auth_bp.py

```
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)




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.py

```
@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,p
rice,stock,image,brand,specificity,CART.user as user FROM CART JOIN PRODUCT ON CART.PRODUCT=PRO
DUCT.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)
```

category_bp.py

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

category_bp = Blueprint("category",__name__)
```

```
@category_bp.route("/get",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
    return jsonify(categories),200
  except Exception as e:
    return exception.handle_exception(e)

@category_bp.route("/create",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)
```

image_bp.py

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


image_bp = Blueprint("image",__name__)

@image_bp.route('/image/<id>',methods=['POST'])
def uploadImage(id):
  try:
    uploaded_file = request.files['file']+datetime.date
    if uploaded_file.filename != '':
        uploaded_file.save(os.path.join('/uploads', uploaded_file.filename))
    insert_sql="UPDATE PRODUCT SET image=? WHERE ID=?"
    prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)
    ibm_db.bind_param(prep_stmt,1,uploaded_file)
    ibm_db.bind_param(prep_stmt,2,id)

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

@image_bp.route('/<filename>')
def upload(filename):
  try:
    return send_from_directory("/uploads", filename)
  except Exception as e:
    return exception.handle_exception(e)
```

order_bp.py

```
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,sto
ck,image,brand,specificity,paid FROM ORDERDETAIL JOIN ORDER ON ORDERDETAIL.ORDER=ORDER.O
RDER_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)
```

product_bp.py

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



product_bp = Blueprint("product",__name__)

@product_bp.route("/create",methods=['POST'])
def add_product():
  try:
     data = request.get_json()
     product_name=data['product_name']
     category=data['category']
     description = data['description']
     stock=data['stock']
     price = data['price']
     insert_sql="INSERT INTO PRODUCT(product_name,category,description,stock,price) VALUES(?,?,?,?,?)"
     prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)
     ibm_db.bind_param(prep_stmt,1,product_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,price)
     ibm_db.execute(prep_stmt)
     return {"message":'Created'},200
  except Exception as e:
    return exception.handle_exception(e)

@product_bp.route("/get",methods=['GET'])
def get_product():
  try:
    # select_sql = "SELECT PRODUCT.ID AS product_id, category,category_name,product_name,description,price,
stock,image FROM PRODUCT JOIN CATEGORY ON CATEGORY.ID=PRODUCT.CATEGORY"
    select_sql = "SELECT * FROM PRODUCT WHERE"
    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 jsonify(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 FROM PRODUCT JOIN CATEGORY ON CATEGORY.ID=PRODUCT.CATEGORY WHERE PRO
DUCT.ID=?"
    select_sql = "SELECT * FROM PRODUCT 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()
     product_name=data['product_name']
     category=data['category']
     description = data['description']
     stock=data['stock']
     price = data['price']
     insert_sql="UPDATE PRODUCT SET product_name=?,category=?,description=?,stock=?,price=? WHERE ID=
?"
     prep_stmt = ibm_db.prepare(db.get_db(), insert_sql)
     ibm_db.bind_param(prep_stmt,1,product_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,price)
     ibm_db.bind_param(prep_stmt,6,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)
```