DATABASE CONNECTION PYTHON FLASK
SPRINT-2

TEAM ID	PNT2022TMID01360
PROJECT NAME	SMART FASHION RECOMMENDER APPLICATION
TEAM MEMBERS	PAVITHRA J, CHELSEE NATALIA A, KRISHNA SREE S B, JEYASHRUTHI M

SPRINT -2:

In sprint-1 we have connected sqllite3 as our database with our python Flask application to make all the hTML pages interactive.

IMPORTING SQL LITE DATABASE:

```
from flask import *
import sqlite3, hashlib, os
from werkzeug.utils import secure_filename


app = Flask(__name__)
app.secret_key = 'random string'
UPLOAD_FOLDER = 'static/uploads'
ALLOWED_EXTENSIONS = set(['jpeg', 'jpg', 'png', 'gif', 'webp', 'avif'])
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

GET USER LOGIN DETAILS:

```
def getLoginDetails():
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        if 'email' not in session:
            loggedIn = False
            firstName = ''
            noOfItems = 0
        else:
            loggedIn = True
            cur.execute("SELECT userId, firstName FROM users WHERE email = ?",
(session['email'], ))
            userId, firstName = cur.fetchone()
            cur.execute("SELECT count(productId) FROM kart WHERE userId = ?", (userId, ))
            noOfItems = cur.fetchone()[0]
    conn.close()
    return (loggedIn, firstName, noOfItems)
```

```
@app.route("/")
def root():
    loggedIn, firstName, noOfItems = getLoginDetails()
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute('SELECT productId, name, price, description, image, stock FROM
products')
        itemData = cur.fetchall()
        cur.execute('SELECT categoryId, name FROM categories')
        categoryData = cur.fetchall()
    itemData = parse(itemData)
    return render_template('home.html', itemData=itemData, loggedIn=loggedIn,
firstName=firstName, noOfItems=noOfItems, categoryData=categoryData)
```

LOGINFORM:

```
@app.route("/loginForm")
def loginForm():
    if 'email' in session:
        return redirect(url_for('root'))
    else:
        return render_template('login.html', error='')

@app.route("/login", methods = ['POST', 'GET'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        if is_valid(email, password):
            session['email'] = email
            return redirect(url_for('root'))
        else:
            error = 'Invalid UserId / Password'
            return render_template('login.html', error=error)
```

REGISTER:

```
@app.route("/register", methods = ['GET', 'POST'])
def register():
    if request.method == 'POST':
        #Parse form data
        password = request.form['password']
        email = request.form['email']
        firstName = request.form['firstName']
        lastName = request.form['lastName']
        address1 = request.form['address1']
        address2 = request.form['address2']
        zipcode = request.form['zipcode']
        city = request.form['city']
        state = request.form['state']
        country = request.form['country']
        phone = request.form['phone']
```

```
        with sqlite3.connect('database.db') as con:
            try:
                cur = con.cursor()
                cur.execute('INSERT INTO users (password, email, firstName, lastName,
address1, address2, zipcode, city, state, country, phone) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?)', (hashlib.md5(password.encode()).hexdigest(), email, firstName, lastName, address1,
address2, zipcode, city, state, country, phone))

                con.commit()

                msg = "Registered Successfully"
            except:
                con.rollback()
                msg = "Error occured"
        con.close()
        return render_template("login.html", error=msg)

@app.route("/registerationForm")
def registrationForm():
    return render_template("register.html")

def allowed_file(filename):
    return '.' in filename and \
            filename.rsplit('.', 1)[1] in ALLOWED_EXTENSIONS
```

ADMIN:

```
@app.route("/add")
def admin():
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute("SELECT categoryId, name FROM categories")
        categories = cur.fetchall()
    conn.close()
    return render_template('add.html', categories=categories)
```

ADD PRODUCTS TO DISPLAY PAGE(ADMIN):

```
@app.route("/addItem", methods=["GET", "POST"])
def addItem():
    if request.method == "POST":
        name = request.form['name']
        price = float(request.form['price'])
        description = request.form['description']
        stock = int(request.form['stock'])
        categoryId = int(request.form['category'])

        image = request.files['image']
        if image and allowed_file(image.filename):
            filename = secure_filename(image.filename)
            image.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        imagename = filename
        with sqlite3.connect('database.db') as conn:
```

```
            try:
                cur = conn.cursor()
                cur.execute('''INSERT INTO products (name, price, description, image, stock,
categoryId) VALUES (?, ?, ?, ?, ?, ?)''', (name, price, description, imagename, stock,
categoryId))
                conn.commit()
                msg="added successfully"
            except:
                msg="error occured"
                conn.rollback()
        conn.close()
        print(msg)
        return redirect(url_for('root'))
```

REMOVE ITEM(ADMIN):

```
@app.route("/remove")
def remove():
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute('SELECT productId, name, price, description, image, stock FROM
products')
        data = cur.fetchall()
    conn.close()
    return render_template('remove.html', data=data)

@app.route("/removeItem")
def removeItem():
    productId = request.args.get('productId')
    with sqlite3.connect('database.db') as conn:
        try:
            cur = conn.cursor()
            cur.execute('DELETE FROM products WHERE productID = ?', (productId, ))
            conn.commit()
            msg = "Deleted successsfully"
        except:
            conn.rollback()
            msg = "Error occured"
    conn.close()
    print(msg)
    return redirect(url_for('root'))
```

DISPLAY PRODUCTS PAGE:

```
@app.route("/displayCategory")
def displayCategory():
        loggedIn, firstName, noOfItems = getLoginDetails()
        categoryId = request.args.get("categoryId")
        with sqlite3.connect('database.db') as conn:
            cur = conn.cursor()
```

```
            cur.execute("SELECT products.productId, products.name, products.price,
products.image, categories.name FROM products, categories WHERE products.categoryId =
categories.categoryId AND categories.categoryId = ?", (categoryId, ))
            data = cur.fetchall()
        conn.close()
        categoryName = data[0][4]
        data = parse(data)
        return render_template('displayCategory.html', data=data, loggedIn=loggedIn,
firstName=firstName, noOfItems=noOfItems, categoryName=categoryName)
```

PRODUCT DESCRIPTION PAGE:

```
@app.route("/productDescription")
def productDescription():
    loggedIn, firstName, noOfItems = getLoginDetails()
    productId = request.args.get('productId')
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute('SELECT productId, name, price, description, image, stock FROM products
WHERE productId = ?', (productId, ))
        productData = cur.fetchone()
    conn.close()
    return render_template("productDescription.html", data=productData, loggedIn = loggedIn,
firstName = firstName, noOfItems = noOfItems)
```

ADD TO CART:

```
@app.route("/addToCart")
def addToCart():
    if 'email' not in session:
        return redirect(url_for('loginForm'))
    else:
        productId = int(request.args.get('productId'))
        with sqlite3.connect('database.db') as conn:
            cur = conn.cursor()
            cur.execute("SELECT userId FROM users WHERE email = ?", (session['email'], ))
            userId = cur.fetchone()[0]
            try:
                cur.execute("INSERT INTO kart (userId, productId) VALUES (?, ?)", (userId,
productId))
                conn.commit()
                msg = "Added successfully"
            except:
                conn.rollback()
                msg = "Error occured"
        conn.close()
        return redirect(url_for('root'))

@app.route("/cart")
def cart():
    if 'email' not in session:
        return redirect(url_for('loginForm'))
```

```
    loggedIn, firstName, noOfItems = getLoginDetails()
    email = session['email']
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute("SELECT userId FROM users WHERE email = ?", (email, ))
        userId = cur.fetchone()[0]
        cur.execute("SELECT products.productId, products.name, products.price,
products.image FROM products, kart WHERE products.productId = kart.productId AND kart.userId
= ?", (userId, ))
        products = cur.fetchall()
    totalPrice = 0
    for row in products:
        totalPrice += row[2]
    return render_template("cart.html", products = products, totalPrice=totalPrice,
loggedIn=loggedIn, firstName=firstName, noOfItems=noOfItems)
```

REMOVE IETM FROM CART:

```
@app.route("/removeFromCart")
def removeFromCart():
    if 'email' not in session:
        return redirect(url_for('loginForm'))
    email = session['email']
    productId = int(request.args.get('productId'))
    with sqlite3.connect('database.db') as conn:
        cur = conn.cursor()
        cur.execute("SELECT userId FROM users WHERE email = ?", (email, ))
        userId = cur.fetchone()[0]
        try:
            cur.execute("DELETE FROM kart WHERE userId = ? AND productId = ?", (userId,
productId))
            conn.commit()
            msg = "removed successfully"
        except:
            conn.rollback()
            msg = "error occured"
    conn.close()
    return redirect(url_for('root'))
```

SIGN OUT:

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

EMAIL & PASSWORD VALIDATION:

```
def is_valid(email, password):
    con = sqlite3.connect('database.db')
    cur = con.cursor()
    cur.execute('SELECT email, password FROM users')
    data = cur.fetchall()
    for row in data:
        if row[0] == email and row[1] == hashlib.md5(password.encode()).hexdigest():
            return True
    return False
```

PARSE DATA:

```
def parse(data):
    ans = []
    i = 0
    while i < len(data):
        curr = []
        for j in range(7):
            if i >= len(data):
                break
            curr.append(data[i])
            i += 1
        ans.append(curr)
    return ans
```

RUN:

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

CART ITEMS PAGE: