|  |  |
| --- | --- |
| **Date** | 11 November 2022 |
| **Team ID** | PNT2022TMID05082 |
| **Project Name** | Inventory Management System |
| **Batch number** | B11-5A1E |

# PROJECT DEVELOPMENT PHASE - SPRINT 4

**ManageSales.html**

<html>

<head>

<meta charset="utf-8">

<title>MyFlaskApp</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">

</head>

<body>

{% include 'includes/\_navbar.html' %}

<div class="container mt-4">

{% include 'includes/\_messages.html' %}

{% block body %}{% endblock %}

</div>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"></script>

</body>

</html>

# Addsales.html

<html>

<head>

<meta charset="utf-8">

<title>MyFlaskApp</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">

</head>

<body>

{% include 'includes/\_navbar.html' %}

<div class="container mt-4">

{% include 'includes/\_messages.html' %}

{% block body %}{% endblock %}

</div>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"></script>

</body>

</html>

# edit\_product.html

{% extends 'layout.html' %}

{% block body %}

<h1>Edit Product</h1>

{% from "includes/\_formhelpers.html" import render\_field %}

<form action="" method="POST">

<div class="form-group">

{{ render\_field(form.product\_id, class\_="form-control") }}

</div>

<div class="form-group">

{{ render\_field(form.product\_cost, class\_="form-control") }}

</div>

<div class="form-group">

{{ render\_field(form.product\_num, class\_="form-control") }}

</div>

<p><input type="submit" value="Update" class="btn btn-primary"></p>

</form>

{% endblock %}

# product\_movement.html

{% extends 'layout.html' %}

{% block body %}

<h1>Product Movements</h1>

<a class="btn btn-success" href="/add\_product\_movements">Add Product Movements</a>

<hr>

<table class="table table-striped">

<thead>

<tr>

<th>Movement ID</th>

<th>Time</th>

<th>From Location</th>

<th>To Location</th>

<th>Product ID</th>

<th>Quantity</th>

</tr>

</thead>

<tbody>

{% for movement in movements %}

<tr>

<td>{{movement.MOVEMENT\_ID}}</td>

<td>{{movement.TIME}}</td>

<td>{{movement.FROM\_LOCATION}}</td>

<td>{{movement.TO\_LOCATION}}</td>

<td>{{movement.PRODUCT\_ID}}</td>

<td>{{movement.QTY}}</td>

<!--<td><a href="edit\_product\_movement/{{movement.MOVEMENT\_ID}}" class="btn btn-primary pull-right">Edit</a></td>-->

<td>

<form action="{{url\_for('delete\_product\_movements', id=movement.MOVEMENT\_ID)}}" method="POST">

<input type="hidden" name="method" value="DELETE">

<input type="submit" value="Delete" class="btn btn-danger">

</form>

</td>

</tr>

{% endfor %}

</tbody>

</table>

{% endblock %}

# app.py

from flask import Flask, render\_template, flash, redirect, url\_for, session, request, logging from flask\_mysqldb import MySQL

from wtforms import Form, StringField, TextAreaField, PasswordField, validators,

SelectField, IntegerField import ibm\_db

from passlib.hash import sha256\_crypt from functools import wraps

import win32api

from sendgrid import \* #creating an app instance app = Flask( name )

conn=ibm\_db.connect("DATABASE=bludb;HOSTNAME=;PORT=;SECURITY=SSL;SSL ServerCertificate=DigiCertGlobalRootCA.crt;UID=;PWD=;",'','')

#Index @app.route('/') def index():

return render\_template('home.html') #Products

@app.route('/products') def products():

sql = "SELECT \* FROM products" stmt = ibm\_db.prepare(conn, sql) result=ibm\_db.execute(stmt) products=[]

row = ibm\_db.fetch\_assoc(stmt) while(row):

products.append(row)

row = ibm\_db.fetch\_assoc(stmt) products=tuple(products) #print(products)

if result>0:

return render\_template('products.html', products = products) else:

msg='No products found'

return render\_template('products.html', msg=msg) #Locations

@app.route('/locations') def locations():

sql = "SELECT \* FROM locations" stmt = ibm\_db.prepare(conn, sql) result=ibm\_db.execute(stmt) locations=[]

row = ibm\_db.fetch\_assoc(stmt) while(row):

locations.append(row)

row = ibm\_db.fetch\_assoc(stmt) locations=tuple(locations)

#print(locations) if result>0:

return render\_template('locations.html', locations = locations) else:

msg='No locations found'

return render\_template('locations.html', msg=msg) #Product Movements @app.route('/product\_movements')

def product\_movements():

sql = "SELECT \* FROM productmovements" stmt = ibm\_db.prepare(conn, sql) result=ibm\_db.execute(stmt)

movements=[]

row = ibm\_db.fetch\_assoc(stmt) while(row):

movements.append(row)

row = ibm\_db.fetch\_assoc(stmt) movements=tuple(movements) #print(movements)

if result>0:

return render\_template('product\_movements.html', movements = movements) else:

msg='No product movements found'

return render\_template('product\_movements.html', msg=msg)

#Register Form Class class RegisterForm(Form):

name = StringField('Name', [validators.Length(min=1, max=50)]) username = StringField('Username', [validators.Length(min=1, max=25)]) email = StringField('Email', [validators.length(min=6, max=50)]) password = PasswordField('Password', [

validators.DataRequired(),

validators.EqualTo('confirm', message='Passwords do not match')

])

confirm = PasswordField('Confirm Password') #user register

@app.route('/register', methods=['GET','POST']) def register():

form = RegisterForm(request.form)

if request.method == 'POST' and form.validate(): name = form.name.data

email = form.email.data username = form.username.data

password = sha256\_crypt.encrypt(str(form.password.data))

sql1="INSERT INTO users(name, email, username, password) VALUES(?,?,?,?)" stmt1 = ibm\_db.prepare(conn, sql1)

ibm\_db.bind\_param(stmt1,1,name) ibm\_db.bind\_param(stmt1,2,email) ibm\_db.bind\_param(stmt1,3,username) ibm\_db.bind\_param(stmt1,4,password) ibm\_db.execute(stmt1)

#for flash messages taking parameter and the category of message to be flashed flash("You are now registered and can log in", "success")

#when registration is successful redirect to home

return redirect(url\_for('login'))

return render\_template('register.html', form = form) #User login

@app.route('/login', methods = ['GET', 'POST']) def login():

if request.method == 'POST': #Get form fields

username = request.form['username'] password\_candidate = request.form['password']

sql1="Select \* from users where username = ?" stmt1 = ibm\_db.prepare(conn, sql1) ibm\_db.bind\_param(stmt1,1,username) result=ibm\_db.execute(stmt1) d=ibm\_db.fetch\_assoc(stmt1)

if result > 0:

#Get the stored hash data = d

password = data['PASSWORD'] #compare passwords

if sha256\_crypt.verify(password\_candidate, password): #Passed

session['logged\_in'] = True session['username'] = username

flash("you are now logged in","success") return redirect(url\_for('dashboard'))

else:

error = 'Invalid Login'

return render\_template('login.html', error=error) #Close connection

cur.close() else:

error = 'Username not found'

return render\_template('login.html', error=error) return render\_template('login.html')

#check if user logged in def is\_logged\_in(f):

@wraps(f)

def wrap(\*args, \*\*kwargs): if 'logged\_in' in session:

return f(\*args, \*\*kwargs) else:

flash('Unauthorized, Please login','danger') return redirect(url\_for('login'))

return wrap #Logout @app.route('/logout') @is\_logged\_in

def logout(): session.clear()

flash("You are now logged out", "success") return redirect(url\_for('login'))

#Dashboard @app.route('/dashboard') @is\_logged\_in

def dashboard():

sql2="SELECT product\_id, location\_id, qty FROM product\_balance" sql3="SELECT location\_id FROM locations"

stmt2 = ibm\_db.prepare(conn, sql2)

stmt3 = ibm\_db.prepare(conn, sql3) result=ibm\_db.execute(stmt2) ibm\_db.execute(stmt3)

products=[]

row = ibm\_db.fetch\_assoc(stmt2) while(row):

products.append(row)

row = ibm\_db.fetch\_assoc(stmt2) products=tuple(products) locations=[]

row2 = ibm\_db.fetch\_assoc(stmt3) while(row2):

locations.append(row2)

row2 = ibm\_db.fetch\_assoc(stmt3) locations=tuple(locations)

locs = []

for i in locations: locs.append(list(i.values())[0])

if result>0:

return render\_template('dashboard.html', products = products, locations = locs) else:

msg='No products found'

return render\_template('dashboard.html', msg=msg) #Product Form Class

class ProductForm(Form):

product\_id = StringField('Product ID', [validators.Length(min=1, max=200)]) product\_cost = StringField('Product Cost', [validators.Length(min=1, max=200)]) product\_num = StringField('Product Num', [validators.Length(min=1, max=200)])

#Add Product

@app.route('/add\_product', methods=['GET', 'POST']) @is\_logged\_in

def add\_product():

form = ProductForm(request.form)

if request.method == 'POST' and form.validate(): product\_id = form.product\_id.data product\_cost = form.product\_cost.data product\_num = form.product\_num.data

sql1="INSERT INTO products(product\_id, product\_cost, product\_num) VALUES(?,?,?)" stmt1 = ibm\_db.prepare(conn, sql1)

ibm\_db.bind\_param(stmt1,1,product\_id) ibm\_db.bind\_param(stmt1,2,product\_cost) ibm\_db.bind\_param(stmt1,3,product\_num)

ibm\_db.execute(stmt1) flash("Product Added", "success") return redirect(url\_for('products'))

return render\_template('add\_product.html', form=form) #Edit Product

@app.route('/edit\_product/<string:id>', methods=['GET', 'POST']) @is\_logged\_in

def edit\_product(id):

sql1="Select \* from products where product\_id = ?" stmt1 = ibm\_db.prepare(conn, sql1) ibm\_db.bind\_param(stmt1,1,id) result=ibm\_db.execute(stmt1) product=ibm\_db.fetch\_assoc(stmt1)

print(product) #Get form

form = ProductForm(request.form)

#populate product form fields form.product\_id.data = product['PRODUCT\_ID']

form.product\_cost.data = str(product['PRODUCT\_COST']) form.product\_num.data = str(product['PRODUCT\_NUM']) if request.method == 'POST' and form.validate():

product\_id = request.form['product\_id'] product\_cost = request.form['product\_cost'] product\_num = request.form['product\_num']

sql2="UPDATE products SET product\_id=?,product\_cost=?,product\_num=? WHERE product\_id=?"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,product\_id) ibm\_db.bind\_param(stmt2,2,product\_cost) ibm\_db.bind\_param(stmt2,3,product\_num) ibm\_db.bind\_param(stmt2,4,id) ibm\_db.execute(stmt2)

flash("Product Updated", "success") return redirect(url\_for('products'))

return render\_template('edit\_product.html', form=form) #Delete Product

@app.route('/delete\_product/<string:id>', methods=['POST']) @is\_logged\_in

def delete\_product(id):

sql2="DELETE FROM products WHERE product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,id)

ibm\_db.execute(stmt2) flash("Product Deleted", "success") return redirect(url\_for('products'))

#Location Form Class class LocationForm(Form):

location\_id = StringField('Location ID', [validators.Length(min=1, max=200)]) #Add Location

@app.route('/add\_location', methods=['GET', 'POST']) @is\_logged\_in

def add\_location():

form = LocationForm(request.form)

if request.method == 'POST' and form.validate(): location\_id = form.location\_id.data sql2="INSERT into locations VALUES(?)" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,location\_id) ibm\_db.execute(stmt2)

flash("Location Added", "success") return redirect(url\_for('locations'))

return render\_template('add\_location.html', form=form) #Edit Location

@app.route('/edit\_location/<string:id>', methods=['GET', 'POST']) @is\_logged\_in

def edit\_location(id):

sql2="SELECT \* FROM locations where location\_id = ?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,id) result=ibm\_db.execute(stmt2) location=ibm\_db.fetch\_assoc(stmt2)

#Get form

form = LocationForm(request.form) print(location)

#populate article form fields

form.location\_id.data = location['LOCATION\_ID'] if request.method == 'POST' and form.validate():

location\_id = request.form['location\_id']

sql2="UPDATE locations SET location\_id=? WHERE location\_id=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,location\_id) ibm\_db.bind\_param(stmt2,2,id)

ibm\_db.execute(stmt2) flash("Location Updated", "success") return redirect(url\_for('locations'))

return render\_template('edit\_location.html', form=form) #Delete Location

@app.route('/delete\_location/<string:id>', methods=['POST']) @is\_logged\_in

def delete\_location(id):

sql2="DELETE FROM locations WHERE location\_id=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,id)

ibm\_db.execute(stmt2) flash("Location Deleted", "success") return redirect(url\_for('locations'))

#Product Movement Form Class class ProductMovementForm(Form):

from\_location = SelectField('From Location', choices=[]) to\_location = SelectField('To Location', choices=[]) product\_id = SelectField('Product ID', choices=[])

qty = IntegerField('Quantity') class CustomError(Exception):

pass

#Add Product Movement

@app.route('/add\_product\_movements', methods=['GET', 'POST']) @is\_logged\_in

def add\_product\_movements():

form = ProductMovementForm(request.form) sql2="SELECT product\_id FROM products" sql3="SELECT location\_id FROM locations" stmt2 = ibm\_db.prepare(conn, sql2)

stmt3 = ibm\_db.prepare(conn, sql3) result=ibm\_db.execute(stmt2) ibm\_db.execute(stmt3)

products=[]

row = ibm\_db.fetch\_assoc(stmt2) while(row): products.append(row)

row = ibm\_db.fetch\_assoc(stmt2) products=tuple(products)

locations=[]

row2 = ibm\_db.fetch\_assoc(stmt3) while(row2):

locations.append(row2)

row2 = ibm\_db.fetch\_assoc(stmt3) locations=tuple(locations)

prods = []

for p in products: prods.append(list(p.values())[0]) locs = []

for i in locations:

locs.append(list(i.values())[0]) form.from\_location.choices = [(l,l) for l in locs]

form.from\_location.choices.append(("Main Inventory","Main Inventory")) form.to\_location.choices = [(l,l) for l in locs] form.to\_location.choices.append(("Main Inventory","Main Inventory")) form.product\_id.choices = [(p,p) for p in prods]

if request.method == 'POST' and form.validate(): from\_location = form.from\_location.data to\_location = form.to\_location.data product\_id = form.product\_id.data

qty = form.qty.data

if from\_location==to\_location:

raise CustomError("Please Give different From and To Locations!!") elif from\_location=="Main Inventory":

sql2="SELECT \* from product\_balance where location\_id=? and product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,to\_location) ibm\_db.bind\_param(stmt2,2,product\_id) result=ibm\_db.execute(stmt2) result=ibm\_db.fetch\_assoc(stmt2) print(" ")

print(result)

print(" ") app.logger.info(result) if result!=False:

if(len(result))>0:

Quantity = result["QTY"] q = Quantity + qty

sql2="UPDATE product\_balance set qty=? where location\_id=? and product\_id=?"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,q) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.execute(stmt2)

sql2="INSERT into productmovements(from\_location, to\_location, product\_id, qty) VALUES(?, ?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.bind\_param(stmt2,4,qty) ibm\_db.execute(stmt2)

else:

sql2="INSERT into product\_balance(product\_id, location\_id, qty) values(?, ?, ?)" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,product\_id) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,qty) ibm\_db.execute(stmt2)

sql2="INSERT into productmovements(from\_location, to\_location, product\_id, qty) VALUES(?, ?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.bind\_param(stmt2,4,qty) ibm\_db.execute(stmt2)

sql = "select product\_num from products where product\_id=?" stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,product\_id) current\_num=ibm\_db.execute(stmt)

current\_num = ibm\_db.fetch\_assoc(stmt)

sql2="Update products set product\_num=? where product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,current\_num['PRODUCT\_NUM']-qty) ibm\_db.bind\_param(stmt2,2,product\_id)

ibm\_db.execute(stmt2) alert\_num=current\_num['PRODUCT\_NUM']-qty if(alert\_num<=0):

alert("Please update the quantity of the product {}, Atleast {} number of pieces must be added to finish the pending Product Movements!".format(product\_id,-alert\_num))

elif to\_location=="Main Inventory":

sql2="SELECT \* from product\_balance where location\_id=? and product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,product\_id) result=ibm\_db.execute(stmt2) result=ibm\_db.fetch\_assoc(stmt2)

app.logger.info(result) if result!=False:

if(len(result))>0:

Quantity = result["QTY"] q = Quantity - qty

sql2="UPDATE product\_balance set qty=? where location\_id=? and product\_id=?"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,q)

ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.execute(stmt2)

sql2="INSERT into productmovements(from\_location, to\_location, product\_id, qty) VALUES(?, ?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.bind\_param(stmt2,4,qty) ibm\_db.execute(stmt2)

flash("Product Movement Added", "success")

sql = "select product\_num from products where product\_id=?" stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,product\_id) current\_num=ibm\_db.execute(stmt)

current\_num = ibm\_db.fetch\_assoc(stmt)

sql2="Update products set product\_num=? where product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,current\_num['PRODUCT\_NUM']+qty) ibm\_db.bind\_param(stmt2,2,product\_id)

ibm\_db.execute(stmt2) alert\_num=q if(alert\_num<=0):

alert("Please Add {} number of {} to {} warehouse!".format(- q,product\_id,from\_location))

else:

raise CustomError("There is no product named {} in

{}.".format(product\_id,from\_location))

else: #will be executed if both from\_location and to\_location are specified

f=0

sql = "SELECT \* from product\_balance where location\_id=? and product\_id=?" stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt,1,from\_location) ibm\_db.bind\_param(stmt,2,product\_id) result=ibm\_db.execute(stmt)

result = ibm\_db.fetch\_assoc(stmt) if result!=False:

if(len(result))>0:

Quantity = result["QTY"] q = Quantity - qty

sql2="UPDATE product\_balance set qty=? where location\_id=? and product\_id=?"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,q) ibm\_db.bind\_param(stmt2,2,from\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.execute(stmt2)

f=1 alert\_num=q

if(alert\_num<=0):

alert("Please Add {} number of {} to {} warehouse!".format(- q,product\_id,from\_location))

else:

raise CustomError("There is no product named {} in

{}.".format(product\_id,from\_location)) if(f==1):

sql = "SELECT \* from product\_balance where location\_id=? and product\_id=?" stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt,1,to\_location)

ibm\_db.bind\_param(stmt,2,product\_id) result=ibm\_db.execute(stmt)

result = ibm\_db.fetch\_assoc(stmt) if result!=False:

if(len(result))>0:

Quantity = result["QTY"] q = Quantity + qty

sql2="UPDATE product\_balance set qty=? where location\_id=? and

product\_id=?"

else: values(?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,q) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.execute(stmt2)

sql2="INSERT into product\_balance(product\_id, location\_id, qty)

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,product\_id) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,qty) ibm\_db.execute(stmt2)

sql2="INSERT into productmovements(from\_location, to\_location, product\_id, qty) VALUES(?, ?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.bind\_param(stmt2,4,qty) ibm\_db.execute(stmt2)

flash("Product Movement Added", "success") render\_template('products.html',form=form) return redirect(url\_for('product\_movements'))

return render\_template('add\_product\_movements.html', form=form) #Delete Product Movements @app.route('/delete\_product\_movements/<string:id>', methods=['POST']) @is\_logged\_in

def delete\_product\_movements(id):

sql2="DELETE FROM productmovements WHERE movement\_id=?" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,id) ibm\_db.execute(stmt2)

flash("Product Movement Deleted", "success") return redirect(url\_for('product\_movements'))

if name == ' main ': app.secret\_key = "secret123"

#when the debug mode is on, we do not need to restart the server again and again app.run(debug=True)

# config.py

from flask import Flask, render\_template, flash, redirect, url\_for, session, request, logging from flask\_mysqldb import MySQL

from wtforms import Form, StringField, TextAreaField, PasswordField, validators,

SelectField, IntegerField import ibm\_db

from passlib.hash import sha256\_crypt from functools import wraps

import win32api

from sendgrid import \* #creating an app instance app = Flask( name )

conn=ibm\_db.connect("DATABASE=bludb;HOSTNAME=;PORT=;SECURITY=SSL;SSL ServerCertificate=DigiCertGlobalRootCA.crt;UID=;PWD=;",'','')

#Index @app.route('/') def index():

return render\_template('home.html') #Products

@app.route('/products') def products():

sql = "SELECT \* FROM products" stmt = ibm\_db.prepare(conn, sql) result=ibm\_db.execute(stmt) products=[]

row = ibm\_db.fetch\_assoc(stmt) while(row):

products.append(row)

row = ibm\_db.fetch\_assoc(stmt) products=tuple(products) #print(products)

if result>0:

return render\_template('products.html', products = products) else:

msg='No products found'

return render\_template('products.html', msg=msg) #Locations

@app.route('/locations')

def locations():

sql = "SELECT \* FROM locations" stmt = ibm\_db.prepare(conn, sql) result=ibm\_db.execute(stmt) locations=[]

row = ibm\_db.fetch\_assoc(stmt) while(row):

locations.append(row)

row = ibm\_db.fetch\_assoc(stmt) locations=tuple(locations) #print(locations)

if result>0:

return render\_template('locations.html', locations = locations) else:

msg='No locations found'

return render\_template('locations.html', msg=msg) #Product Movements @app.route('/product\_movements')

def product\_movements():

sql = "SELECT \* FROM productmovements" stmt = ibm\_db.prepare(conn, sql) result=ibm\_db.execute(stmt)

movements=[]

row = ibm\_db.fetch\_assoc(stmt) while(row):

movements.append(row)

row = ibm\_db.fetch\_assoc(stmt) movements=tuple(movements) #print(movements)

if result>0:

return render\_template('product\_movements.html', movements = movements) else:

msg='No product movements found'

return render\_template('product\_movements.html', msg=msg) #Register Form Class

class RegisterForm(Form):

name = StringField('Name', [validators.Length(min=1, max=50)]) username = StringField('Username', [validators.Length(min=1, max=25)]) email = StringField('Email', [validators.length(min=6, max=50)]) password = PasswordField('Password', [

validators.DataRequired(),

validators.EqualTo('confirm', message='Passwords do not match')

])

confirm = PasswordField('Confirm Password') #user register

@app.route('/register', methods=['GET','POST']) def register():

form = RegisterForm(request.form)

if request.method == 'POST' and form.validate(): name = form.name.data

email = form.email.data username = form.username.data

password = sha256\_crypt.encrypt(str(form.password.data))

sql1="INSERT INTO users(name, email, username, password) VALUES(?,?,?,?)" stmt1 = ibm\_db.prepare(conn, sql1)

ibm\_db.bind\_param(stmt1,1,name) ibm\_db.bind\_param(stmt1,2,email) ibm\_db.bind\_param(stmt1,3,username)

ibm\_db.bind\_param(stmt1,4,password) ibm\_db.execute(stmt1)

#for flash messages taking parameter and the category of message to be flashed flash("You are now registered and can log in", "success")

#when registration is successful redirect to home return redirect(url\_for('login'))

return render\_template('register.html', form = form) #User login

@app.route('/login', methods = ['GET', 'POST']) def login():

if request.method == 'POST': #Get form fields

username = request.form['username'] password\_candidate = request.form['password'] sql1="Select \* from users where username = ?" stmt1 = ibm\_db.prepare(conn, sql1) ibm\_db.bind\_param(stmt1,1,username) result=ibm\_db.execute(stmt1) d=ibm\_db.fetch\_assoc(stmt1)

if result > 0:

#Get the stored hash data = d

password = data['PASSWORD'] #compare passwords

if sha256\_crypt.verify(password\_candidate, password): #Passed

session['logged\_in'] = True session['username'] = username flash("you are now logged in","success")

return redirect(url\_for('dashboard')) else:

error = 'Invalid Login'

return render\_template('login.html', error=error) #Close connection

cur.close() else:

error = 'Username not found'

return render\_template('login.html', error=error) return render\_template('login.html')

#check if user logged in def is\_logged\_in(f):

@wraps(f)

def wrap(\*args, \*\*kwargs): if 'logged\_in' in session:

return f(\*args, \*\*kwargs) else:

flash('Unauthorized, Please login','danger') return redirect(url\_for('login'))

return wrap #Logout @app.route('/logout') @is\_logged\_in

def logout(): session.clear()

flash("You are now logged out", "success") return redirect(url\_for('login'))

#Dashboard @app.route('/dashboard')

@is\_logged\_in def dashboard():

sql2="SELECT product\_id, location\_id, qty FROM product\_balance" sql3="SELECT location\_id FROM locations"

stmt2 = ibm\_db.prepare(conn, sql2) stmt3 = ibm\_db.prepare(conn, sql3) result=ibm\_db.execute(stmt2) ibm\_db.execute(stmt3)

products=[]

row = ibm\_db.fetch\_assoc(stmt2) while(row):

products.append(row)

row = ibm\_db.fetch\_assoc(stmt2) products=tuple(products) locations=[]

row2 = ibm\_db.fetch\_assoc(stmt3) while(row2):

locations.append(row2)

row2 = ibm\_db.fetch\_assoc(stmt3) locations=tuple(locations)

locs = []

for i in locations: locs.append(list(i.values())[0])

if result>0:

return render\_template('dashboard.html', products = products, locations = locs) else:

msg='No products found'

return render\_template('dashboard.html', msg=msg) #Product Form Class

class ProductForm(Form):

product\_id = StringField('Product ID', [validators.Length(min=1, max=200)]) product\_cost = StringField('Product Cost', [validators.Length(min=1, max=200)]) product\_num = StringField('Product Num', [validators.Length(min=1, max=200)])

#Add Product

@app.route('/add\_product', methods=['GET', 'POST']) @is\_logged\_in

def add\_product():

form = ProductForm(request.form)

if request.method == 'POST' and form.validate(): product\_id = form.product\_id.data product\_cost = form.product\_cost.data product\_num = form.product\_num.data

sql1="INSERT INTO products(product\_id, product\_cost, product\_num) VALUES(?,?,?)"

stmt1 = ibm\_db.prepare(conn, sql1) ibm\_db.bind\_param(stmt1,1,product\_id) ibm\_db.bind\_param(stmt1,2,product\_cost) ibm\_db.bind\_param(stmt1,3,product\_num) ibm\_db.execute(stmt1)

flash("Product Added", "success") return redirect(url\_for('products'))

return render\_template('add\_product.html', form=form) #Edit Product

@app.route('/edit\_product/<string:id>', methods=['GET', 'POST']) @is\_logged\_in

def edit\_product(id):

sql1="Select \* from products where product\_id = ?" stmt1 = ibm\_db.prepare(conn, sql1)

ibm\_db.bind\_param(stmt1,1,id) result=ibm\_db.execute(stmt1) product=ibm\_db.fetch\_assoc(stmt1)

print(product) #Get form

form = ProductForm(request.form) #populate product form fields

form.product\_id.data = product['PRODUCT\_ID'] form.product\_cost.data = str(product['PRODUCT\_COST']) form.product\_num.data = str(product['PRODUCT\_NUM']) if request.method == 'POST' and form.validate():

product\_id = request.form['product\_id'] product\_cost = request.form['product\_cost'] product\_num = request.form['product\_num']

sql2="UPDATE products SET product\_id=?,product\_cost=?,product\_num=? WHERE product\_id=?"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,product\_id) ibm\_db.bind\_param(stmt2,2,product\_cost) ibm\_db.bind\_param(stmt2,3,product\_num) ibm\_db.bind\_param(stmt2,4,id) ibm\_db.execute(stmt2)

flash("Product Updated", "success") return redirect(url\_for('products'))

return render\_template('edit\_product.html', form=form) #Delete Product

@app.route('/delete\_product/<string:id>', methods=['POST']) @is\_logged\_in

def delete\_product(id):

sql2="DELETE FROM products WHERE product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,id)

ibm\_db.execute(stmt2) flash("Product Deleted", "success") return redirect(url\_for('products'))

#Location Form Class class LocationForm(Form):

location\_id = StringField('Location ID', [validators.Length(min=1, max=200)]) #Add Location

@app.route('/add\_location', methods=['GET', 'POST']) @is\_logged\_in

def add\_location():

form = LocationForm(request.form)

if request.method == 'POST' and form.validate(): location\_id = form.location\_id.data sql2="INSERT into locations VALUES(?)" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,location\_id) ibm\_db.execute(stmt2)

flash("Location Added", "success") return redirect(url\_for('locations'))

return render\_template('add\_location.html', form=form) #Edit Location

@app.route('/edit\_location/<string:id>', methods=['GET', 'POST']) @is\_logged\_in

def edit\_location(id):

sql2="SELECT \* FROM locations where location\_id = ?" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,id) result=ibm\_db.execute(stmt2) location=ibm\_db.fetch\_assoc(stmt2) #Get form

form = LocationForm(request.form) print(location)

#populate article form fields

form.location\_id.data = location['LOCATION\_ID'] if request.method == 'POST' and form.validate():

location\_id = request.form['location\_id']

sql2="UPDATE locations SET location\_id=? WHERE location\_id=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,location\_id) ibm\_db.bind\_param(stmt2,2,id)

ibm\_db.execute(stmt2) flash("Location Updated", "success") return redirect(url\_for('locations'))

return render\_template('edit\_location.html', form=form) #Delete Location

@app.route('/delete\_location/<string:id>', methods=['POST']) @is\_logged\_in

def delete\_location(id):

sql2="DELETE FROM locations WHERE location\_id=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,id)

ibm\_db.execute(stmt2) flash("Location Deleted", "success") return redirect(url\_for('locations'))

#Product Movement Form Class

class ProductMovementForm(Form):

from\_location = SelectField('From Location', choices=[]) to\_location = SelectField('To Location', choices=[]) product\_id = SelectField('Product ID', choices=[])

qty = IntegerField('Quantity') class CustomError(Exception):

pass

#Add Product Movement

@app.route('/add\_product\_movements', methods=['GET', 'POST']) @is\_logged\_in

def add\_product\_movements():

form = ProductMovementForm(request.form) sql2="SELECT product\_id FROM products" sql3="SELECT location\_id FROM locations" stmt2 = ibm\_db.prepare(conn, sql2)

stmt3 = ibm\_db.prepare(conn, sql3) result=ibm\_db.execute(stmt2) ibm\_db.execute(stmt3)

products=[]

row = ibm\_db.fetch\_assoc(stmt2) while(row):

products.append(row)

row = ibm\_db.fetch\_assoc(stmt2) products=tuple(products) locations=[]

row2 = ibm\_db.fetch\_assoc(stmt3) while(row2):

locations.append(row2)

row2 = ibm\_db.fetch\_assoc(stmt3)

locations=tuple(locations) prods = []

for p in products: prods.append(list(p.values())[0]) locs = []

for i in locations: locs.append(list(i.values())[0])

form.from\_location.choices = [(l,l) for l in locs] form.from\_location.choices.append(("Main Inventory","Main Inventory")) form.to\_location.choices = [(l,l) for l in locs] form.to\_location.choices.append(("Main Inventory","Main Inventory")) form.product\_id.choices = [(p,p) for p in prods]

if request.method == 'POST' and form.validate(): from\_location = form.from\_location.data to\_location = form.to\_location.data product\_id = form.product\_id.data

qty = form.qty.data

if from\_location==to\_location:

raise CustomError("Please Give different From and To Locations!!")

elif from\_location=="Main Inventory":

sql2="SELECT \* from product\_balance where location\_id=? and product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,to\_location) ibm\_db.bind\_param(stmt2,2,product\_id) result=ibm\_db.execute(stmt2) result=ibm\_db.fetch\_assoc(stmt2) print(" ")

print(result)

print(" ") app.logger.info(result) if result!=False:

if(len(result))>0:

Quantity = result["QTY"] q = Quantity + qty

sql2="UPDATE product\_balance set qty=? where location\_id=? and product\_id=?"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,q) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.execute(stmt2)

sql2="INSERT into productmovements(from\_location, to\_location, product\_id, qty) VALUES(?, ?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.bind\_param(stmt2,4,qty) ibm\_db.execute(stmt2)

else:

sql2="INSERT into product\_balance(product\_id, location\_id, qty) values(?, ?, ?)" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,product\_id) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,qty) ibm\_db.execute(stmt2)

sql2="INSERT into productmovements(from\_location, to\_location, product\_id, qty) VALUES(?, ?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.bind\_param(stmt2,4,qty) ibm\_db.execute(stmt2)

sql = "select product\_num from products where product\_id=?" stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,product\_id) current\_num=ibm\_db.execute(stmt)

current\_num = ibm\_db.fetch\_assoc(stmt)

sql2="Update products set product\_num=? where product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,current\_num['PRODUCT\_NUM']-qty) ibm\_db.bind\_param(stmt2,2,product\_id)

ibm\_db.execute(stmt2) alert\_num=current\_num['PRODUCT\_NUM']-qty if(alert\_num<=0):

alert("Please update the quantity of the product {}, Atleast {} number of pieces must be added to finish the pending Product Movements!".format(product\_id,-alert\_num))

elif to\_location=="Main Inventory":

sql2="SELECT \* from product\_balance where location\_id=? and product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,product\_id) result=ibm\_db.execute(stmt2) result=ibm\_db.fetch\_assoc(stmt2)

app.logger.info(result) if result!=False:

if(len(result))>0:

Quantity = result["QTY"] q = Quantity - qty

sql2="UPDATE product\_balance set qty=? where location\_id=? and product\_id=?"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,q) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.execute(stmt2)

sql2="INSERT into productmovements(from\_location, to\_location, product\_id, qty) VALUES(?, ?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.bind\_param(stmt2,4,qty) ibm\_db.execute(stmt2)

flash("Product Movement Added", "success")

sql = "select product\_num from products where product\_id=?" stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,product\_id) current\_num=ibm\_db.execute(stmt)

current\_num = ibm\_db.fetch\_assoc(stmt)

sql2="Update products set product\_num=? where product\_id=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,current\_num['PRODUCT\_NUM']+qty) ibm\_db.bind\_param(stmt2,2,product\_id)

ibm\_db.execute(stmt2) alert\_num=q

if(alert\_num<=0):

alert("Please Add {} number of {} to {} warehouse!".format(- q,product\_id,from\_location))

else:

raise CustomError("There is no product named {} in

{}.".format(product\_id,from\_location))

else: #will be executed if both from\_location and to\_location are specified

f=0

sql = "SELECT \* from product\_balance where location\_id=? and product\_id=?" stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt,1,from\_location) ibm\_db.bind\_param(stmt,2,product\_id) result=ibm\_db.execute(stmt)

result = ibm\_db.fetch\_assoc(stmt) if result!=False:

if(len(result))>0:

Quantity = result["QTY"] q = Quantity - qty

sql2="UPDATE product\_balance set qty=? where location\_id=? and product\_id=?"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,q) ibm\_db.bind\_param(stmt2,2,from\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.execute(stmt2)

f=1 alert\_num=q

if(alert\_num<=0):

alert("Please Add {} number of {} to {} warehouse!".format(- q,product\_id,from\_location))

else:

raise CustomError("There is no product named {} in

{}.".format(product\_id,from\_location)) if(f==1):

sql = "SELECT \* from product\_balance where location\_id=? and product\_id=?" stmt = ibm\_db.prepare(conn, sql)

ibm\_db.bind\_param(stmt,1,to\_location) ibm\_db.bind\_param(stmt,2,product\_id) result=ibm\_db.execute(stmt)

result = ibm\_db.fetch\_assoc(stmt) if result!=False:

if(len(result))>0:

Quantity = result["QTY"] q = Quantity + qty

sql2="UPDATE product\_balance set qty=? where location\_id=? and

product\_id=?"

else: values(?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,q) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.execute(stmt2)

sql2="INSERT into product\_balance(product\_id, location\_id, qty)

stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,product\_id) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,qty) ibm\_db.execute(stmt2)

sql2="INSERT into productmovements(from\_location, to\_location, product\_id, qty) VALUES(?, ?, ?, ?)"

stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,from\_location) ibm\_db.bind\_param(stmt2,2,to\_location) ibm\_db.bind\_param(stmt2,3,product\_id) ibm\_db.bind\_param(stmt2,4,qty) ibm\_db.execute(stmt2)

flash("Product Movement Added", "success") render\_template('products.html',form=form) return redirect(url\_for('product\_movements'))

return render\_template('add\_product\_movements.html', form=form) #Delete Product Movements @app.route('/delete\_product\_movements/<string:id>', methods=['POST']) @is\_logged\_in

def delete\_product\_movements(id):

sql2="DELETE FROM productmovements WHERE movement\_id=?" stmt2 = ibm\_db.prepare(conn, sql2)

ibm\_db.bind\_param(stmt2,1,id) ibm\_db.execute(stmt2)

flash("Product Movement Deleted", "success") return redirect(url\_for('product\_movements'))

if name == ' main ': app.secret\_key = "secret123"

#when the debug mode is on, we do not need to restart the server again and again app.run(debug=True)