|  |  |
| --- | --- |
|  | # Inventory Manager |
|  |  |
|  | from flask import jsonify, json |
|  |  |
|  | from flask import Flask, request, redirect, render\_template, session |
|  | from cs50 import SQL |
|  | from tempfile import mkdtemp |
|  | from flask\_session import Session |
|  | from werkzeug.exceptions import default\_exceptions, HTTPException, InternalServerError |
|  | from werkzeug.security import check\_password\_hash, generate\_password\_hash |
|  |  |
|  | from functions import login\_required, apology |
|  |  |
|  | # Configure application |
|  | app = Flask(\_\_name\_\_) |
|  |  |
|  | # Ensure tamplates are auto-reloaded |
|  | app.config["TEMPLATES\_AUTO\_RELOAD"] = True |
|  |  |
|  | # Ensure responses aren't cached |
|  | @app.after\_request |
|  | def after\_request(response): |
|  | response.headers["Cache-Control"] = "no-cache, no-store, must-revalidate" |
|  | response.headers["Expires"] = 0 |
|  | response.headers["Pragma"] = "nocache" |
|  | return response |
|  |  |
|  | # Configure session to use filesystem |
|  | app.config["SESSION\_FILE\_DIR"] = mkdtemp() |
|  | app.config["SESSION\_PERMANENT"] = False |
|  | app.config["SESSION\_TYPE"] = "filesystem" |
|  | Session(app) |
|  |  |
|  |  |
|  | # Configure CS50 Library to use SQLite database |
|  | db = SQL("sqlite:///ims.db") |
|  |  |
|  |  |
|  | @app.route("/") |
|  | @login\_required |
|  | def index(): |
|  | """Homepage""" |
|  | totals = db.execute("SELECT COUNT(productName) as total\_products, SUM(inventoryOut) as total\_orders, SUM(inventoryIn) as total\_purchases FROM products") |
|  | ups = db.execute("SELECT COUNT(id) as total\_ups FROM products WHERE inventoryOnHand < minimumReq AND minimumReq != 0") |
|  |  |
|  | return render\_template("index.html", totals = totals, ups = ups) |
|  |  |
|  | @app.route("/get\_data") |
|  | def get\_data(): |
|  | """ Sends data to charts""" |
|  | totals = db.execute("SELECT COUNT(productName) as total\_products, SUM(inventoryOut) as total\_orders, SUM(inventoryIn) as total\_purchases FROM products") |
|  | labels = ["Total Products", "Total Orders", "Total Purchases"] |
|  | data = [ totals[0]["total\_products"],totals[0]["total\_orders"],totals[0]["total\_purchases"]] |
|  |  |
|  |  |
|  | mtotal = db.execute( |
|  | "SELECT SUM(CASE WHEN strftime('%m', oDate) = '01' THEN 1 ELSE 0 END) as jan, SUM(CASE WHEN strftime('%m', oDate) = '02' THEN 1 ELSE 0 END) as feb, SUM(CASE WHEN strftime('%m', oDate) = '03' THEN 1 ELSE 0 END) as mar, SUM(CASE WHEN strftime('%m', oDate) = '04' THEN 1 ELSE 0 END) as apr, SUM(CASE WHEN strftime('%m', oDate) = '05' THEN 1 ELSE 0 END) as may, SUM(CASE WHEN strftime('%m', oDate) = '06' THEN 1 ELSE 0 END) as jun, SUM(CASE WHEN strftime('%m', oDate) = '07' THEN 1 ELSE 0 END) as jul, SUM(CASE WHEN strftime('%m', oDate) = '08' THEN 1 ELSE 0 END) as aug, SUM(CASE WHEN strftime('%m', oDate) = '09' THEN 1 ELSE 0 END) as sep, SUM(CASE WHEN strftime('%m', oDate) = '10' THEN 1 ELSE 0 END) as oct, SUM(CASE WHEN strftime('%m', oDate) = '11' THEN 1 ELSE 0 END) as nov, SUM(CASE WHEN strftime('%m', oDate) = '12' THEN 1 ELSE 0 END) as dec FROM orders" |
|  | ) |
|  |  |
|  | mtotals = [ mtotal[0]["jan"], mtotal[0]["feb"], mtotal[0]["mar"], mtotal[0]["apr"], mtotal[0]["may"], mtotal[0]["jun"], mtotal[0]["jul"], mtotal[0]["aug"], mtotal[0]["sep"], mtotal[0]["oct"], mtotal[0]["nov"], mtotal[0]["dec"]] |
|  |  |
|  | return jsonify({'payload':json.dumps({'data':data, 'labels':labels, 'mtotals':mtotals})}) |
|  |  |
|  |  |
|  | @app.route("/login", methods=["GET", "POST"]) |
|  | def login(): |
|  | """Log user in""" |
|  |  |
|  | # Forget any user\_id |
|  | session.clear() |
|  |  |
|  | # User reached via POST |
|  | if request.method == "POST": |
|  |  |
|  | # Query database for username |
|  | rows = db.execute("SELECT \* FROM users WHERE username = :username", |
|  | username = request.form.get("username")) |
|  |  |
|  | if len(rows) != 1 or not check\_password\_hash(rows[0]["hash"], request.form.get("pwd")): |
|  | return "Incorrect ID or Pass" #render\_template("404.html") |
|  |  |
|  | # Remember which user has logged in |
|  | session["user\_id"] = rows[0]["id"] |
|  | session["fullname"] = rows[0]["fullname"] |
|  |  |
|  | # Redirect user to Dashboard |
|  | return redirect("/") |
|  |  |
|  | else: |
|  | return render\_template("login.html") |
|  |  |
|  | @app.route("/logout") |
|  | def logout(): |
|  | """Log user out""" |
|  |  |
|  | # Forget any user\_id |
|  | session.clear() |
|  |  |
|  | # Redirect user to login form |
|  | return redirect("/") |
|  |  |
|  | @app.route("/register", methods=["GET", "POST"]) |
|  | def register(): |
|  | """Registers User""" |
|  |  |
|  | # User reached via POST |
|  | if request.method == "POST": |
|  |  |
|  | hash = generate\_password\_hash(request.form.get("pwd")) |
|  |  |
|  | new\_user = db.execute("INSERT INTO users (username, hash, fullname) VALUES (:username, :hash, :fullname)", |
|  | username = request.form.get("username"), hash = hash, fullname = request.form.get("fullname")) |
|  |  |
|  | if not new\_user: |
|  | return render\_template("404.html") |
|  |  |
|  | session["user\_id"] = new\_user |
|  |  |
|  | return redirect("/") |
|  |  |
|  | else: |
|  | return render\_template("register.html") |
|  |  |
|  |  |
|  | @app.route("/change", methods=["GET", "POST"]) |
|  | def change(): |
|  | """Change user's password""" |
|  | if request.method == "POST": |
|  |  |
|  | if request.form.get("new-pwd") != request.form.get("new-pwd-again"): |
|  | return "Password not matched" |
|  |  |
|  | hash = generate\_password\_hash(request.form.get("new-pwd")) |
|  |  |
|  | db.execute("UPDATE users SET hash = :hash WHERE id = :user\_id", hash = hash, user\_id = session["user\_id"]) |
|  |  |
|  | return redirect("/") |
|  |  |
|  | else: |
|  | return render\_template("change.html") |
|  |  |
|  | @app.route("/productv", methods=["GET", "POST"]) |
|  | @login\_required |
|  | def productv(): |
|  |  |
|  | if request.method == "POST": |
|  |  |
|  |  |
|  | test = db.execute("INSERT INTO products (productName, partNumber, productLabel, startingInventory, inventoryIn, inventoryOut, inventoryOnHand, minimumReq) VALUES (:productName, :partNumber, :productLabel, :startingInventory, :inventoryIn, :inventoryOut, :inventoryOnHand, :minimumReq)", productName = request.form.get("product-name"), partNumber = request.form.get("part-number"), productLabel = request.form.get("product-label"), startingInventory = request.form.get("starting-inventory"), inventoryIn = request.form.get("inventory-in"), inventoryOut = request.form.get("inventory-out"), inventoryOnHand = request.form.get("inventory-on-hand"), minimumReq = request.form.get("minimum-req")) |
|  | if not test: |
|  | return "Can't Insert the data" |
|  |  |
|  |  |
|  | return redirect("/productv") |
|  |  |
|  | else: |
|  | return render\_template("productv.html") |
|  |  |
|  |  |
|  | @app.route("/orderv", methods=["GET", "POST"]) |
|  | @login\_required |
|  | def orderv(): |
|  | """Adds Orders""" |
|  | if request.method == "POST": |
|  |  |
|  | try: |
|  | orders = int(request.form.get("orders")) |
|  | except: |
|  | return "Orders Must be positive number" |
|  |  |
|  | if orders <= 0: |
|  | return "Orders Must be greater than 0" |
|  |  |
|  |  |
|  | db.execute("UPDATE products SET inventoryOnHand = inventoryOnHand - :orders, inventoryOut = inventoryOut + :orders WHERE productName = :productName", productName = request.form.get("product-name"), orders = request.form.get("orders")) |
|  | db.execute("INSERT INTO orders (first, middle, last, product\_name, numberOut, oDate) VALUES (:first, :middle, :last, :product\_name, :numberOut, :oDate)", first = request.form.get("first-name"), middle = request.form.get("middle-name"), last = request.form.get("last-name"), product\_name = request.form.get("product-name"), numberOut = request.form.get("orders"), oDate = request.form.get("order-date")) |
|  |  |
|  |  |
|  | return redirect("/orderv") |
|  | else: |
|  | products = db.execute("SELECT id, productName, partNumber FROM products") |
|  |  |
|  | return render\_template("orderv.html", products = products) |
|  |  |
|  | @app.route("/purchasev", methods=["GET", "POST"]) |
|  | @login\_required |
|  | def purchasev(): |
|  | """Adds Purchase Record""" |
|  | if request.method == "POST": |
|  |  |
|  | try: |
|  | purchases = int(request.form.get("purchases")) |
|  | except: |
|  | return "Purchases Must be positive number" |
|  |  |
|  | if purchases <= 0: |
|  | return "Purchases Must be greater than 0" |
|  |  |
|  |  |
|  | db.execute("UPDATE products SET inventoryOnHand = inventoryOnHand + :purchases, inventoryIn = inventoryIn + :purchases WHERE productName = :productName", productName = request.form.get("product-name"), purchases = request.form.get("purchases")) |
|  | db.execute("INSERT INTO purchases (supplier\_name, product\_name, numberIn, pDate) VALUES (:supplier\_name, :product\_name, :numberIn, :pDate)", supplier\_name = request.form.get("supplier-name"), product\_name = request.form.get("product-name"), numberIn = request.form.get("purchases"), pDate = request.form.get("purchase-date")) |
|  |  |
|  |  |
|  | return redirect("/purchasev") |
|  | else: |
|  | products = db.execute("SELECT id, productName, partNumber FROM products") |
|  | suppliers = db.execute("SELECT id, supplier FROM suppliers") |
|  |  |
|  | return render\_template("purchasev.html", products = products, suppliers = suppliers) |
|  |  |
|  | @app.route("/supplierv", methods=["GET", "POST"]) |
|  | @login\_required |
|  | def supplierv(): |
|  | """Add Suppliers""" |
|  | if request.method == "POST": |
|  |  |
|  | db.execute("INSERT INTO suppliers (supplier) VALUES (:supplier)", supplier = request.form.get("supplier")) |
|  |  |
|  | return render\_template("supplierv.html") |
|  |  |
|  |  |
|  | @app.route("/productl", methods=["POST", "GET"]) |
|  | @login\_required |
|  | def productl(): |
|  | """ Lists Products""" |
|  |  |
|  | rows = db.execute("SELECT \* FROM products") |
|  |  |
|  | return render\_template("productl.html", rows = rows) |
|  |  |
|  | @app.route("/orderl", methods=["GET", "POST"]) |
|  | @login\_required |
|  | def orderl(): |
|  | """ Lists Orders""" |
|  |  |
|  | rows = db.execute("SELECT orderId, first, middle, last, product\_name, numberOut, oDate, partNumber FROM orders JOIN products ON orders.product\_name = products.productName") |
|  |  |
|  | return render\_template("orderl.html", rows = rows) |
|  |  |
|  |  |
|  | @app.route("/purchasel", methods=["GET", "POST"]) |
|  | @login\_required |
|  | def purchasel(): |
|  | """Lists Purchses""" |
|  |  |
|  | rows = db.execute("SELECT purchaseId, supplier\_name, product\_name, numberIn, pDate, partNumber FROM purchases JOIN products ON purchases.product\_name = products.productName") |
|  |  |
|  | return render\_template("purchasel.html", rows = rows) |
|  |  |
|  |  |
|  |  |
|  |  |
|  | app.run(debug=True,port=5001) |