

ASSIGNMENT 3

G.SRIARUNA

VV COLLEGE OF ENGINEERING

III- YEAR CSE

(TEAM MEMBER 1)

Task 1

INVENTORY MANAGEMENT SYSTEM


Login

Username

Password

Login

INVENTORY MANAGEMENT SYSTEM



Efficiency in inventory means the ability to quickly receive and store products as they come in and retrieve and ship when they go out. Every extra second spent in these processes adds to the costs of inventory management.

Register

Username

Email

Password

Register

Already have an account? [Login](#)

Docker image to the IBM container registry

```
app.secret_key = 'a'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-89
print("connected")
```

```
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/home')
def home1():
    return render_template('index.html')
@app.route('/about')
def about():
    return render_template('index.html')
@app.route('/meme')
def mem():
    return render_template('meme.html')
@app.route('/login')
def login():
    return render_template('login.html')
@app.route('/register')
def register():
    return render_template('register.html')
@app.route('/logout')
def logout():
    return render_template('index.html')
```

```

@app.route("/reg", methods=['POST', 'GET'])
def signup():
    msg = ''
    if request.method == 'POST':
        name = request.form["name"]
        email = request.form["email"]
        password = request.form["password"]
        sql = "SELECT * FROM REGISTER WHERE name= ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, name)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            return render_template('login.html', error=True)
        elif not re.match(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z]+', email):
            msg = "Invalid Email Address!"
        else:
            insert_sql = "INSERT INTO REGISTER VALUES (?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            # this username & password should be same as db-2 details & order also
            ibm_db.bind_param(prepare_stmt, 1, name)
            ibm_db.bind_param(prepare_stmt, 2, email)
            ibm_db.bind_param(prepare_stmt, 3, password)
            ibm_db.execute(prepare_stmt)
            msg = "You have successfully registered !"
    return render_template('login.html', msg=msg)

```

```

@app.route("/log", methods=['POST', 'GET'])
def login1():
    if request.method == "POST":
        email = request.form["email"]
        password = request.form["password"]
        sql = "SELECT * FROM REGISTER WHERE EMAIL=? AND PASSWORD=?" # from db2 sql table
        stmt = ibm_db.prepare(conn, sql)
        # this username & password should be same as db-2 details & order also
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['loggedin'] = True
            session['id'] = account['EMAIL']
            session['email'] = account['EMAIL']
            return render_template('meme.html')
        else:
            msg = "Incorrect Email/password"
            return render_template('login.html', msg=msg)
    else:
        return render_template('login.html')

```

```

@app.route('/test', methods=['POST', 'GET'])
def meme():
    if request.method == 'POST':
        keywords = request.form["key"]
        print(keywords)
        url = "https://humor-jokes-and-memes.p.rapidapi.com/memes/search"

        querystring = {"keywords": keywords, "media-type": "image", "keywords-in-image": "false", "min-rating": "3",
            "number": "2"}

        headers = {
            "X-RapidAPI-Key": "42114a62a7mshb20f7b5bef96ebep1e7842jsn2d4c884c1e23",
            "X-RapidAPI-Host": "humor-jokes-and-memes.p.rapidapi.com"}

        response = requests.request("GET", url, headers=headers, params=querystring)
        print(response.text)
        output = json.loads(response.text)
        print(output)
        op1 = output['memes'][0]['url']
        webbrowser.open(op1)
        op2 = output['memes'][1]['url']
        webbrowser.open(op2)
    return render_template('meme.html', output1=op1, output2=op2)

```

Output



Docker container using kubernetes



Rapid API Search for APIs **API Hub** Organizations Apps My APIs

Humor-Jokes-and-Memes **FREE** Verified Popularity 9.3 / 10 Latency 548ms Service Level 100%

By humorapi | Updated 9 months ago | Entertainment

Endpoints About Tutorials Discussions Pricing Subscribed

Humor-Jokes-and-Memes API Documentation

Humor API lets you search through 50,000 jokes, over 100,000 memes, and gifs. From Chuck Norris, over Knock Knock to Yo Mama jokes, you'll find something for every occasion.

Search endpoints **GET search-memes** **Test Endpoint**

jokes

- GET search-jokes
- GET random-joke
- POST analyze-joke
- POST submit-joke

Search for memes. See <https://humorapi.com/docs/#Search-Memes> for more.

[Search Memes - Official Documentation](#)

Code Snippets **Example Responses** **Results**

(Python) Requests **Copy Code**

```
import requests

url = "https://humor-jokes-and-memes.p.rapidapi.com/memes/search"

querystring = {"keywords":"rocket","media-type":"image","keywords-in-image":"false","min-rating":"3","number":"3"}
```