FINAL DELIVERABLES
Sprint 4

Team ID	PNT2022TMID30875
Project Name	SMART FASHION RECOMMENDER APPLICATION

SOURCE CODE

```
import secrets
from turtle import title
from unicodedata import category
from flask import Flask, render_template, request, redirect, url_for, session
import ibm_db
import bcrypt
import base64
import os

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=2f3279a5-73d1-4859-88f
0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=30756;
SECURITY=SSL;
SSLServerCertificateDigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=nhl8074
8;PWD=3yD0G9e6VuQHsOBX;", "", "")
#url_for('static', filename='style.css')



app = Flask(__name__)
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

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

def home():
    if 'email' not in session:
        return redirect(url_for('index'))
    shirt_list=[]
    pant_list=[]
    watch_list=[]
    ring_list=[]

    #selecting_shirt
    sql = "SELECT * FROM SHIRT"
    stmt = ibm_db.exec_immediate(conn, sql)
    shirt = ibm_db.fetch_both(stmt)
```

```
        while shirt != False :
                shirt_list.append(shirt)
                shirt = ibm_db.fetch_both(stmt)
        print(shirt_list)

    #selecting_pant

        sql1="SELECT * FROM PANT"
        stmt1 = ibm_db.exec_immediate(conn, sql1)
        pant=ibm_db.fetch_both(stmt1)
        while pant != False :
                pant_list.append(pant)
                pant = ibm_db.fetch_both(stmt1)
        print(pant_list)

    #selecting_watch
        sql2="SELECT * FROM WATCH"
        stmt2 = ibm_db.exec_immediate(conn, sql2)
        watch=ibm_db.fetch_both(stmt2)
        while watch != False :
                watch_list.append(watch)
                watch = ibm_db.fetch_both(stmt2)
        print(watch_list)

        #selecting_rings
        sql3="SELECT * FROM RINGS"
        stmt3 = ibm_db.exec_immediate(conn, sql3)
        ring=ibm_db.fetch_both(stmt3)
        while ring != False :
                ring_list.append(ring)
                ring = ibm_db.fetch_both(stmt3)
        print(ring_list)
        #returning to HTML
        return render_template('home.html',dictionary=
shirt_list,pants=pant_list,watchs=watch_list,rings=ring_list)

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

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

        if not username or not email or not phoneno or not password:
            return render_template('register.html',error='Please fill all fields')
```

```
        hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())
        query = "SELECT * FROM user_detail WHERE email=? OR phoneno=?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,phoneno)
        ibm_db.execute(stmt)
        isUser = ibm_db.fetch_assoc(stmt)
        if not isUser:
            insert_sql = "INSERT INTO user_detail(username, email, phoneno, password)
VALUES (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, phoneno)
            ibm_db.bind_param(prep_stmt, 4, hash)
            ibm_db.execute(prep_stmt)
            return render_template('register.html',success="You can login")
        else:
            return render_template('register.html',error='Invalid Credentials')

    return render_template('register.html',name='Home')

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

            if not email or not password:
                return render_template('login.html',error='Please fill all fields')
            query = "SELECT * FROM user_detail WHERE email=?"
            stmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(stmt,1,email)
            ibm_db.execute(stmt)
            isUser = ibm_db.fetch_assoc(stmt)
            print(isUser,password)

            if not isUser:
                return render_template('login.html',error='Invalid Credentials')

            isPasswordMatch =
bcrypt.checkpw(password.encode('utf-8'),isUser['PASSWORD'].encode('utf-8'))

            if not isPasswordMatch:
                return render_template('login.html',error='Invalid Credentials')

            session['email'] = isUser['EMAIL']
            return redirect(url_for('home'),emails=session['email'] )

        return render_template('login.html',name='Home')
```

```
@app.route("/admin",methods=['GET','POST'])
def adregister():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        phoneno = request.form['phoneno']
        password = request.form['password']

        if not username or not email or not phoneno or not password:
            return render_template('adminregister.html',error='Please fill all fields')
        hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())
        query = "SELECT * FROM admin_detail WHERE email=? OR phoneno=?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,phoneno)
        ibm_db.execute(stmt)
        isUser = ibm_db.fetch_assoc(stmt)
        if not isUser:
            insert_sql = "INSERT INTO admin_detail(username, email, phoneno,
password) VALUES (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, phoneno)
            ibm_db.bind_param(prep_stmt, 4, hash)
            ibm_db.execute(prep_stmt)
            return render_template('adminregister.html',success="You can login")
        else:
            return render_template('adminregister.html',error='Invalid Credentials')

    return render_template('adminregister.html',name='Home')

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

            if not email or not password:
                return render_template('adminlogin.html',error='Please fill all fields')
            query = "SELECT * FROM admin_detail WHERE email=?"
            stmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(stmt,1,email)
            ibm_db.execute(stmt)
            isUser = ibm_db.fetch_assoc(stmt)
            print(isUser,password)

            if not isUser:
                return render_template('adminlogin.html',error='Invalid Credentials')
```

```
            isPasswordMatch =
bcrypt.checkpw(password.encode('utf-8'),isUser['PASSWORD'].encode('utf-8'))

            if not isPasswordMatch:
                return render_template('adminlogin.html',error='Invalid Credentials')

            session['email'] = isUser['EMAIL']
            return redirect(url_for('addproduct'))

        return render_template('adminlogin.html',name='Home')

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

def addproduct():
    if request.method == 'POST':
        types=request.form['cc']
        name = request.form['name']
        image = request.form['image']
        rate = request.form['rate']
        categorie = request.form['categorie']
        if types =='shirt':
            insert_sql = "INSERT INTO SHIRT(name, image, categorie,rate) VALUES
(?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)
        if types =='pant':
            insert_sql = "INSERT INTO SHIRT(name, image, categorie,rate) VALUES
(?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)
        if types =='watch':
            insert_sql = "INSERT INTO WATCH(name, image, rate) VALUES (?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, rate)
            ibm_db.execute(prep_stmt)
        if types =='ring':
            insert_sql = "INSERT INTO RINGS(name, image, categorie,rate) VALUES
(?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
```

```
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, image)
            ibm_db.bind_param(prep_stmt, 3, categorie)
            ibm_db.bind_param(prep_stmt, 4, rate)
            ibm_db.execute(prep_stmt)

    return render_template('addproduct.html',success="You can login")

@app.route("/data")
def display():
    shirt_list=[]
    pant_list=[]
    watch_list=[]
    ring_list=[]

    #selecting_shirt
    sql = "SELECT * FROM SHIRT"
    stmt = ibm_db.exec_immediate(conn, sql)
    shirt = ibm_db.fetch_both(stmt)
    while shirt != False :
            shirt_list.append(shirt)
            shirt = ibm_db.fetch_both(stmt)
    print(shirt_list)

  #selecting_pant

    sql1="SELECT * FROM PANT"
    stmt1 = ibm_db.exec_immediate(conn, sql1)
    pant=ibm_db.fetch_both(stmt1)
    while pant != False :
            pant_list.append(pant)
            pant = ibm_db.fetch_both(stmt1)
    print(pant_list)

#selecting_watch
    sql2="SELECT * FROM WATCH"
    stmt2 = ibm_db.exec_immediate(conn, sql2)
    watch=ibm_db.fetch_both(stmt2)
    while watch != False :
            watch_list.append(watch)
            watch = ibm_db.fetch_both(stmt2)
    print(watch_list)

    #selecting_rings
    sql3="SELECT * FROM RINGS"
    stmt3 = ibm_db.exec_immediate(conn, sql3)
    ring=ibm_db.fetch_both(stmt3)
    while ring != False :
            ring_list.append(ring)
            ring = ibm_db.fetch_both(stmt3)
```

```
    print(ring_list)
    #returning to HTML
    return render_template('home.html',dictionary=
shirt_list,pants=pant_list,watchs=watch_list,rings=ring_list)

@app.route("/orderplaced",methods=['GET','POST'])
def dis():
    if request.method == 'POST':
        pname=request.form['name']
        img=request.form['image']
        rate=request.form['rate']
        categorie=request.form['categorie']
    return
render_template('order.html',pname=pname,img=img,rate=rate,categorie=categorie)

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

def orderdisplay():
    if request.method == 'POST':
        name = request.form['order_name']
        image = request.form['order_image']
        rate = request.form['order_rate']
        categorie = request.form['order_categorie']
        insert_sql = "INSERT INTO ORDERS(oname, oimage,orate, ocategorie)
VALUES (?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, name)
        ibm_db.bind_param(prep_stmt, 2, image)
        ibm_db.bind_param(prep_stmt, 3, rate)
        ibm_db.bind_param(prep_stmt, 4, categorie)
        ibm_db.execute(prep_stmt)
    return render_template('success.html',success="You can login")

@app.route("/displayorder")
def displayorder():
    details_list=[]
    #selecting_shirt
    sql = "SELECT * FROM ORDERS"
    stmt = ibm_db.exec_immediate(conn, sql)
    detail = ibm_db.fetch_both(stmt)
    while detail != False :
            details_list.append(detail)
            detail = ibm_db.fetch_both(stmt)
    print(details_list)
    return render_template('displayorder.html',details=details_list)

@app.route('/logout')
def logout():
        session.pop('email', None)
        return redirect(url_for('login'))
```

```
if __name__ == '__main__':
        port=int(os.environ.get('PORT',5000))
        app.run(port=port,host='0.0.0.0')
```