Phase 4 - E-Commerce Application on IBM Cloud Foundry

DEVELOPMENT PHASE – II

**Problem Statement**:

Build a artisanal e-Commerce platform using IBM foundry. Connect the skilled artisans with the global audience. Showcase handmade products from exquisite jewellry to artistic home decor. Implement secure shopping carts, smooth payment gateway and an intuitive checkout process. Nurture creativity and support small business through an artisan's dream marketplace.

**Development phase – II**

This was the problem statement had been given to us. In the previous phases we had already given the problem definition, requirements, diagrammatic representation for the problem, and the ideas to implement the features like wish list, review, and personalized recommendation. By following these components we had developed the e-commerce application.

**LANGUAGES USED:**

* HTML
* Python with Flask
* CSS
* SQLite

In previous phase we had been developed the following items:

* Database
* Layouts
* Stylesheets

In this phase we will be updating the database by adding products, users to the e-commerce application.

**DATABASE:**

UPDATE products set price = '999' WHERE productId = 2;

UPDATE products set price = '399' WHERE productId = 3;

UPDATE products set price = '4499' WHERE productId = 4;

UPDATE products set price = '99' WHERE productId = 5;

UPDATE products set price = '9999' WHERE productId = 6;

UPDATE products set price = '599' WHERE productId = 7;

UPDATE products set price = '899' WHERE productId = 8;

UPDATE products set price = '599' WHERE productId = 9;

UPDATE products set price = '499' WHERE productId = 10;

UPDATE products set price = '299' WHERE productId = 11;

SELECT \* FROM products;

UPDATE products set price = '899' WHERE productId = 16;

UPDATE products set name = 'Jewellery' WHERE productId = 16;

UPDATE products set description = 'Fit to Any Dress' WHERE productId = 16;

UPDATE products set image = 'jewel\_2.jpeg' WHERE productId = 16;

UPDATE products set stock = '200' WHERE productId = 16;

UPDATE products set categoryId = 'J29' WHERE productId = 16;

UPDATE categories set name = 'Wearings' WHERE categoryId = 1;

UPDATE categories set name = 'Hangings' WHERE categoryId = 2;

UPDATE categories set name = 'Handmade' WHERE categoryId = 3;

SELECT \*FROM categories;

DELETE FROM categories WHERE categoryId = 4 ;

DELETE FROM categories WHERE categoryId = 5 ;

DELETE FROM categories WHERE categoryId = 6 ;

SELECT \* FROM categories;

UPDATE products set price = '899' WHERE productId = 17;

UPDATE products set name = 'Saree' WHERE productId = 17;

UPDATE products set description = 'Gorgious Looking' WHERE productId = 17;

UPDATE products set image = 'saree.png' WHERE productId = 17;

UPDATE products set stock = '200' WHERE productId = 17;

UPDATE products set categoryId = 'RI23' WHERE productId = 17;

SELECT \* FROM products;

UPDATE products set image = 'jewel\_3.png' WHERE productId = 11;

UPDATE products set image = 'jewel\_2.png' WHERE productId = 16;

UPDATE products set image = 'jewellery.png' WHERE productId = 8;

UPDATE products set price = '499' WHERE productId = 12;

UPDATE products set name = 'Sandal' WHERE productId = 12;

UPDATE products set description = 'Good Quality, Good Comfort' WHERE productId = 12;

UPDATE products set image = 'sandal1.png' WHERE productId = 12;

UPDATE products set stock = '200' WHERE productId = 12;

UPDATE products set categoryId = 'S10' WHERE productId = 12;

SELECT \* FROM products;

UPDATE products set price = '499' WHERE productId = 14;

UPDATE products set name = 'Sandal' WHERE productId = 14;

UPDATE products set description = 'Good Quality, Good Comfort' WHERE productId = 14;

UPDATE products set image = 'sandal2.png' WHERE productId = 14;

UPDATE products set stock = '200' WHERE productId = 14;

UPDATE products set categoryId = 'S10' WHERE productId = 14;

UPDATE products set price = '7999' WHERE productId = 13;

UPDATE products set name = 'Wall Painting' WHERE productId = 13;

UPDATE products set description = 'Amazing and Give a Fabulous Look' WHERE productId = 13;

UPDATE products set image = 'paint1.png' WHERE productId = 13;

UPDATE products set stock = '100' WHERE productId = 13;

UPDATE products set categoryId = 'WP23' WHERE productId = 13;

UPDATE products set price = '7599' WHERE productId = 15;

UPDATE products set name = 'Wall Painting' WHERE productId = 15;

UPDATE products set description = 'Amazing and Give a Fabulous Look' WHERE productId = 15;

UPDATE products set image = 'paint2.png' WHERE productId = 15;

UPDATE products set stock = '100' WHERE productId = 15;

UPDATE products set categoryId = 'WP23' WHERE productId = 15;

By the above queries the product are added to the website when the user updates the product then the database will be change.

**PYTHON with Flask:**

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'])

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

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)

@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)

@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'])

#Uploading image procedure

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'))

@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'))

@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)

@app.route("/account/profile")

def profileHome():

if 'email' not in session:

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

loggedIn, firstName, noOfItems = getLoginDetails()

return render\_template("profileHome.html", loggedIn=loggedIn, firstName=firstName, noOfItems=noOfItems)

@app.route("/account/profile/edit")

def editProfile():

if 'email' not in session:

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

loggedIn, firstName, noOfItems = getLoginDetails()

with sqlite3.connect('database.db') as conn:

cur = conn.cursor()

cur.execute("SELECT userId, email, firstName, lastName, address1, address2, zipcode, city, state, country, phone FROM users WHERE email = ?", (session['email'], ))

profileData = cur.fetchone()

conn.close()

return render\_template("editProfile.html", profileData=profileData, loggedIn=loggedIn, firstName=firstName, noOfItems=noOfItems)

@app.route("/account/profile/changePassword", methods=["GET", "POST"])

def changePassword():

if 'email' not in session:

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

if request.method == "POST":

oldPassword = request.form['oldpassword']

oldPassword = hashlib.md5(oldPassword.encode()).hexdigest()

newPassword = request.form['newpassword']

newPassword = hashlib.md5(newPassword.encode()).hexdigest()

with sqlite3.connect('database.db') as conn:

cur = conn.cursor()

cur.execute("SELECT userId, password FROM users WHERE email = ?", (session['email'], ))

userId, password = cur.fetchone()

if (password == oldPassword):

try:

cur.execute("UPDATE users SET password = ? WHERE userId = ?", (newPassword, userId))

conn.commit()

msg="Changed successfully"

except:

conn.rollback()

msg = "Failed"

return render\_template("changePassword.html", msg=msg)

else:

msg = "Wrong password"

conn.close()

return render\_template("changePassword.html", msg=msg)

else:

return render\_template("changePassword.html")

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

def updateProfile():

if request.method == 'POST':

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('UPDATE users SET firstName = ?, lastName = ?, address1 = ?, address2 = ?, zipcode = ?, city = ?, state = ?, country = ?, phone = ? WHERE email = ?', (firstName, lastName, address1, address2, zipcode, city, state, country, phone, email))

con.commit()

msg = "Saved Successfully"

except:

con.rollback()

msg = "Error occured"

con.close()

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

@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)

@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)

@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)

@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'))

@app.route("/logout")

def logout():

session.pop('email', None)

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

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

@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

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

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

app.run(debug=True)

**Register.html**

<html>

<head>

<title>Registration</title>

<script type="text/javascript" src="{{ url\_for('static', filename = 'js/validateForm.js') }}">

</script>

</head>

<body>

<form action="/register" method="POST" onsubmit="return validate()">

<p>Email: <input type="email" name="email"></p>

<P>Password: <input type="password" name="password" id="password" required></p>

<p>Confirm Password: <input type="password" name="cpassword" id="cpassword"></p>

<p>First Name: <input type="text" name="firstName"></p>

<p>Last Name: <input type="text" name="lastName"></p>

<p>Address Line 1: <input type="text" name="address1"></p>

<p>Address Line 2: <input type="text" name="address2"></p>

<p>Zipcode: <input type="text" name="zipcode"></p>

<p>City: <input type="text" name="city"></p>

<p>State: <input type="text" name="state"></p>

<p>Country: <input type="text" name="country"></p>

<p>Phone Number: <input type="text" name="phone"></p>

<p><input type="submit" value="Register"></p>

<a href="/loginForm">Login here</a>

</form>

</body></html>

**LOGIN.html**

<html>

<head>

<title> First flask app </title>

</head>

<body>

<p> {{error}} </p>

<form action="/login" method="POST">

<p>Email: <input type="text" name="email"></p>

<p>Password: <input type="password" name="password"></p>

<p><input type="submit"></p>

<a href="/registerationForm">Register here</a>

</form>

</body>

</html>

**Productdescription.html**

<!DOCTYPE HTML>

<html>

<head>

<title>Product Description</title>

<link rel="stylesheet" href={{url\_for('static', filename='css/productDescription.css')}} />

<link rel="stylesheet" href={{ url\_for('static', filename='css/topStyle.css')}} />

</head>

<body>

<div id="title">

<a href="/">

<img id="logo" src= {{ url\_for('static', filename='images/logo.png') }} />

</a>

<form>

<input id="searchBox" type="text" name="searchQuery">

<input id="searchButton" type="submit" value="Search">

</form>

{% if not loggedIn %}

<div id="signInButton">

<a class="link" href="/loginForm">Sign In</a>

</div>

{% else %}

<div class="dropdown">

<button class="dropbtn">Hello, <br>{{firstName}}</button>

<div class="dropdown-content">

<a href="/account/orders">Your orders</a>

<a href="/account/profile">Your profile</a>

<hr>

<a href="/logout">Sign Out</a>

</div>

</div>

{% endif %}

<div id="kart">

<a class="link" href="/cart">

<img src={{url\_for('static', filename='images/shoppingCart.png')}} id="cartIcon" />

CART {{noOfItems}}

</a>

</div>

</div>

<div id="display">

<div id="productName">

<h1>{{data[1]}}</h1>

</div>

<div>

<img src={{url\_for('static', filename='uploads/'+data[4]) }} id="productImage"/>

</div>

<div id="productDescription">

<h2>Details</h2>

<table id="descriptionTable">

<tr>

<td>Name</td>

<td>{{data[1]}}</td>

</tr>

<tr>

<td>Price</td>

<td>${{data[2]}}</td>

</tr>

<tr>

<td>Stock</td>

<td>{{data[5]}}</td>

</tr>

</table>

<h2>Description</h2>

<p>{{data[3]}}</p>

</div>

<div id="addToCart">

<a href="/addToCart?productId={{request.args.get('productId')}}">Add to Cart</a>

</div>

</div>

</body>

</html>

**Changepassword.html**

<html>

<head>

<title>Change Password</title>

<script src={{ url\_for('static', filename='js/changePassword.js') }}></script>

</head>

<body>

<h1>Change password</h1>

<p>{{ msg }}</p>

<form action={{ url\_for('changePassword') }} method="POST" onsubmit="return validate()">

<p>Old Password: <input type="password" name="oldpassword"></p>

<p>New Password: <input type="password" name="newpassword" id="newpassword"></p>

<p>Confirm Password: <input type="password" name="cpassword" id="cpassword"></p>

<input type="submit" value="Save">

</form>

<a href="{{ url\_for('profileHome') }}">Go to Profile</a>

</body>

</html>

**Displaycategory.html**

<!DOCTYPE HTML>

<html>

<head>

<title>Category: {{categoryName}}</title>

<link rel="stylesheet" href={{ url\_for('static', filename='css/home.css') }} />

<link rel="stylesheet" href={{ url\_for('static', filename='css/topStyle.css') }} />

</head>

<body>

<div id="title">

<a href="/">

<img id="logo" src= {{ url\_for('static', filename='images/logo.png') }} />

</a>

<form>

<input id="searchBox" type="text" name="searchQuery">

<input id="searchButton" type="submit" value="Search">

</form>

{% if not loggedIn %}

<div id="signInButton">

<a class="link" href="/loginForm">Sign In</a>

</div>

{% else %}

<div class="dropdown">

<button class="dropbtn">Hello, <br>{{firstName}}</button>

<div class="dropdown-content">

<a href="/account/orders">Your orders</a>

<a href="/account/profile">Your profile</a>

<hr>

<a href="/logout">Sign Out</a>

</div>

</div>

{% endif %}

<div id="kart">

<a class="link" href="/cart">

<img src={{url\_for('static', filename='images/shoppingCart.png')}} id="cartIcon" />

CART {{noOfItems}}

</a>

</div>

</div>

<div>

<h2>Showing all products of Category {{categoryName}}:</h2>

{% for itemData in data %}

<table>

<tr id="productName">

{% for row in itemData %}

<td>

{{row[1]}}

</td>

{% endfor %}

</tr>

<tr id="productImage">

{% for row in itemData %}

<td>

<a href="/productDescription?productId={{row[0]}}">

<img src={{ url\_for('static', filename='uploads/' + row[3]) }} id="itemImage" />

</a>

</td>

{% endfor %}

</tr>

<tr id="productPrice">

{% for row in itemData %}

<td>

${{row[2]}}

</td>

{% endfor %}

</tr>

</table>

{% endfor %}

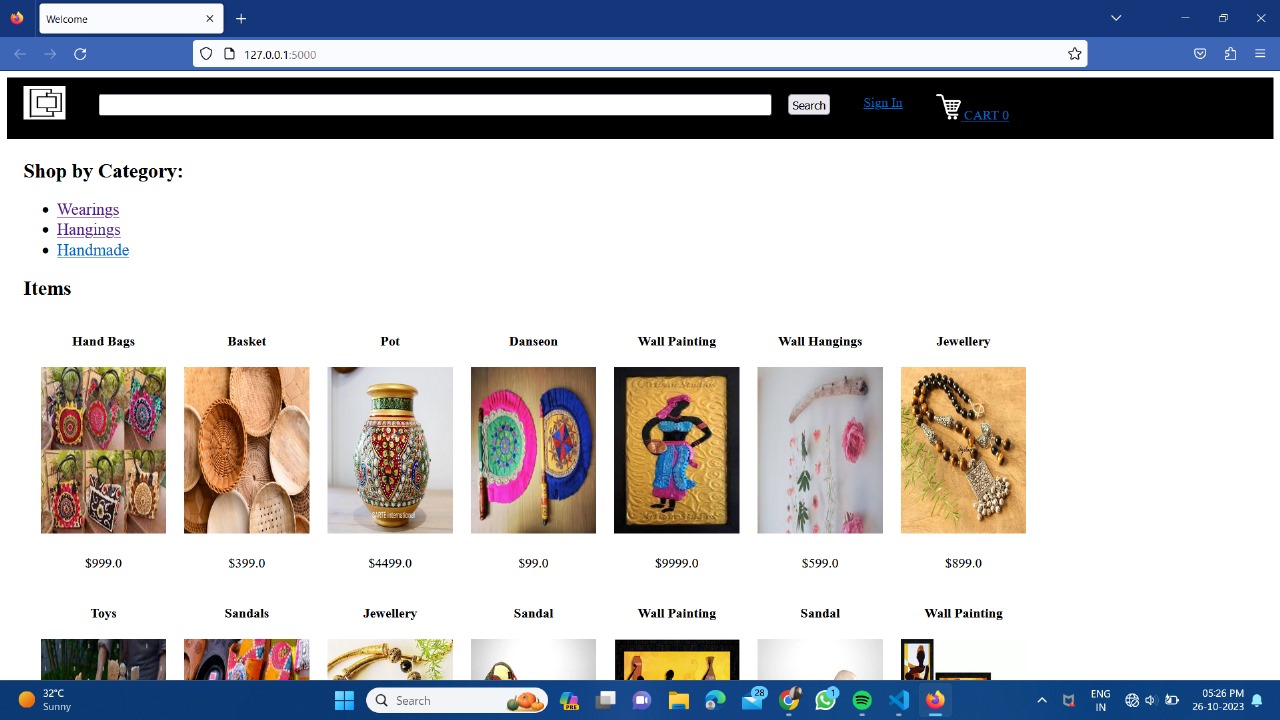
</div>

</body>

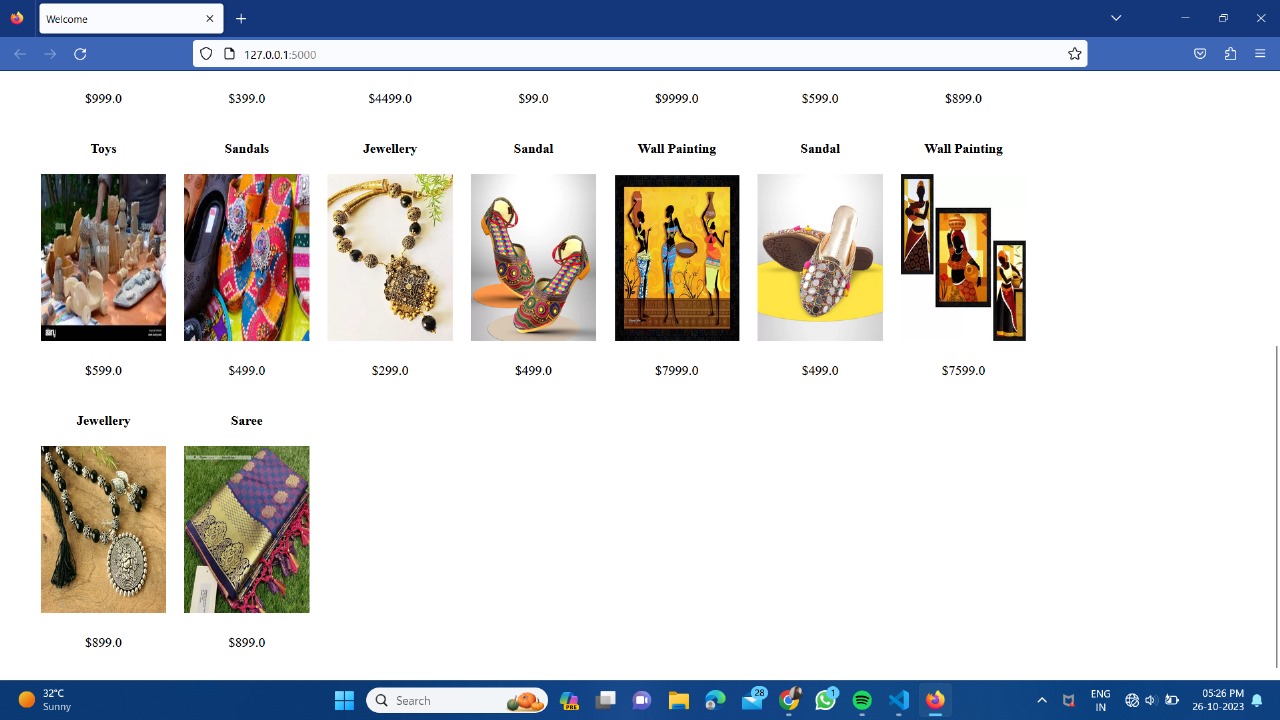
</html>

**Screenshot:**

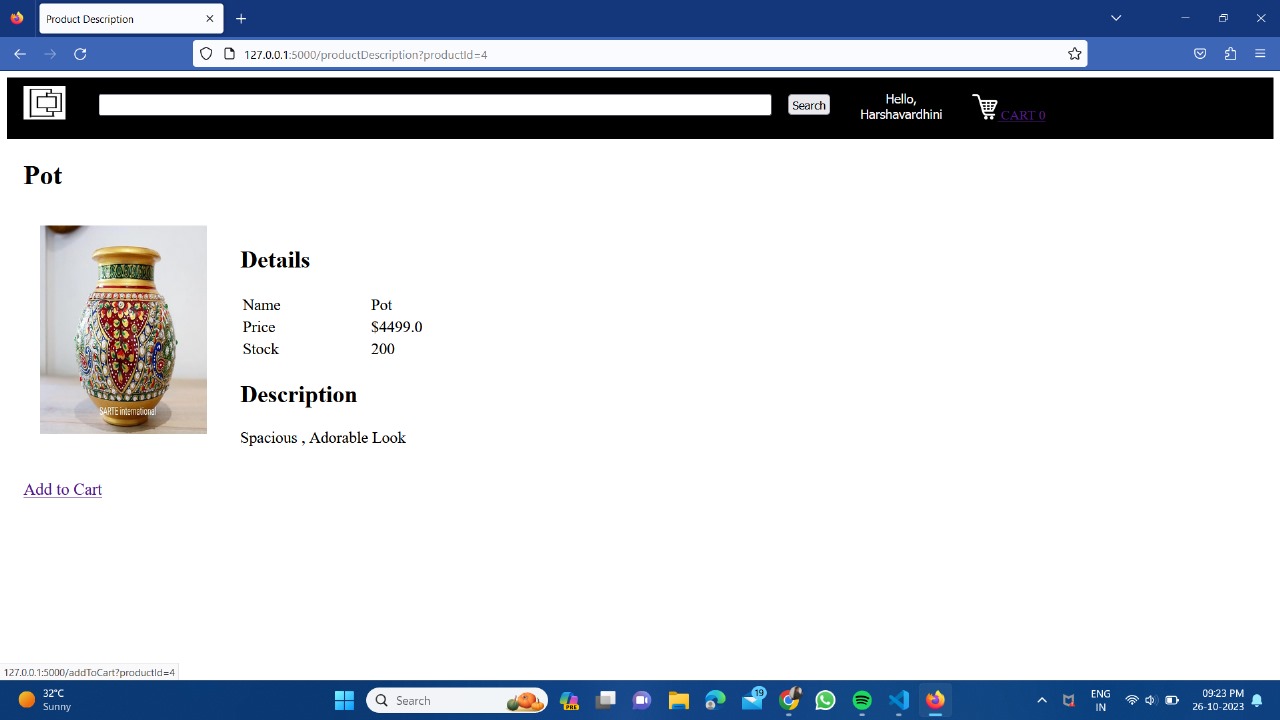
This is the home page of e-commerce application

****

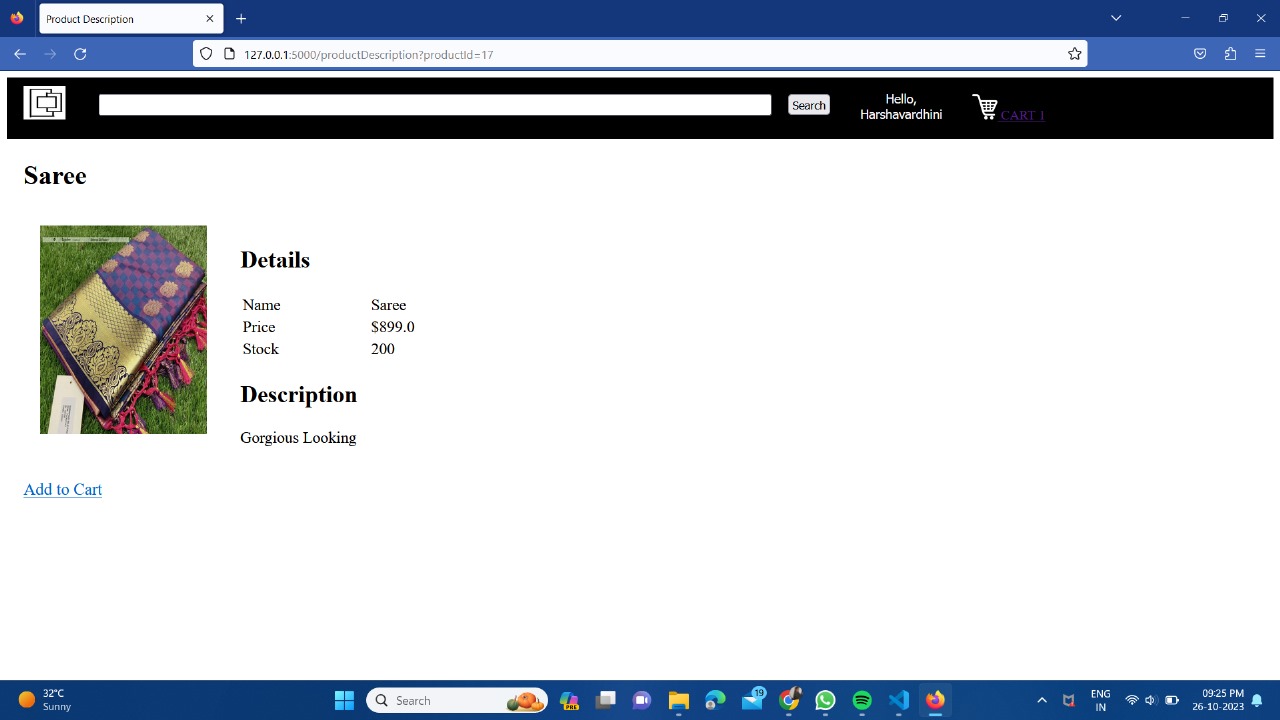
In this the user will be search for the product that is needed



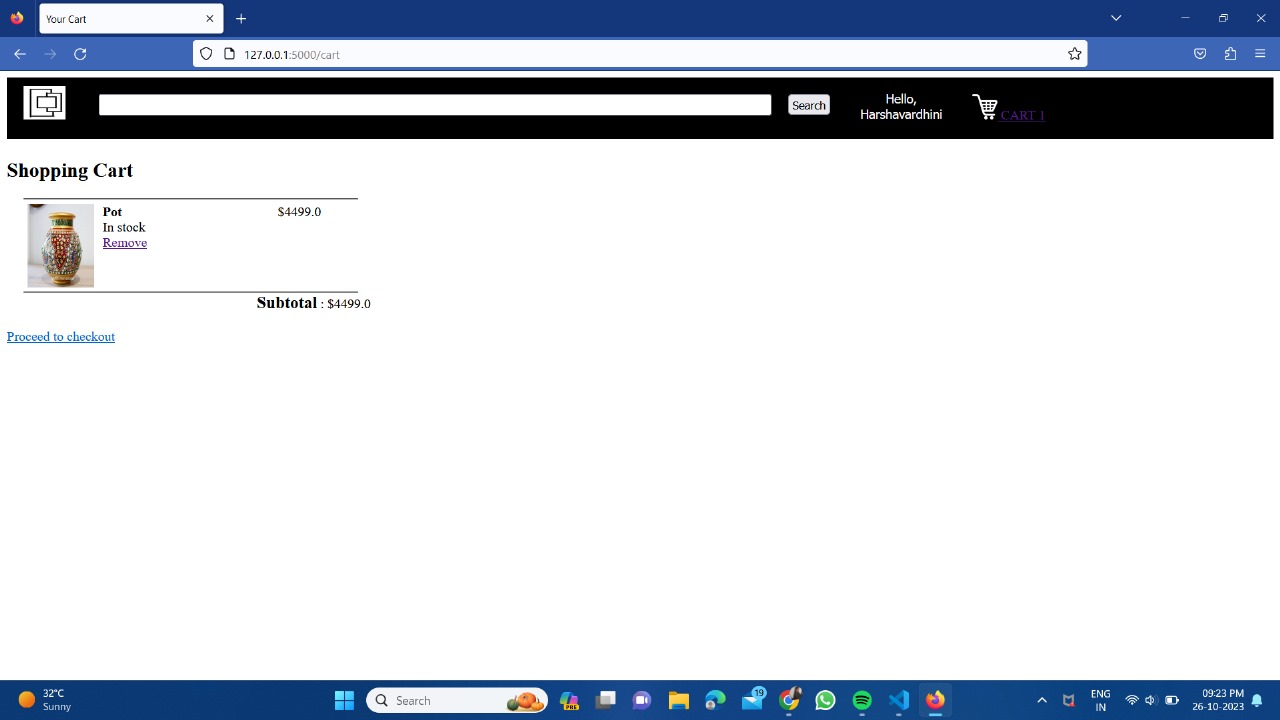
Then the user can select the particular product and view the product description and price of the product.

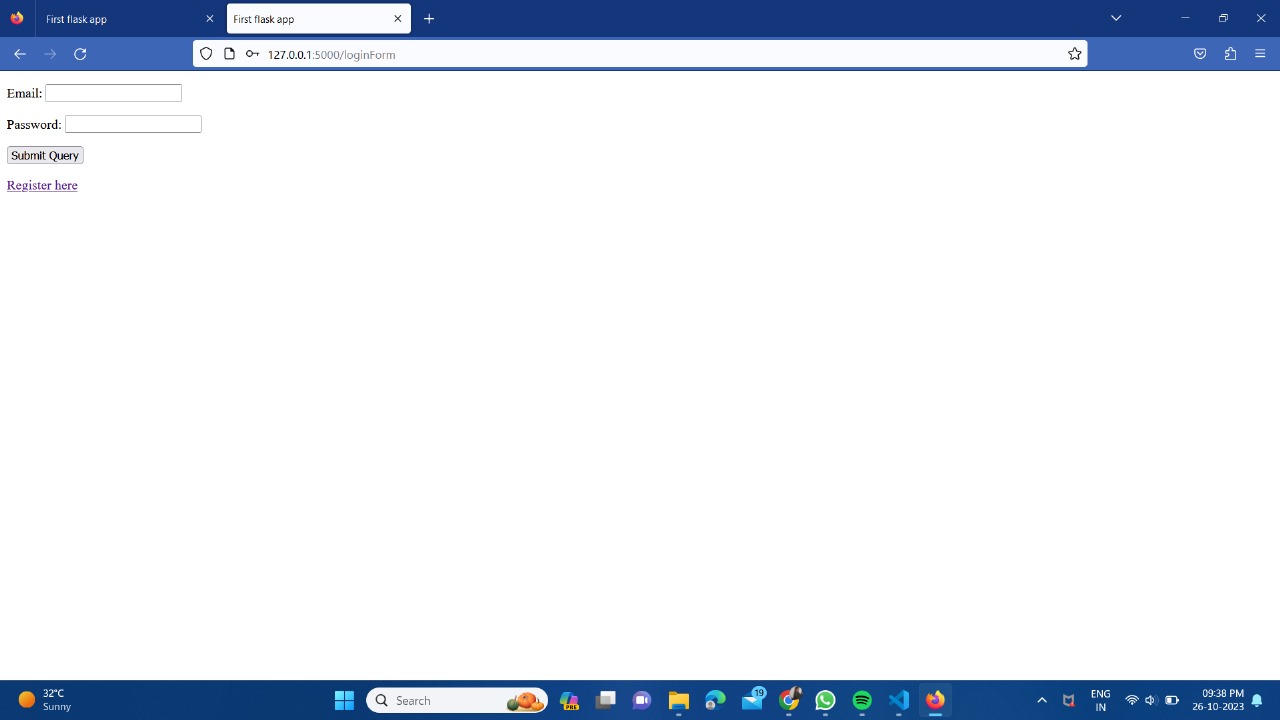


This a another page of the product.

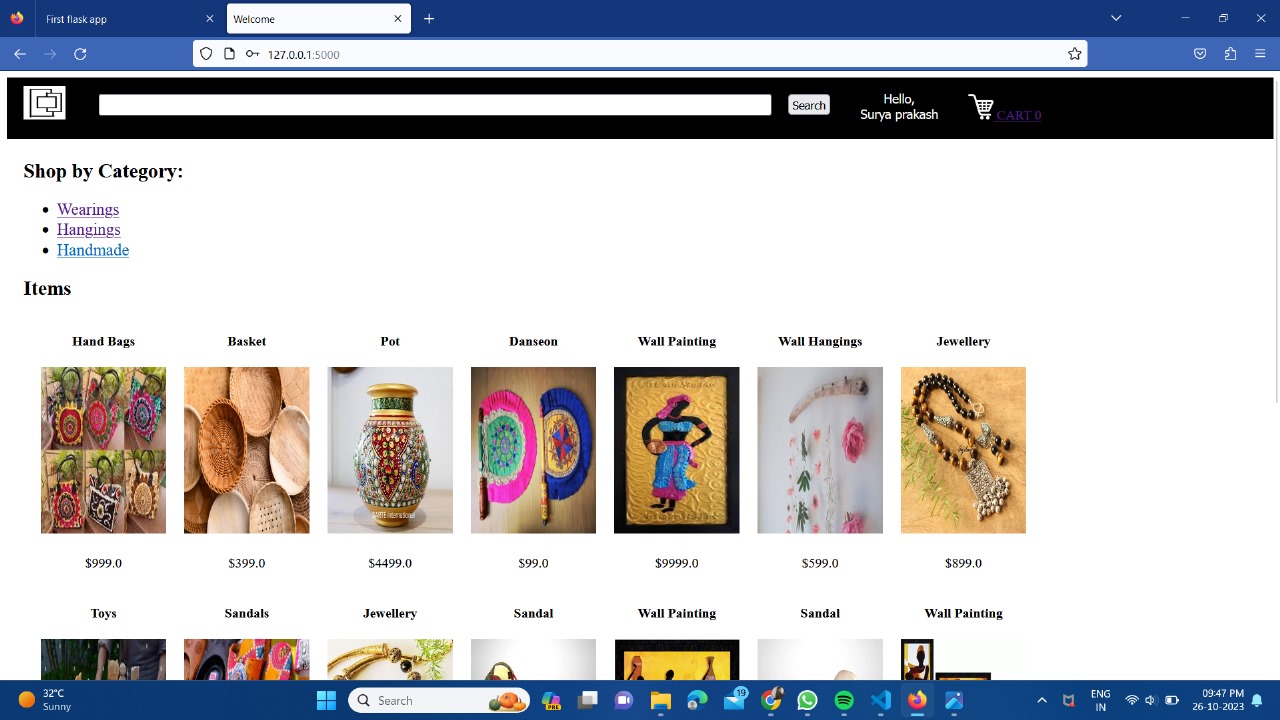


After viewing the product then, the user can add the product to cart and also we can add this product to wishlist.

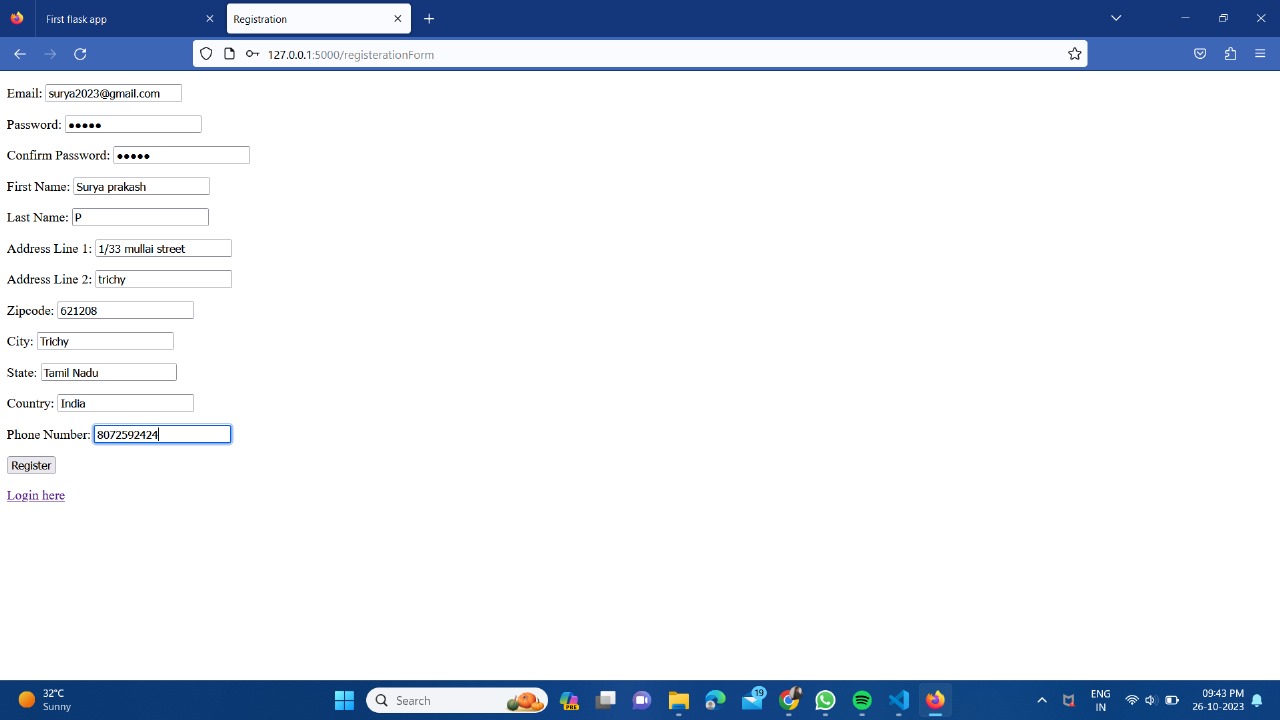


If the user is already have an account then, the user can login using credentials. 

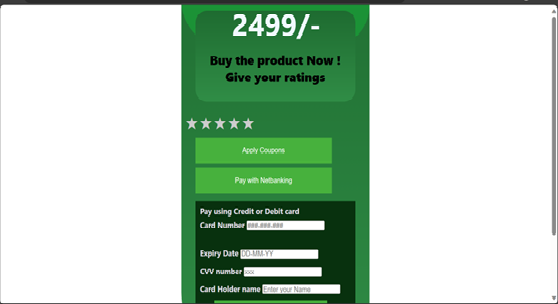
If the user is already have an account then the user will be redirected to his account, that will be display as “**HELLO username”.**



If the user doesn’t have an account, then the user can register himself to the application. Then he can create his own account.



For purchasing the product the user must be complete the payment process.



The payment can be through credit card or debit card and the user can also review the product by submitting the stars so that the other user can get the knowledge about the product.