PROJECT DEVELOPMENT PHASE

SPRINT - 2

DATE	05 NOVEMBER 2022
TEAM ID	PNT2022TMID00109
PROJECT NAME	SMART FASHION RECOMMENDER APPLICATION

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-0
border-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-around
justify-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:

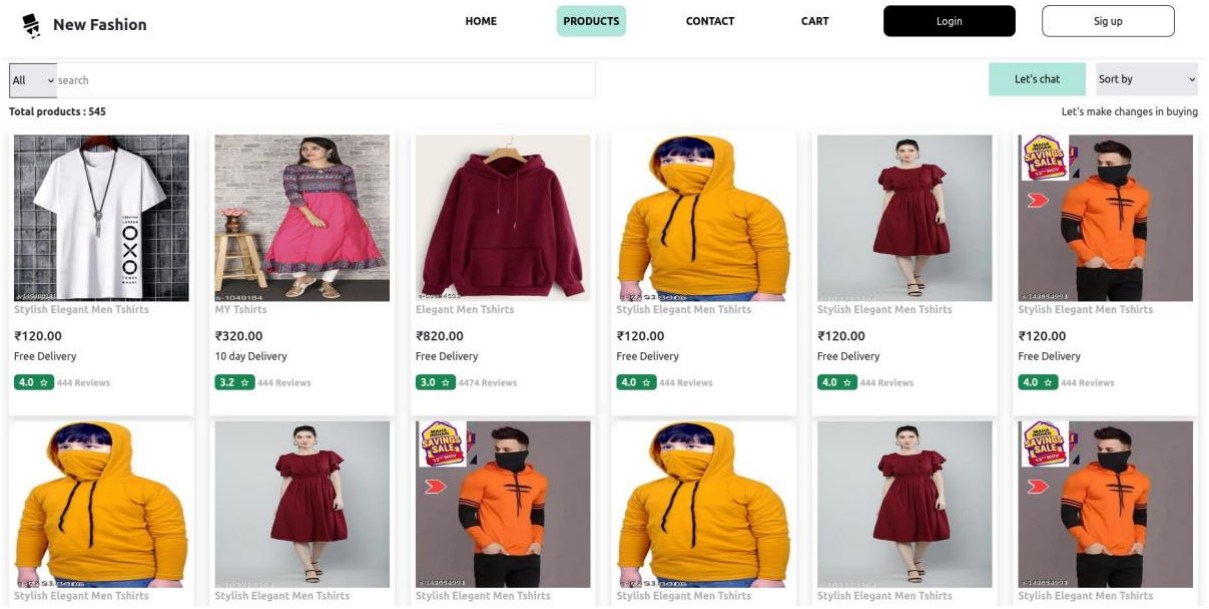
```
.chat-btn {
    border: none;
    padding: 0.7rem 2.5rem;
    background-color: var(--primary-color);
}


.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,request

import ibm_db

from passlib.hash import sha256_crypt

import jwt



from ..lib import validation_error
from ..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=data['name']
       email=data['email']
       password=data['password']
       mobile_no=data['mobileNo']
       print(email,password,name,mobile_no)
       insert_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_param(prep_stmt,2,email)
       ibm_db.bind_param(prep_stmt,3,sha256_crypt.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['id'])
        ibm_db.execute(prep_stmt)
        isUser=ibm_db.fetch_assoc(prep_stmt)
        return isUser
    except Exception as e:
            return exception.handle_exception(e)


@auth_bp.route('/login',methods=['POST'])
def auth_log():
    try:
        data = request.get_json()
        print(data)
        email=data['email']
        password=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)
        if not isUser:
            return validation_error.throw_validation("Invalid
Credentials",400)
        if not sha256_crypt.verify(password,isUser['PASSWORD']):
            return validation_error.throw_validation("Invalid
Credentials",400)
        encoded_jwt =
jwt.encode({"id":isUser['ID'],"role":isUser['ROLE']},"secret",algorithm
="HS256")
        isUser["token"] = encoded_jwt
        return isUser
    except Exception as e:
            return exception.handle_exception(e)
```

Category API

```
from flask import Blueprint,request
import ibm_db
from ..lib import exception
from ..lib import db
```

```python
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_assoc(prep_stmt)
   while(category != False):
     categories.append(category)
     category = ibm_db.fetch_assoc(prep_stmt)
   print(categories)
   return categories,200
 except 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.execute(prep_stmt)
   return {"message":'Created'},201
 except 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:
   return exception.handle_exception(e)
```

```
from flask import Blueprint,request
import ibm_db
from ..lib import exception
from ..lib import db


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

Product API

```
@product_bp.route("/",methods=['POST'])
def add_product():
 try:
    data = request.get_json()
    name=data['name']
    category=data['category']
    description = data['description']
    stock=data['stock']
    specificity = data['specificity']
    price = data['price']
    brand=data['brand']
    insert_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_db.bind_param(prep_stmt,2,category)
    ibm_db.bind_param(prep_stmt,3,description)
    ibm_db.bind_param(prep_stmt,4,stock)
    ibm_db.bind_param(prep_stmt,5,specificity)
    ibm_db.bind_param(prep_stmt,6,price)
    ibm_db.bind_param(prep_stmt,7,brand)
    ibm_db.execute(prep_stmt)
    return {"message":'Created'},201
 except 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 ON
CATEGORY.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(products)
    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.CATEGORY
WHERE 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_stmt)
    print(product)
    return product or [],200
 except 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']
     description = data['description']
     stock=data['stock']
     specificity = data['specificity']
     price = data['price']
     brand=data['brand']
     insert_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_db.bind_param(prep_stmt,2,category)
    ibm_db.bind_param(prep_stmt,3,description)
    ibm_db.bind_param(prep_stmt,4,stock)
    ibm_db.bind_param(prep_stmt,5,specificity)
    ibm_db.bind_param(prep_stmt,6,price)
    ibm_db.bind_param(prep_stmt,7,brand)
    ibm_db.bind_param(prep_stmt,8,id)
    ibm_db.execute(prep_stmt)
    return {"message":'Updated'},200
 except 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'},200
 except Exception as e:
   return exception.handle_exception(e)
```

Cart API

```
from flask import Blueprint,request
import ibm_db
from ..lib import validation_error
from ..lib.auth import check_auth
from ..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=request.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_param(prepare_select,1,product)
   ibm_db.execute(prepare_select)
   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_stmt)

    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['STOCK']-1 or 0)
    ibm_db.bind_param(update_stmt,2,product)
    ibm_db.execute(update_stmt)


    print("sdd")
    return {"message":'Created'},201
  except 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="DELETE FROM CART WHERE 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)
    return {"message":'Deleted'},201
  except 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,price,stock,image,brand
,specificity,CART.user as user FROM CART JOIN PRODUCT ON
CART.PRODUCT=PRODUCT.ID JOIN CATEGORY ON PRODUCT.CATEGORY = CATEGORY.ID
WHERE 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(products)

   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(product,id,user_id)
```

```
    select_sql = "SELECT * FROM PRODUCT WHERE ID=?"
    prepare_select =ibm_db.prepare(db.get_db(),select_sql)

    ibm_db.bind_param(prepare_select,1,product)

    ibm_db.execute(prepare_select)

    is_product = ibm_db.fetch_assoc(prepare_select)


    print(is_product)


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

Order API:

```
from flask import Blueprint,request
import ibm_db
from ..lib import exception
from ..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=request.get_json()
   products=data['products']
   insert_sql="SELECT ORDER_ID FROM FINAL TABLE (INSERT INTO
ORDER(user) VALUES(?))" prep_stmt =
   ibm_db.prepare(db.get_db(), insert_sql)
   ibm_db.bind_param(prep_stmt,1,user_id)
   ibm_db.execute(prep_stmt)
   order = ibm_db.fetch_assoc(prep_stmt)
   print(order)

   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'},201
 except 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 ON
ORDERDETAIL.ORDER=ORDER.ORDER_ID JOIN PRODUCT ON
ORDERDETAIL.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(products)
  return products or [],200

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