```
from flask import  render_template,url_for,redirect,flash,request,jsonify
from flaskinventory import app,db
from flaskinventory.forms import addproduct,addlocation,moveproduct,editproduct,editlocation
from flaskinventory.models import Location,Product,Movement,Balance
import time,datetime
from sqlalchemy.exc import IntegrityError


@app.route("/Overview")
@app.route("/")
def overview():
    balance = Balance.query.all()
    exists = bool(Balance.query.all())
    if exists== False :
        flash(f'Add products,locations and make transfers to view','info')
    return render_template('overview.html' ,balance=balance)


@app.route("/Product", methods = ['GET','POST'])
def product():
    form = addproduct()
    eform = editproduct()
    details = Product.query.all()
    exists = bool(Product.query.all())
    if exists== False and request.method == 'GET' :
            flash(f'Add products to view','info')
    elif eform.validate_on_submit() and request.method == 'POST':

        p_id = request.form.get("productid","")
        pname = request.form.get("productname","")
        details = Product.query.all()
```

```
    prod = Product.query.filter_by(prod_id = p_id).first()
    prod.prod_name = eform.editname.data
    prod.prod_qty= eform.editqty.data
    Balance.query.filter_by(product=pname).update(dict(product=eform.editname.data))
    Movement.query.filter_by(pname=pname).update(dict(pname=eform.editname.data))
    try:
        db.session.commit()
        flash(f'Your product  has been updated!', 'success')
        return redirect('/Product')
    except IntegrityError :
        db.session.rollback()
        flash(f'This product already exists','danger')
        return redirect('/Product')
    return render_template('product.html',title = 'Products',details=details,eform=eform)

elif form.validate_on_submit() :
    product = Product(prod_name=form.prodname.data,prod_qty=form.prodqty.data)
    db.session.add(product)
    try:
        db.session.commit()
        flash(f'Your product {form.prodname.data} has been added!', 'success')
        return redirect(url_for('product'))
    except IntegrityError :
        db.session.rollback()
        flash(f'This product already exists','danger')
        return redirect('/Product')
return render_template('product.html',title = 'Products',eform=eform,form = form,details=details)
```

```
@app.route("/Location", methods = ['GET', 'POST'])
def loc():
    form = addlocation()
    lform = editlocation()
    details = Location.query.all()
    exists = bool(Location.query.all())
    if exists== False and request.method == 'GET':
        flash(f'Add locations  to view','info')
    if lform.validate_on_submit() and request.method == 'POST':
        p_id = request.form.get("locid","")
        locname = request.form.get("locname","")
        details = Location.query.all()
        loc = Location.query.filter_by(loc_id = p_id).first()
        loc.loc_name = lform.editlocname.data
        Balance.query.filter_by(location=locname).update(dict(location=lform.editlocname.data))
        Movement.query.filter_by(frm=locname).update(dict(frm=lform.editlocname.data))
        Movement.query.filter_by(to=locname).update(dict(to=lform.editlocname.data))
        try:
            db.session.commit()
            flash(f'Your location  has been updated!', 'success')
            return redirect(url_for('loc'))
        except IntegrityError :
            db.session.rollback()
            flash(f'This location already exists','danger')
            return redirect('/Location')
    elif form.validate_on_submit() :
        loc = Location(loc_name=form.locname.data)
        db.session.add(loc)
        try:
            db.session.commit()
            flash(f'Your location {form.locname.data} has been added!', 'success')
```

```
            return redirect(url_for('loc'))
        except IntegrityError :
            db.session.rollback()
            flash(f'This location already exists','danger')
            return redirect('/Location')
    return render_template('loc.html',title = 'Locations',lform=lform,form = form,details=details)



@app.route("/Transfers", methods = ['GET', 'POST'])
def move():
    form = moveproduct()

    details = Movement.query.all()
    pdetails = Product.query.all()
    exists = bool(Movement.query.all())
    if exists== False and request.method == 'GET' :
            flash(f'Transfer products  to view','info')
    #-------------------------------------------------------------
    prod_choices = Product.query.with_entities(Product.prod_name,Product.prod_name).all()
    loc_choices = Location.query.with_entities(Location.loc_name,Location.loc_name).all()
    prod_list_names = []
    src_list_names,dest_list_names=[('Warehouse','Warehouse')],[('Warehouse','Warehouse')]
    prod_list_names+=prod_choices
    src_list_names+=loc_choices
    dest_list_names+=loc_choices
    #passing list_names to the form for select field
    form.mprodname.choices = prod_list_names
    form.src.choices = src_list_names
    form.destination.choices = dest_list_names
    #-------------------------------------------------------------
```

```
    #send to db
    if form.validate_on_submit() and request.method == 'POST' :


        timestamp = datetime.datetime.now()
        boolbeans =
check(form.src.data,form.destination.data,form.mprodname.data,form.mprodqty.data)
        if boolbeans == False:
            flash(f'Retry with lower quantity than source location', 'danger')
        elif boolbeans == 'same':
            flash(f'Source and destination cannot be the same.', 'danger')
        elif boolbeans == 'no prod':
            flash(f'Not enough products in this loaction .Please add products', 'danger')
        else:
            mov = Movement(ts=timestamp,frm=form.src.data,to = form.destination.data,
                        pname=form.mprodname.data,pqty=form.mprodqty.data)
            db.session.add(mov)
            db.session.commit()
            flash(f'Your  activity has been added!', 'success')
        return redirect(url_for('move'))
    return render_template('move.html',title = 'Transfers',form = form,details= details)


def check(frm,to,name,qty):
    if frm == to :
        a = 'same'
        return a
    elif frm =='Warehouse' and to != 'Warehouse':
        prodq = Product.query.filter_by(prod_name=name).first()
        if prodq.prod_qty >= qty:
            prodq.prod_qty-= qty
            bal = Balance.query.filter_by(location=to,product=name).first()
            a=str(bal)
```

```
            if(a=='None'):
                new = Balance(product=name,location=to,quantity=qty)
                db.session.add(new)
            else:
                bal.quantity += qty
            db.session.commit()
        else :
            return False
    elif to == 'Warehouse' and frm != 'Warehouse':
        bal = Balance.query.filter_by(location=frm,product=name).first()
        a=str(bal)
        if(a=='None'):
            return 'no prod'
        else:
            if bal.quantity >= qty:
                prodq = Product.query.filter_by(prod_name=name).first()
                prodq.prod_qty = prodq.prod_qty + qty
                bal.quantity -= qty
                db.session.commit()
            else :
                 return False


    else: #from='?' and to='?'
        bl = Balance.query.filter_by(location=frm,product=name).first() #check if from location is in
Balance
        a=str(bl)
        if(a=='None'):#if not
            return 'no prod'


        elif (bl.quantity - 100) > qty:
           #if from qty is sufficiently large, check to  in Balance
```

```
            bal = Balance.query.filter_by(location=to,product=name).first()
            a = str(bal)
            if a=='None':
                #if not add entry
                new = Balance(product=name,location=to,quantity=qty)
                db.session.add(new)
                bl = Balance.query.filter_by(location=frm,product=name).first()
                bl.quantity -= qty
                db.session.commit()
            else:#else add to 'from' qty and minus from 'to' qty
                    bal.quantity += qty #if yes,add to to qty
                    bl = Balance.query.filter_by(location=frm,product=name).first()
                    bl.quantity -= qty
                    db.session.commit()
        else  :
                return False
@app.route("/delete")
def delete():
    type = request.args.get('type')
    if type == 'product':
        pid = request.args.get('p_id')
        product = Product.query.filter_by(prod_id=pid).delete()
        db.session.commit()
        flash(f'Your product  has been deleted!', 'success')
        return redirect(url_for('product'))
        return render_template('product.html',title = 'Products')
    else:
        pid = request.args.get('p_id')
        loc = Location.query.filter_by(loc_id = pid).delete()
        db.session.commit()
        flash(f'Your location  has been deleted!', 'success')
```

```
return redirect(url_for('loc'))

return render_template('loc.html',title = 'Locations')
```