INVENTORY MANAGEMENT

LOGIN

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Current File (Integrated Terminal)",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal"
        },
        {
            "name": "Python: Attach",
            "type": "python",
            "request": "attach",
            "port": 5678,
            "host": "localhost"
        },
        {
            "name": "Python: Django",
            "type": "python",
            "request": "launch",
            "program": "${workspaceFolder}/manage.py",
            "console": "integratedTerminal",
            "args": [
                "runserver",
                "--noreload",
                "--nothreading"
            ],
            "django": true
        },
        {
            "name": "Python: Flask",
            "type": "python",
            "request": "launch",
            "module": "flask",
            "env": {
                "FLASK_APP": "app.py"
            },
            "args": [
                "run",
                "--no-debugger",
                "--no-reload"
            ],
            "jinja": true
        },
        {
            "name": "Python: Current File (External Terminal)",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "externalTerminal"
        }
    ]
}
```

ADDING A PRODUCT MOVEMENT

```
from
flask_wtf
import
FlaskForm
from inventorymanagement import app, mysql, db
from wtforms import StringField, PasswordField, SubmitField, BooleanField, IntegerField,
from wtforms.validators import DataRequired, Length, Email, EqualTo, ValidationError
from inventorymanagement.models import User
#######################Users####################################
class RegistrationForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired(), Length(min=2, max=20)
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password', validators=[DataRequired(), Equ
    submit = SubmitField('Sign Up')

    def validate_username(self, username):
        user = User.query.filter_by(username = username.data).first()
        if user:
            raise ValidationError('Username Taken')
    def validate_email(self, email):
        user = User.query.filter_by(email = email.data).first()
        if user:
            raise ValidationError('Email Taken')
class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember = BooleanField('Remember Me')
    submit = SubmitField('Login')
########################Products######################################
class AddProduct(FlaskForm):
    name = StringField('Product Name', validators=[DataRequired()])
    submit = SubmitField('Add Product')
########################Locations#####################################
class AddLocation(FlaskForm):
    name = StringField('Location Name', validators=[DataRequired()])
    submit = SubmitField('Add Location')
```

```
########################ProductMovements#######################################
```

```
class AddProductMovement(FlaskForm):
    conn = mysql.connect()
    cursor = conn.cursor()
    cursor.execute("SELECT location_name FROM location")
    locations = cursor.fetchall()
    values = []


    for location in locations:
        values.append((location[0], location[0]))
    name = StringField('Product Name', validators=[DataRequired()])
    fromLocation = SelectField('From Location', choices=values, validators=[DataRequired
    toLocation = SelectField('To Location', choices=values, validators=[DataRequired()])
    quantity = IntegerField('Product Quantity', validators=[DataRequired()])
    timestamp = DateField('Date', validators=[DataRequired()])
    email = StringField('Email', validators=[DataRequired(), Email()])
class ProductMovement(FlaskForm):
    name = StringField('Product Name', validators=[DataRequired()])
    #timestamp = DateField('Date', validators=[DataRequired()])
    #fromLocation = SelectField('From Location', validators=[DataRequired()])
    #toLocation = SelectField('To Location', validators=[DataRequired()])
    #quantity = SelectField('Quantity', validators=[DataRequired()])
    #email = StringField('Email', validators=[DataRequired(), Email()])
    submit = SubmitField('Move Product')
```

26 inventorymanagement/routes.py

```
@@ -1,7 +1,7 @@
import datetime
from flask import render_template, url_for, flash, redirect, request
from inventorymanagement import app, bcrypt, mysql, db
from inventorymanagement.forms import RegistrationForm, LoginForm, AddProduct, AddLocation, Ad
from inventorymanagement.forms import RegistrationForm, LoginForm, AddProduct, AddLocation, Pr
from inventorymanagement.models import User
from flask_login import login_user, current_user, logout_user, login_required
from wtforms import StringField, PasswordField, SubmitField, BooleanField, IntegerField, DateT
@@ -20,7 +20,7 @@ def home():
    products = cursor.fetchone()
    cursor.execute("SELECT COUNT(location_id) FROM location")
    locations = cursor.fetchone()
    cursor.execute("SELECT COUNT(*) FROM inventorymovement WHERE user_id="+ str(current_user.u
    cursor.execute("SELECT COUNT(*) FROM productmovement WHERE user_id="+ str(current_user.use
    movements = cursor.fetchone()
    sales = 5
    return render_template('home.html', title='Home', products=products[0], locations=locatio
@@ -255,35 +255,35 @@ def view_location():
@app.route("/add_productmovement?<int:product_id>", methods=['GET', 'POST'])
```

```
@login_required
def add_productmovement(product_id):
    form = AddProductMovement()
    print(form)
    form = ProductMovement()


    conn = mysql.connect()
    cursor = conn.cursor()
    time = datetime.date.today()
    form = AddProductMovement()
    cursor.execute("SELECT product_name FROM product WHERE product_id="+ str(product_id) +"")
    product_name = cursor.fetchone()
    cursor.execute("SELECT location_name FROM location")
    locations = cursor.fetchall()
    time = datetime.date.today()
    ranges = len(locations)
    cursor.execute("SELECT * FROM locationinventory WHERE locationinventory_id="+ str(product_
    quantities = cursor.fetchall()
    print(form.validate_on_submit())
    if form.validate_on_submit():
        product_name = form.name.data
        from_location = form.fromLocation.data
        to_location = form.toLocation.data
        quantity = form.quantity.data
        date = form.timestamp.data
        email = form.email.data
        print(product_name,from_location,to_location,quantity)
        from_location = request.values.get('fromLocation')
        to_location = request.values.get('toLocation')
        quantity = request.values.get('quantity')
        date = request.values.get('timestamp')
        email = request.values.get('email')
        print(product_name,from_location,to_location,quantity,date,email)
        flash('Updated!', 'success')
        return redirect(url_for('view_location'))
    return render_template('add_productmovement.html', title='Movement', form=form, time=time
ranges=ranges)
```

```
@app.route("/edit_productmovement")
@login_required
def edit_productmovement():
-    form = AddProductMovement()
+    form = ProductMovement()
    return render_template('edit_productmovement.html', title='Movement', form=form)


@app.route("/view_productmovement")
```

61 inventorymanagement/templates/add_productmovement.html

```
@@ -11,32 +11,50 @@
                <label class="form-control-label">Product Name</label>
                <input name="name" id="name" class="form-control form-control-lg
            </div>
+            <div class="form-group">
+                <label class="form-control-label">Product Quantities at Respecti
+            </div>
            <div class="form-group" style="height:100px;overflow: auto;">
                {% for index in range(ranges) %}
-                <div class="form-group">
-                    <input class="form-control form-control-md" value="Quantity
-                </div>
+                    {% if quantities[0][index] != 0%}
+                        <div class="form-group">
+                            <input class="form-control form-control-md" value="Q
+                        </div>
+                    {% endif %}
                {% endfor %}
            </div>
            <div class="form-group">
-                {{ form.fromLocation.label(class="form-control-label") }}
-                {{ form.fromLocation(class="form-control form-control-lg") }}
+                <label class="form-control-label">From Location</label>
+                <select name="fromLocation" id="fromLocation" class="form-contro
+                {% for index in range(ranges) %}
+                    {% if quantities[0][index] != 0%}
```

```
                                    <option value="{{ locations[index][0] }}" data-qty="{{ q
                                {% endif %}
                            {% endfor %}
                            </select>
                        </div>
                        <div class="form-group">
                            {{ form.toLocation.label(class="form-control-label") }}
                            {{ form.toLocation(class="form-control form-control-lg") }}
                            <label class="form-control-label">To Location</label>
                            <select name="toLocation" id="toLocation" class="form-control fo
                            {% for index in range(ranges) %}
                                <option value="{{ locations[index][0] }}">{{ locations[index
                            {% endfor %}
                            </select>
                        </div>
                        <div class="form-group">
                            {{ form.quantity.label(class="form-control-label") }}
                            {{ form.quantity(class="form-control form-control-lg") }}
                            <label class="form-control-label">Quantity</label>
                            <select name="quantity" id="quantity" class="form-control form-c
                                <div id="remove">
                                </div>
                            </select>
                        </div>
                        <div class="form-group">
                            <label class="form-control-label">Date</label>
                            <input name="timestamp" id="timestamp" class="form-control form-
                            <input name="timestamp" id="timestamp" class="form-control form-
                        </div>
                        <div class="form-group">
                            <label class="form-control-label">Email</label>
                            <input name="email" id="email" class="form-control form-control-
                            <input name="email" id="email" class="form-control form-control-
                        </div>
                    </fieldset>
                    <div class="form-group">
@@ -46,10 +64,21 @@
            </div>
```

```
        </div>
    </div>
    <script>
        function value(){


            console.log(conceptName)
        }
    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
    <script language="JavaScript" type="text/javascript">
        $("#fromLocation").change(function () {
            value = $(this).find(':selected').data('qty');
            select = document.getElementById('quantity');
            remove = document.getElementById('remove');
            append = '';
            empty = '';
            for (i = 1; i < value+1; i++) {
                append = append + '<option id="remove" value="'+ i +'">'+ i +'</option>';
            }
            select.innerHTML += append;
        });
    </script>
    <script type="text/javascript">


    </script>
```

ROUTES AND LOCATION

```
def
add_location()
:
                              form = AddLocation()
                              if form.validate_on_submit():
                                  location = Location(location_name=form.name.data)
                                  db.session.add(location)
                                  db.session.commit()
                                  conn = mysql.connect()
                                  cursor = conn.cursor()
```

```
        cursor.execute("INSERT INTO `location`(`location_name`) VALUES ('"+ form.na
        conn.commit()
        location = (form.name.data).replace(" ","")
        print(location)
        print("ALTER TABLE locationinventory ADD COLUMN "+ form.name.data +" INTEGE
        cursor.execute("ALTER TABLE locationinventory ADD COLUMN "+ (form.name.data
DEFAULT 0")
        conn.commit()
        conn.close()
        print("INSERT INTO `location`(`location_name`) VALUES ('"+ form.name.data +
        print("ALTER TABLE locationinventory ADD COLUMN "+ form.name.data +" INTEGE
        flash('Location Added')
        return redirect(url_for('view_location'))
    return render_template('add_location.html', title='Location', form=form)
@@ -119,25 +128,27 @@ def add_location():
@login_required
def edit_location(location_id):
    form = AddLocation()
    location = Location.query.get(location_id)
    print(location.location_id)
    print(form.name.data)
    conn = mysql.connect()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM location WHERE location_id='"+ str(location_id) +
    conn.commit()
    location = cursor.fetchone()
    if form.validate_on_submit():
        location.location_id = location.location_id
        location.location_name = form.name.data
        print(location.location_id)
        print(location.location_name)
        db.session.commit()
        cursor.execute("UPDATE location SET location_name='"+ form.name.data +"' WH
str(location_id) +"'")
        print(cursor)
        flash('Updated!', 'success')
        return redirect(url_for('view_location'))
    elif request.method == 'GET':
        form.name.data = location.location_name
```

```
            form.name.data = location_id
        return render_template('edit_location.html', title='Location', form=form, locat


    @app.route("/view_location")
    @login_required
    def view_location():
        locations = Location.query.all()
        conn = mysql.connect()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM location")
        locations = cursor.fetchall()
        return render_template('view_location.html', title='Location', locations=locati


    ########################ProductMovements#######################################
```

2 inventorymanagement/templates/edit_location.html

```
@@ -9,7 +9,7 @@
                <fieldset class="form-group">
                    <div class="form-group">
                        <label class="form-control-label" for="name">Edit Location Name<
                        <input class="form-control form-control-lg" id="name" name="name
                        <input class="form-control form-control-lg" id="name" name="name
                    </div>
                </fieldset>
                <div class="form-group">
```

36 inventorymanagement/templates/view_location.html

```
@@ -49,35 +49,27 @@ <h2>Inventory Location</h2>
                    <br>
                </div>
                <div class="col-md-12" style="height:600px; overflow: auto;">
                    {% for location in locations %}
                        <div class="row">
                            <div class="col-md-12 border rounded pt-2" style=" height:50
                                <div class="row">
```

```
                                          <div class="col-md-4 border-left pt-2">
                                              {{ location.location_id }}
                                          </div>
                                          <div class="col-md-4 border-left pt-2">
                                              {{ location.location_name }}
                                          </div>
                                          <div class="col-md-4 border-left pt-2">
                                                  <a href="{{ url_for('edit_location', locatio
                                          </div>
                            {% for location in locations %}
                                <div class="row">
                                    <div class="col-md-12 border rounded pt-2" style=" height:50px;"
                                        <div class="row">
                                            <div class="col-md-4 border-left pt-2">
                                                {{ location[0] }}
                                            </div>
                                            <div class="col-md-4 border-left pt-2">
                                                {{ location[1] }}
                                            </div>
                                            <div class="col-md-4 border-left pt-2">
                                                    <a href="{{ url_for('edit_location', location_id
                                            </div>
                                        </div>
                                    </div>
                                {% endfor %}
                                </div>
                            {% endfor %}
                            </div>
                        </div>
                    </div>
                </div>
                <br>
            </div>
            <script>
                jQuery(document).ready(function($) {
                    $(".clickable-row").click(function() {
                        window.location = $(this).data("href");
                        console.log('A')
                    });
```

```
          });
      </script>
```

```
import
datetime
           from flask import render_template, url_for, flash, redirect, request
           from inventorymanagement import app, bcrypt, mysql, db
           from inventorymanagement.forms import RegistrationForm, LoginForm, AddProduct,
           AddLocation, ProductMovement
           from inventorymanagement.models import User
           from flask_login import login_user, current_user, logout_user, login_required
           from wtforms import StringField, PasswordField, SubmitField, BooleanField,
           IntegerField, DateTimeField, SelectField, Label
           from wtforms.validators import DataRequired, Length, Email, EqualTo,
           ValidationError
           @app.route("/")
           @app.route("/anon")
           def anon():
               return redirect(url_for('login'))
           @app.route("/home")
           def home():
               conn = mysql.connect()
               cursor = conn.cursor()
               cursor.execute("SELECT COUNT(product_id) FROM product WHERE user_id="+
           str(current_user.user_id) +"")
               products = cursor.fetchone()
               cursor.execute("SELECT COUNT(location_id) FROM location")
               locations = cursor.fetchone()
               cursor.execute("SELECT COUNT(*) FROM productmovement WHERE user_id="+
           str(current_user.user_id) +"")
               movements = cursor.fetchone()
               sales = 5
               return render_template('home.html', title='Home', products=products[0],
           locations=locations[0], sales=sales, movements=movements[0])
           @app.route("/about")
           def about():
               return render_template('about.html', title="About")
           ########################Users#######################################
           @app.route("/register", methods=['GET', 'POST'])
           def register():
               if current_user.is_authenticated:
```

```
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password =
bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data,
password=hashed_password)
        db.session.add(user)
        db.session.commit()
        flash('Your Account has been created', 'success')
        return redirect(url_for('login'))
    return render_template('register.html', title='Register', form=form)
@app.route("/login", methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            login_user(user, remember=form.remember.data)
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('home'))
        else:
            flash('Login Unsuccessful', 'danger')
    return render_template('login.html', title='Login', form=form)
@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for('anon'))
########################Products#######################################
@app.route("/add_product", methods=['GET', 'POST'])
@login_required
def add_product():
    conn = mysql.connect()
    cursor = conn.cursor()
    cursor.execute("SELECT location_name FROM location")
    locations = cursor.fetchall()
    places = []
    for location in locations:
        places.append(location[0])

    form = AddProduct()
    if form.validate_on_submit():
        name = form.name.data
        input_values = request.form.getlist('places[]')
        totalquantity = 0

        locationinventory = "INSERT INTO `locationinventory`("

        for count,place in enumerate(places):
            locationinventory = locationinventory + "`"+ place +"`"
            if count != len(places)-1:
                locationinventory = locationinventory + ","
        locationinventory = locationinventory + ",`user_id`) VALUES ("
        for count,input_value in enumerate(input_values):
            totalquantity = totalquantity + int(input_value)
            locationinventory = locationinventory + "'"+ input_value +"'"
            if count != len(input_values)-1:
                locationinventory = locationinventory + ","

        locationinventory = locationinventory + ",'"+ str(current_user.user_id)
+"')"
        location = "INSERT INTO
`product`(`product_name`,`product_quantity`,`user_id`) VALUES ('"+ name +"','"+
str(totalquantity) +"','"+ str(current_user.user_id) +"')"
        print(locationinventory)
        cursor.execute(locationinventory)
        conn.commit()
        cursor.execute(location)
        conn.commit()
        conn.close()
        flash('Done', 'success')
        return redirect(url_for('view_product'))
    return render_template('add_product.html', title='Product', form=form,
locations=locations)
@app.route("/edit_product?<int:product_id>", methods=['GET', 'POST'])
def edit_product(product_id):
    form = AddProduct()
    conn = mysql.connect()
    cursor = conn.cursor()
    values = "Select * from product WHERE product_id="+ str(product_id) +""
    values = cursor.execute(values)
    values = cursor.fetchone()
    locations = "SELECT location_name FROM location"
    locations = cursor.execute(locations)
    locations = cursor.fetchall()
    places = []
    for location in locations:
        places.append(location[0])
    inventory = "Select * from locationinventory WHERE locationinventory_id="+
str(product_id) +""
    inventory = cursor.execute(inventory)
    inventory = cursor.fetchone()
    ranges = len(locations)
    quantities = []
    for inventory in inventory:
        quantities.append(inventory)
    print(quantities)
    for count in range(2):
        quantities.pop(0)
    print(quantities)
    if form.validate_on_submit():
        name = form.name.data
        input_values = request.form.getlist('places[]')
        print(input_values)
        totalquantity = 0
        locationinventory = "UPDATE `locationinventory` SET "

        for index in range(ranges):
            locationinventory = locationinventory + "`" + locations[index][0] +
"`='" + str(input_values[index]) +"'"
            totalquantity = totalquantity + int(input_values[index])
            if index != len(input_values)-1:
                locationinventory = locationinventory + ","
        locationinventory = locationinventory + " WHERE `locationinventory_id`=" +
str(product_id)
        location = "UPDATE `product` SET `product_name`='"+ name
+"',`product_quantity`='"+ str(totalquantity) +"' WHERE product_id="+
str(product_id)
        print(locationinventory)
        cursor.execute(locationinventory)
        conn.commit()
        print(location)
        cursor.execute(location)
        conn.commit()
        conn.close()
        flash('Done', 'success')
        return redirect(url_for('view_product'))
    return render_template('edit_product.html', title='Product', form=form,
values=values, locations=locations, quantities=quantities, ranges=ranges)
@app.route("/product_info?<int:product_id>", methods=['GET', 'POST'])
def product_info(product_id):
    form = AddProduct()
    conn = mysql.connect()
    cursor = conn.cursor()
    values = "Select * from product WHERE product_id="+ str(product_id) +""
    values = cursor.execute(values)
    values = cursor.fetchone()
    locations = "SELECT location_name FROM location"
    locations = cursor.execute(locations)
    locations = cursor.fetchall()
    places = []
    for location in locations:
        places.append(location[0])
    inventory = "Select * from locationinventory WHERE locationinventory_id="+
str(product_id) +""
    inventory = cursor.execute(inventory)
    inventory = cursor.fetchone()
    ranges = len(locations)
    quantities = []
    for inventory in inventory:
        quantities.append(inventory)
    for count in range(2):
        quantities.pop(0)
    return render_template('product_info.html', title='Product', form=form,
values=values, locations=locations, quantities=quantities, ranges=ranges)
@app.route("/view_product")
@login_required
def view_product():
    conn = mysql.connect()
    cursor = conn.cursor()
    products = cursor.execute("SELECT * FROM product WHERE user_id='"+
str(current_user.user_id)+"'")
    products = cursor.fetchall()
    inventory_places = cursor.execute("SELECT * FROM locationinventory")
    inventory_places = cursor.fetchall()
    return render_template('view_product.html', title='Product', products=products,
inventory_places=inventory_places)
########################Locations#######################################
@app.route("/add_location", methods=['GET', 'POST'])
@login_required
```

```
def add_location():
    form = AddLocation()
    if form.validate_on_submit():
        conn = mysql.connect()
        cursor = conn.cursor()
        count = cursor.execute("SELECT location_name FROM location WHERE
location_name='"+ (form.name.data).replace(" ", "_") +"'")
        if count == 0:
            cursor.execute("INSERT INTO `location`(`location_name`) VALUES ('"+
(form.name.data).replace(" ", "_") +"')")
            conn.commit()
            cursor.execute("ALTER TABLE locationinventory ADD COLUMN "+
(form.name.data).replace(" ", "_") +" INTEGER DEFAULT 0")
            conn.commit()
            conn.close()
            flash('Location Added', 'success')
            return redirect(url_for('view_location'))
        else:
            conn.close()
            flash('Location Exixts', 'danger')
            return redirect(url_for('add_location'))

    return render_template('add_location.html', title='Location', form=form)
@app.route("/edit_location?<int:location_id>", methods=['GET', 'POST'])
@login_required
def edit_location(location_id):
    form = AddLocation()
    conn = mysql.connect()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM location WHERE location_id='"+ str(location_id)
+"'")
    location = cursor.fetchone()
    if form.validate_on_submit():
        cursor.execute("UPDATE location SET location_name='"+ form.name.data +"'
WHERE location_id='"+ str(location_id) +"'")
        conn.commit()
        flash('Updated!', 'success')
        return redirect(url_for('view_location'))
    elif request.method == 'GET':
        form.name.data = location_id
    return render_template('edit_location.html', title='Location', form=form,
location=location)
@app.route("/view_location")
```

```
@login_required
def view_location():
    conn = mysql.connect()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM location")
    locations = cursor.fetchall()
    return render_template('view_location.html', title='Location',
locations=locations)
########################ProductMovements#######################################
@app.route("/add_productmovement?<int:product_id>", methods=['GET', 'POST'])
@login_required
def add_productmovement(product_id):
    form = ProductMovement()
    conn = mysql.connect()
    cursor = conn.cursor()
    cursor.execute("SELECT product_name FROM product WHERE product_id="+
str(product_id) +"")
    product_name = cursor.fetchone()
    cursor.execute("SELECT location_name FROM location")
    locations = cursor.fetchall()
    time = datetime.date.today()
    ranges = len(locations)
    cursor.execute("SELECT * FROM locationinventory WHERE locationinventory_id="+
str(product_id) +"")
    inventory = cursor.fetchone()
    quantities = []
    for inventory in inventory:
        quantities.append(inventory)
    for count in range(2):
        quantities.pop(0)
    print(quantities[5])
    if form.validate_on_submit():
        product_name = form.name.data
        from_location = request.values.get('fromLocation')
        to_location = request.values.get('toLocation')
        quantity = request.values.get('quantity')
        date = request.values.get('timestamp')
        email = request.values.get('email')
        query = "SELECT "+ str(from_location) +","+ str(to_location) +" FROM
locationinventory WHERE locationinventory_id="+ str(product_id) +""
        cursor.execute(query)
        value = cursor.fetchone()
        from_location_qty = value[0] - int(quantity)
        to_location_qty = value[1] + int(quantity)
        query = "UPDATE locationinventory SET "+ str(from_location) +"='"+ 
str(from_location_qty) +"', "+ str(to_location) +"='"+ str(to_location_qty) +"' 
WHERE locationinventory_id="+ str(product_id) +""
        print(query)
        cursor.execute(query)
        conn.commit()
        query = "INSERT INTO `productmovement`(`product_id`, `product_name`, 
`from_location_name`, `to_location_name`, `product_quantity`, `timestamp`, 
`user_id`) VALUES ('"+ str(product_id) +"','"+ form.name.data +"','"+ 
str(from_location) +"','"+ str(to_location) +"','"+ quantity +"','"+ date +"','"+ 
str(current_user.user_id) +"')"
        print(query)
        cursor.execute(query)
        conn.commit()
        conn.close()
        flash('Updated!', 'success')
        return redirect(url_for('view_location'))
    return render_template('add_productmovement.html', title='Movement', form=form, 
time=time, email=current_user.email, product_name=product_name[0], 
locations=locations, quantities=quantities, ranges=ranges)
@app.route("/edit_productmovement?<int:productmovement_id>")
@login_required
def edit_productmovement(productmovement_id):
    conn = mysql.connect()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM productmovement WHERE productmovement_id="+ 
str(productmovement_id) +"")
    query = cursor.fetchone()
    product_id = query[1]
    from_location = query[3]
    to_location = query[4]
    quantity = query[5]
    cursor.execute("SELECT "+ str(from_location) +","+ str(to_location) +" FROM 
locationinventory WHERE locationinventory_id="+ str(product_id) +"")
    inventory = cursor.fetchone()
    from_location_qty = inventory[0] + quantity
    to_location_qty = inventory[1] - quantity
    cursor.execute("UPDATE locationinventory SET "+ str(from_location) +"='"+ 
str(from_location_qty) +"', "+ str(to_location) +"='"+ str(to_location_qty) +"' 
WHERE locationinventory_id="+ str(product_id) +"")
    conn.commit()
    cursor.execute("DELETE FROM productmovement WHERE productmovement_id="+
str(productmovement_id) +"")
    conn.commit()
    return redirect(url_for('view_productmovement'))
    return render_template('', title='Movement', form=form)
@app.route("/view_productmovement")
@login_required
def view_productmovement():
    conn = mysql.connect()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM productmovement WHERE user_id="+
str(current_user.user_id) +"")
    movements = cursor.fetchall()
    counts = len(movements)
    return render_template('view_productmovement.html', title='Movement',
movements=movements, counts=count
```

SQL

```
--
phpMyAdmin
SQL Dump
             -- version 4.6.6deb5
             -- https://www.phpmyadmin.net/
             --
             -- Host: localhost:3306
             -- Generation Time: Sep 13, 2018 at 01:11 PM
             -- Server version: 5.7.23-0ubuntu0.18.04.1
             -- PHP Version: 7.2.7-0ubuntu0.18.04.2
             SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
             SET time_zone = "+00:00";
             /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
             /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
             /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
             /*!40101 SET NAMES utf8mb4 */;
             --
             -- Database: `inventory_management`
             --
             -- --------------------------------------------------------
             --
             -- Table structure for table `location`
             --
             CREATE TABLE `location` (
               `location_id` int(11) NOT NULL,
               `location_name` varchar(60) NOT NULL
             ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
             --
             -- Dumping data for table `location`
             --
             INSERT INTO `location` (`location_id`, `location_name`) VALUES
             (1, 'Mumbai'),
             (2, 'Delhi'),
             (3, 'New_Delhi'),
             (4, 'Chennai'),
             (5, 'Bangalore'),
             (6, 'Hyderabad'),
             (7, 'Kolkata'),
             (8, 'Ahmedabad'),
             (11, 'Mirzapur'),
             (12, 'Nagpur'),
             (13, 'Pune'),
             (19, 'Navi_Mumbai'),
             (21, 'Kundapura'),
             (22, 'Kozhikode');
             -- --------------------------------------------------------
             --
             -- Table structure for table `locationinventory`
             --
             CREATE TABLE `locationinventory` (
               `locationinventory_id` int(11) NOT NULL,
               `user_id` int(11) NOT NULL,
               `Mumbai` int(11) DEFAULT '0',
               `Delhi` int(11) DEFAULT '0',
               `New_Delhi` int(11) DEFAULT '0',
               `Chennai` int(11) DEFAULT '0',
               `Bangalore` int(11) DEFAULT '0',
               `Hyderabad` int(11) DEFAULT '0',
               `Kolkata` int(11) DEFAULT '0',
               `Ahmedabad` int(11) DEFAULT '0',
               `Mirzapur` int(11) DEFAULT '0',
               `Nagpur` int(11) DEFAULT '0',
               `Pune` int(11) DEFAULT '0',
               `Navi_Mumbai` int(11) DEFAULT '0',
               `Kundapura` int(11) DEFAULT '0',
               `Kozhikode` int(11) DEFAULT '0'
             ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
             --
```

```sql
-- Dumping data for table `locationinventory`
--
INSERT INTO `locationinventory` (`locationinventory_id`, `user_id`, `Mumbai`,
`Delhi`, `New_Delhi`, `Chennai`, `Bangalore`, `Hyderabad`, `Kolkata`,
`Ahmedabad`, `Mirzapur`, `Nagpur`, `Pune`, `Navi_Mumbai`, `Kundapura`,
`Kozhikode`) VALUES
(1, 1, 0, 0, 65, 78, 1465, 0, 152, 0, 0, 0, 0, 0, 0, 0),
(2, 1, 37, 6, 489, 1, 6, 5, 0, 5, 0, 0, 5, 0, 0, 15),
(3, 1, 5, 3, 1, 3, 3, 3, 3, 3, 52, 6, 489, 1, 6, 5),
(4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(5, 1, 15, 30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(6, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(7, 1, 48, 52, 6, 475, 1, 6, 5, 0, 5, 0, 0, 5, 0, 0),
(8, 1, 46, 46, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(9, 1, 13, 12, 12, 12, 12, 12, 12, 12, 0, 0, 0, 0, 0, 0),
(10, 1, 20, 20, 20, 20, 20, 20, 20, 20, 0, 0, 0, 0, 0, 0),
(11, 2, 12, 12, 12, 12, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(12, 2, 12, 12, 12, 12, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(13, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(14, 2, 212, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(15, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(16, 2, 456, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(17, 2, 456, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
-- --------------------------------------------------------
--
-- Table structure for table `product`
--
CREATE TABLE `product` (
  `product_id` int(11) NOT NULL,
  `product_name` varchar(60) NOT NULL,
  `product_quantity` int(11) NOT NULL,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `product`
--
INSERT INTO `product` (`product_id`, `product_name`, `product_quantity`,
`user_id`) VALUES
(1, 'Product 1', 1760, 1),
(2, 'Product 2', 569, 1),
(3, 'Product 3', 583, 1),
(4, 'Product 4', 0, 1),
(5, 'Product 5', 45, 1),
(6, 'Product 6', 0, 1),
(7, 'Product 7', 597, 1),
(8, 'Product 8', 92, 1),
(9, 'Product 9', 97, 1),
(10, 'Product 10', 160, 1),
(11, 'Product 11', 60, 2),
(12, 'Product 12', 60, 2),
(13, 'Product 13', 0, 2),
(14, 'Product 14', 224, 2),
(15, 'Product 15', 2, 2),
(16, 'Product 16', 456, 2),
(17, 'Product 17', 461, 2);
-- --------------------------------------------------------
--
-- Table structure for table `productmovement`
--
CREATE TABLE `productmovement` (
  `productmovement_id` int(11) NOT NULL,
  `product_id` int(11) NOT NULL,
  `product_name` varchar(60) NOT NULL,
  `from_location_name` varchar(60) NOT NULL,
  `to_location_name` varchar(60) NOT NULL,
  `product_quantity` int(11) NOT NULL,
  `timestamp` date NOT NULL,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `productmovement`
--
INSERT INTO `productmovement` (`productmovement_id`, `product_id`,
`product_name`, `from_location_name`, `to_location_name`, `product_quantity`,
`timestamp`, `user_id`) VALUES
(1, 3, 'Product 3', 'New_Delhi', 'Mumbai', 2, '2018-09-13', 1),
(3, 7, 'Product 7', 'Chennai', 'Mumbai', 14, '2018-09-13', 1),
(5, 5, 'Product 5', 'Delhi', 'Mumbai', 15, '2018-09-13', 1),
(7, 2, 'Product 2', 'Mumbai', 'Kozhikode', 15, '2018-09-13', 1);
-- --------------------------------------------------------
--
-- Table structure for table `user`
--
CREATE TABLE `user` (
  `user_id` int(11) NOT NULL,
  `user_name` varchar(20) NOT NULL,
  `user_email` varchar(60) NOT NULL,
  `user_password` varchar(60) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `user`
--
INSERT INTO `user` (`user_id`, `user_name`, `user_email`, `user_password`) VALUES
(1, 'himanshu', 'himanshuwarekar@yahoo.com',
'$2b$12$IjFgSnAJCdxQrksUSbTq9Oox7YzBx4Q0WdcRM3FJaLP5CAJ4H9O8i');
--
-- Indexes for dumped tables
--
--
-- Indexes for table `location`
--
ALTER TABLE `location`
  ADD PRIMARY KEY (`location_id`);
--
-- Indexes for table `locationinventory`
--
ALTER TABLE `locationinventory`
  ADD PRIMARY KEY (`locationinventory_id`);
--
-- Indexes for table `product`
--
ALTER TABLE `product`
  ADD PRIMARY KEY (`product_id`);
--
-- Indexes for table `productmovement`
--
ALTER TABLE `productmovement`
  ADD PRIMARY KEY (`productmovement_id`);
--
-- Indexes for table `user`
--
ALTER TABLE `user`
  ADD PRIMARY KEY (`user_id`);
--
-- AUTO_INCREMENT for dumped tables
--
--
-- AUTO_INCREMENT for table `location`
--
```

```sql
ALTER TABLE `location`
  MODIFY `location_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=25;
--
-- AUTO_INCREMENT for table `locationinventory`
--
ALTER TABLE `locationinventory`
  MODIFY `locationinventory_id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=18;
--
-- AUTO_INCREMENT for table `product`
--
ALTER TABLE `product`
  MODIFY `product_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=18;
--
-- AUTO_INCREMENT for table `productmovement`
--
ALTER TABLE `productmovement`
  MODIFY `productmovement_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;
--
-- AUTO_INCREMENT for table `user`
--
ALTER TABLE `user`
  MODIFY `user_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

RUN

```python
from
inventorymanagement
import app
                      if __name__=='__main__':
                          app.run(debug=True)
```