("Hello world")
            location = request.form['location']
            insert_sql = 'INSERT INTO suppliers (NAME,LOCATION) VALUES (?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, name)
            ibm_db.bind_param(pstmt, 2, location)
            ibm_db.execute(pstmt)

        except Exception as e:
            msg = e
            print(msg)

        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('/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')




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