PROJECT DEVELOPMENT PHASE

SPRINT-2

DATE	05 NOVEMBER 2022
TEAM ID	PNT2022TMID52517
PROJECT NAME	SMART FASHION RECOMMENDERAPPLICATION

PRODUCT LIST PAGE:

```
<section class="container-fluid pt-3 pt-md-4">
   <!-- search filters -->
   <div class="row">
       <div class="col-12 col-md-6">
           <div class="d-flex border">
               <select name="category" class="border border-end-
0border-dark py-3">
                   <option value="all">All</option>
                   <option value="t-shirt">T-shirt</option>
                   <option value="t-shirt">T-shirt</option>
                   <option value="t-shirt">T-shirt</option>
               </select>
               <input type="text" placeholder="search" style="flex:
1;"class="border-0 py-2">
           </div>
       </div>
       <div class="col-12 col-md-6">
           <div class="d-flex justify-content-
aroundjustify-content-md-end mt-3 mt-md-0">
               <!-- chat bot -->
               <button class="chat-btn me-md-3">Let's chat</button>
               <!-- sort by -->
               <select name="sort" class="border-0 py-3">
                   <option value="">Sort by</option>
                   <option value="low-high">Price: Low to High</option>
                   <option value="high-low">Price: High to Low</option>
               </select>
           </div>
       </div>
   </div>
   <!-- product list items -->
```

```
    <div class="row mt-2">
        <div class="col-12">
            <div class="d-flex justify-content-between">
                <p class="fw-bold">Total products : 545</p>
                <p class="d-none d-md-block">Let's make changes in
buying</p>
            </div>
            <app-product-card
[product]="productDatails"></app-product-card>
        </div>
        <!-- pagination -->
        <div class="col-12 mt-5 d-flex justify-content-center">
            <nav aria-label="Page navigation example">
                <ul class="pagination">
                    <li class="page-item"><a class="page-link"
href="#">1</a></li>
                    <li class="page-item"><a class="page-link"
href="#">2</a></li>
                    <li class="page-item"><a class="page-link"
href="#">3</a></li>
                    <li class="page-item"><a class="page-link"
href="#">Next</a></li>
                </ul>
            </nav>
        </div>
    </div>
</section>
```

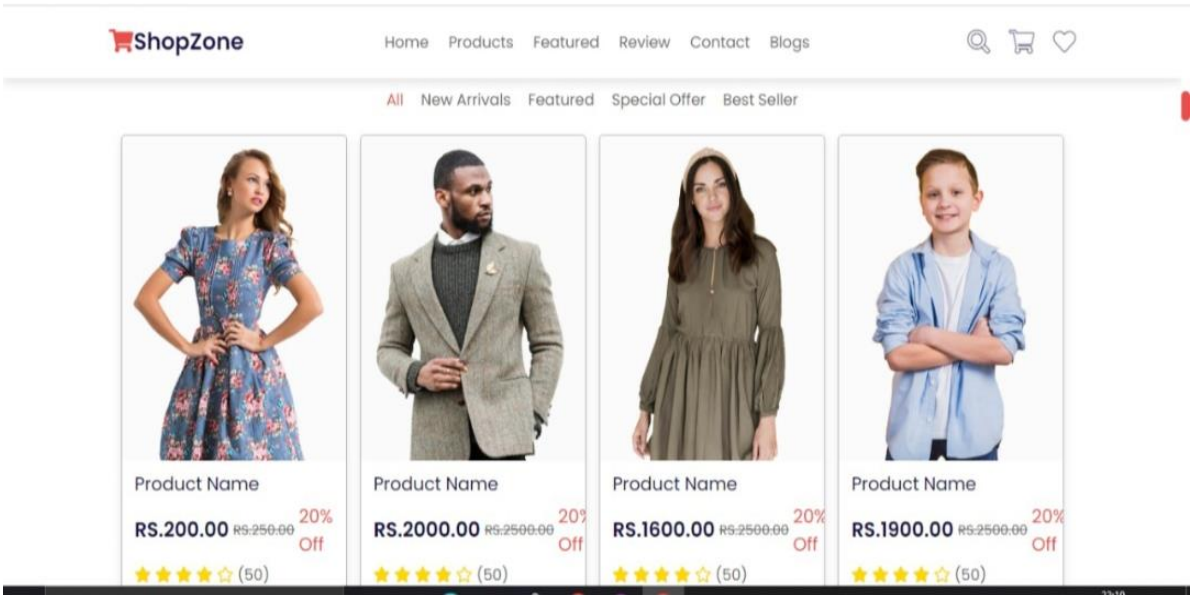
CSS:

```
.chat-btn
    {border:none;
    padding: 0.7rem 2.5rem;
    background-color: var(--primary-color);
}

.chat-btn:hover {
    border: 3px solid var(--primary-
    color);background-color: transparent;
}

.pagination .page-link
    {color: black;
    border: 3px solid var(--primary-color);
```

OUTPUT SCREEN:



BACKEND API'S:

Login API

```
from flask import
Blueprint,jsonify,g,requestimport ibm_db
from passlib.hash import
sha256_cryptimport jwt

from ..lib import
validation_errorfrom ..lib import
exception
from ..lib import db

auth_bp = Blueprint("auth",name)



@auth_bp.route("/",methods=["GET"])
def check():
   print(g.get("db"))
   return jsonify({"msg":"hi"})

@auth_bp.route('/register',methods=['POST'])
def reg():
   try:
       data =
       request.get_json()name=da
       ta['name']
```

```
        email=data['email']password=data['pa
        ssword']mobile_no=data['mobileNo']pr
        int(email,password,name,mobile_no)in
        sert_sql="INSERT INTO
USER(name,email,password,role,mobilenumber)
        VALUES(?,?,?,?,?)"prep_stmt = ibm_db.prepare(db.get_db(),
        insert_sql)ibm_db.bind_param(prep_stmt,1,name)ibm_db.bind_par
        am(prep_stmt,2,email)ibm_db.bind_param(prep_stmt,3,sha256_cry
        pt.encrypt(password))ibm_db.bind_param(prep_stmt,4,"user")ibm
        _db.bind_param(prep_stmt,5,mobile_no)ibm_db.execute(prep_stmt
        )
        return {"message":'Created'},201

    except Exception as e:
            return exception.handle_exception(e)


@auth_bp.route('/me',methods=['GET'])
def getMe():
    try:
        token =
        request.headers['Authorization']if (not
        token):
            return validation_error.throw_validation("Please
        login",401)decoded =
        jwt.decode(token,"secret",algorithms=["HS256"])select_sql =
        "SELECT * FROM USER WHERE ID=?"
        prep_stmt = ibm_db.prepare(db.get_db(),
        select_sql)ibm_db.bind_param(prep_stmt,1,decoded['i
        d'])ibm_db.execute(prep_stmt)isUser=ibm_db.fetch_as
        soc(prep_stmt)
        return
    isUserexceptExceptiona
    se:
            return exception.handle_exception(e)


@auth_bp.route('/login',methods=['POST'])
def auth_log():
    try:
        data =
        request.get_json()print(d
        ata)email=data['email']pa
        ssword=data['password']
```

```
        select_sql = "SELECT * FROM USER WHERE EMAIL=?"
        prep_stmt = ibm_db.prepare(db.get_db(),
        select_sql)ibm_db.bind_param(prep_stmt,1,email)ibm_
        db.execute(prep_stmt)isUser=ibm_db.fetch_assoc(prep
        _stmt)
        print(isUser)i
        fnotisUser:
            return
validation_error.throw_validation("InvalidCredentials",400)
        if not
            sha256_crypt.verify(password,isUser['PASSWORD']):ret
            urn validation_error.throw_validation("Invalid
Credentials",400)
        encoded_jwt
=jwt.encode({"id":isUser['ID'],"role":isUser['ROLE']},"secret",algorith
m
="HS256")
        isUser["token"] =
        encoded_jwtreturn isUser
   except Exception as e:
            return exception.handle_exception(e)
```

Category API

```
from flask import
Blueprint,requestimport ibm_db
from ..lib import
exceptionfrom ..lib import
db


category_bp = Blueprint("category",name)


@category_bp.route("/",methods=["GET"])
def get_category():
 try:
    select_sql = "SELECT * FROM CATEGORY WHERE"
    prep_stmt = ibm_db.prepare(db.get_db(),
    select_sql)ibm_db.execute(prep_stmt)
    categories=[]category=ibm_db.fetch_ass
    oc(prep_stmt)while(category != False):
      categories.append(category)
      category =
    ibm_db.fetch_assoc(prep_stmt)print(categor
    ies)
```

```
    return
 categories,200except
 Exception as e:
    return exception.handle_exception(e)

@category_bp.route("/",methods=["POST"])
def add_category():
 try:
    data =
    request.get_json()category=
    data['category']
    insert_sql="INSERT INTO CATEGORY(category_name)
    VALUES(?)"prep_stmt = ibm_db.prepare(db.get_db(),
    insert_sql)ibm_db.bind_param(prep_stmt,1,category)ibm_db.e
    xecute(prep_stmt)
    return
 {"message":'Created'},201except
 Exception as e:
    return exception.handle_exception(e)
```

```
@category_bp.route("/<id>",methods=["DELETE"])
def get_category_id(id):
 try:
    print(id)
    select_sql = "DELETE FROM CATEGORY WHERE ID=?"
    prep_stmt = ibm_db.prepare(db.get_db(),
    select_sql)ibm_db.bind_param(prep_stmt,1,id)ibm_db.
    execute(prep_stmt)


    return
            {"message":'Deleted'},200
 except Exception as e:
```

Product API

```
from flask import
Blueprint,requestimport ibm_db
from ..lib import
exceptionfrom ..lib import
db


product_bp = Blueprint("product",name)
```

```
@product_bp.route("/",methods=['POST'])
def add_product():
 try:
    data =
    request.get_json()name=data['name
    ']category=data['category']descri
    ption =
    data['description']stock=data['st
    ock']
    specificity =
    data['specificity']price =
    data['price']brand=data['brand']i
    nsert_sql="INSERT INTO
PRODUCT(product_name,category,description,stock,specificity,price,brand
) VALUES(?,?,?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(db.get_db(),
    insert_sql)ibm_db.bind_param(prep_stmt,1,name)ibm_d
    b.bind_param(prep_stmt,2,category)ibm_db.bind_param
    (prep_stmt,3,description)ibm_db.bind_param(prep_stm
    t,4,stock)ibm_db.bind_param(prep_stmt,5,specificity
    )ibm_db.bind_param(prep_stmt,6,price)ibm_db.bind_pa
    ram(prep_stmt,7,brand)ibm_db.execute(prep_stmt)
    return
 {"message":'Created'},201except
 Exception as e:
   return exception.handle_exception(e)

@product_bp.route("/",methods=['GET'])
def get_product():
 try:
   select_sql = "SELECT PRODUCT.ID AS
product_id,category,category_name,product_name,description,price,stock,
image,brand
,specificity FROM PRODUCT JOIN CATEGORY
ONCATEGORY.ID=PRODUCT.CATEGORY"
   prep_stmt = ibm_db.prepare(db.get_db(),
   select_sql)ibm_db.execute(prep_stmt)
   products=[]product=ibm_db.fetch_assoc
   (prep_stmt)while(product != False):
     products.append(product)
     product =
   ibm_db.fetch_assoc(prep_stmt)print(produc
   ts)
   return products or [],200
```

```
 except Exception as e:
   return exception.handle_exception(e)


@product_bp.route("/<id>",methods=['GET'])
def get_product_id(id):
 try:
   select_sql = "SELECT PRODUCT.ID AS
product_id,category,category_name,product_name,description,price,stock,
image,brand
,specificity FROM PRODUCT JOIN CATEGORY ON
CATEGORY.ID=PRODUCT.CATEGORYWHERE PRODUCT.ID=?"
   prep_stmt = ibm_db.prepare(db.get_db(),
   select_sql)ibm_db.bind_param(prep_stmt,1,id)ibm_db.
   execute(prep_stmt)product=ibm_db.fetch_assoc(prep_s
   tmt)print(product)
   return product or
 [],200except Exception as
 e:
   return exception.handle_exception(e)


@product_bp.route("/<id>",methods=['PUT'])
def update_product(id):
 try:
    data =
    request.get_json()name=data['name
    ']category=data['category']descri
    ption =
    data['description']stock=data['st
    ock']
    specificity =
    data['specificity']price =
    data['price']brand=data['brand']i
    nsert_sql="UPDATE PRODUCT SET
product_name=?,category=?,description=?,stock=?,specificity=?,price=?,b
rand=? WHERE ID=?"
    prep_stmt = ibm_db.prepare(db.get_db(),
    insert_sql)ibm_db.bind_param(prep_stmt,1,name)ibm_d
    b.bind_param(prep_stmt,2,category)ibm_db.bind_param
    (prep_stmt,3,description)ibm_db.bind_param(prep_stm
    t,4,stock)ibm_db.bind_param(prep_stmt,5,specificity
    )ibm_db.bind_param(prep_stmt,6,price)ibm_db.bind_pa
    ram(prep_stmt,7,brand)
```

```
    ibm_db.bind_param(prep_stmt,8,id)
    ibm_db.execute(prep_stmt)
    return
 {"message":'Updated'},200except
 Exception as e:
   return exception.handle_exception(e)


@product_bp.route("/<id>",methods=['DELETE'])
def delete_product(id):
 try:
    insert_sql="DELETE FROM PRODUCT WHERE ID=?"
    prep_stmt = ibm_db.prepare(db.get_db(),
    insert_sql)ibm_db.bind_param(prep_stmt,1,id)ibm_db.
    execute(prep_stmt)
    return
 {"message":'Deleted'},200except
 Exception as e:
   return exception.handle_exception(e)
```

Cart API

```
from flask import
Blueprint,requestimport ibm_db
from ..lib import
validation_errorfrom ..lib.auth
import check_authfrom ..lib import
exception
from ..lib import db


cart_bp = Blueprint("cart",name)


@cart_bp.route("/",methods=['POST'])
def add_cart():
 try:
   user_id
   =check_auth(request)data=req
   uest.get_json()product=data[
   'product']
   select_sql = "SELECT * FROM PRODUCT WHERE ID=?"
   prepare_select
   =ibm_db.prepare(db.get_db(),select_sql)ibm_db.bind_par
   am(prepare_select,1,product)ibm_db.execute(prepare_sel
   ect)
   is_product = ibm_db.fetch_assoc(prepare_select)

   print(is_product)
```

```
    if not is_product:
      return validation_error.throw_validation("No Product found",404)

    if(is_product['STOCK']<=0):
      return validation_error.throw_validation("No Stock found",404)

    print("Hey")
    insert_sql="INSERT INTO CART(user,product)
    VALUES(?,?)"prep_stmt = ibm_db.prepare(db.get_db(),
    insert_sql)ibm_db.bind_param(prep_stmt,1,user_id)ibm_db
    .bind_param(prep_stmt,2,product)ibm_db.execute(prep_stm
    t)

    print("heyy")

    update_sql="UPDATE PRODUCT SET stock=? WHERE
    ID=?"update_stmt = ibm_db.prepare(db.get_db(),
    update_sql)ibm_db.bind_param(update_stmt,1,is_product['STOC
    K']-1 or
    0)ibm_db.bind_param(update_stmt,2,product)ibm_db.execute(up
    date_stmt)


    print("sdd")
    return
 {"message":'Created'},201except
 Exception as e:
    return exception.handle_exception(e)

@cart_bp.route("/",methods=['DELETE'])
def delete_user_cart():
 try:
    user_id
    =check_auth(request)insert_sql="DELETEFROMCA
    RTWHEREUSER=?"
    prep_stmt = ibm_db.prepare(db.get_db(),
    insert_sql)ibm_db.bind_param(prep_stmt,1,user_id)

    ibm_db.execute(prep_stmt)
    return
 {"message":'Deleted'},201except
 Exception as e:
    return exception.handle_exception(e)
```

```
@cart_bp.route("/",methods=['GET'])
def get_cart():
 try:
   user_id =check_auth(request)
   insert_sql="SELECT  PRODUCT.ID AS
product_id,cart_id,category,category_name,product_name,description,pric
e,stock,image,brand
,specificity,CART.user as user FROM CART JOIN PRODUCT
ONCART.PRODUCT=PRODUCT.ID JOIN CATEGORY ON PRODUCT.CATEGORY =
CATEGORY.IDWHERE CART.USER=?"
   prep_stmt = ibm_db.prepare(db.get_db(),
   insert_sql)ibm_db.bind_param(prep_stmt,1,user_id)

   ibm_db.execute(prep_stmt)
   products=[]
   product=ibm_db.fetch_assoc(prep_stmt)
   while(product != False):
     products.append(product)
     product =
   ibm_db.fetch_assoc(prep_stmt)print(produc
   ts)
   return products or [],200

 except Exception as e:
   return exception.handle_exception(e)


@cart_bp.route("/<product>/<id>",methods=['DELETE'])
def delete_cart(product,id):
 try:
   user_id
   =check_auth(request)print(pr
   oduct,id,user_id)

   select_sql = "SELECT * FROM PRODUCT WHERE ID=?"
   prepare_select
   =ibm_db.prepare(db.get_db(),select_sql)ibm_db.bind_par
   am(prepare_select,1,product)ibm_db.execute(prepare_sel
   ect)
   is_product =

   ibm_db.fetch_assoc(prepare_select)print(is_prod

   uct)

   if not is_product:
     return validation_error.throw_validation("No Product

   found",404)print("ff")
```

```
    insert_sql="DELETE FROM CART WHERE CART_ID=? AND
    user=?"prep_stmt = ibm_db.prepare(db.get_db(),
    insert_sql)ibm_db.bind_param(prep_stmt,1,id)ibm_db.bind_
    param(prep_stmt,2,user_id)ibm_db.execute(prep_stmt)
    print("aa")
    update_sql="UPDATE PRODUCT SET stock=? WHERE
    ID=?"update_stmt = ibm_db.prepare(db.get_db(),
    update_sql)ibm_db.bind_param(update_stmt,1,is_product[
    'STOCK']+1)ibm_db.bind_param(update_stmt,2,product)ibm
    _db.execute(update_stmt)
    return
 {"message":'Deleted'},200except
 Exception as e:
    return exception.handle_exception(e)
```

OrderAPI:

```
from flask import
Blueprint,requestimport ibm_db
from ..lib import
exceptionfrom ..lib import
db,auth


order_bp = Blueprint("order",name)


@order_bp.route("/",methods=['POST'])
def add_order():
 try:
    user_id
    =auth.check_auth(request)data=req
    uest.get_json()products=data['pro
    ducts']
    insert_sql="SELECT ORDER_ID FROM FINAL TABLE (INSERT
INTOORDER(user) VALUES(?))"
    prep_stmt = ibm_db.prepare(db.get_db(),
    insert_sql)ibm_db.bind_param(prep_stmt,1,user_id)ib
    m_db.execute(prep_stmt)
    order =
    ibm_db.fetch_assoc(prep_stmt)print(or
    der)

    for product in products:
```

```
    print(product)
    insert1_sql="INSERT INTO ORDERDETAIL(order,product)
    VALUES(?,?)"prep1_stmt = ibm_db.prepare(db.get_db(),
    insert1_sql)ibm_db.bind_param(prep1_stmt,1,order['ORDER_ID'])ibm
    _db.bind_param(prep1_stmt,2,product)ibm_db.execute(prep1_stmt)

  return
 {"message":'Created'},201except
 Exception as e:
  return exception.handle_exception(e)


@order_bp.route("/<id>",methods=['GET'])
def get_order(id):
 try:
  insert_sql="SELECT  PRODUCT.ID AS
product_id,category,category_name,product_name,description,price,stock,
image,brand
,specificity,paid FROM ORDERDETAIL JOIN ORDER
ONORDERDETAIL.ORDER=ORDER.ORDER_ID JOIN PRODUCT
ONORDERDETAIL.PRODUCT=PRODUCT.ID JOIN CATEGORY ON PRODUCT.CATEGORY
=CATEGORY.ID WHERE ORDER.USER=?"
  prep_stmt = ibm_db.prepare(db.get_db(),
  insert_sql)ibm_db.bind_param(prep_stmt,1,id)ibm_db.
  execute(prep_stmt)
  products=[]product=ibm_db.fetch_assoc
  (prep_stmt)while(product != False):
    products.append(product)
    product =
  ibm_db.fetch_assoc(prep_stmt)print(produc
  ts)
  return products or [],200

 except Exception as e:
  return exception.handle_exception(e)
```