PROJECT REPORT

Date	**17.11.2022**
Team ID	**PNT2022TMID022014**
Project Name	**Skill/Job Recommender Application**
Team Members	**KAVITHA.G** **LOGAVARSHINI.P** **BOLLU GEETA** **ARCHANA.C**

XIII Git Hub

1. INTRODUCTION

Project Overview

Exploring and finding a job has never been easy, it's because you don't know enough about the organization's mission, workplace culture, or open positions, or because you can't seem to locate the appropriate individual with the correct qualities.

Since then to solve these issues, the online job search portals have been developed, making it easier for both parties to find jobs and to recruit candidates. The job portal brings together recruiters and job seekers with the goal of satisfying each party's specific needs.

They are the quickest and easily accessible form of communication, regardless of the distance between the recruiter and the candidate who is in search of a perfect job role and company for him.

Purpose

Online employment portals have been around for a while, however they have simply created more difficulties, such as:

- Individual skill development is not usually a priority in the education system.
- Spending hours sifting through the vast volume of web posts to uncover relevant information.
- Those who lack industry knowledge are unsure of exactly what they need to study to find a job.

2. LITERATURE SURVEY

Existing problem

- Job portals consider the information available in the curriculum vitae to select the candidate rather than considering his skill set.
- Recruiters see many number of resumes for a single job role, so they face difficulty in choosing the ideal candidate for particular role.

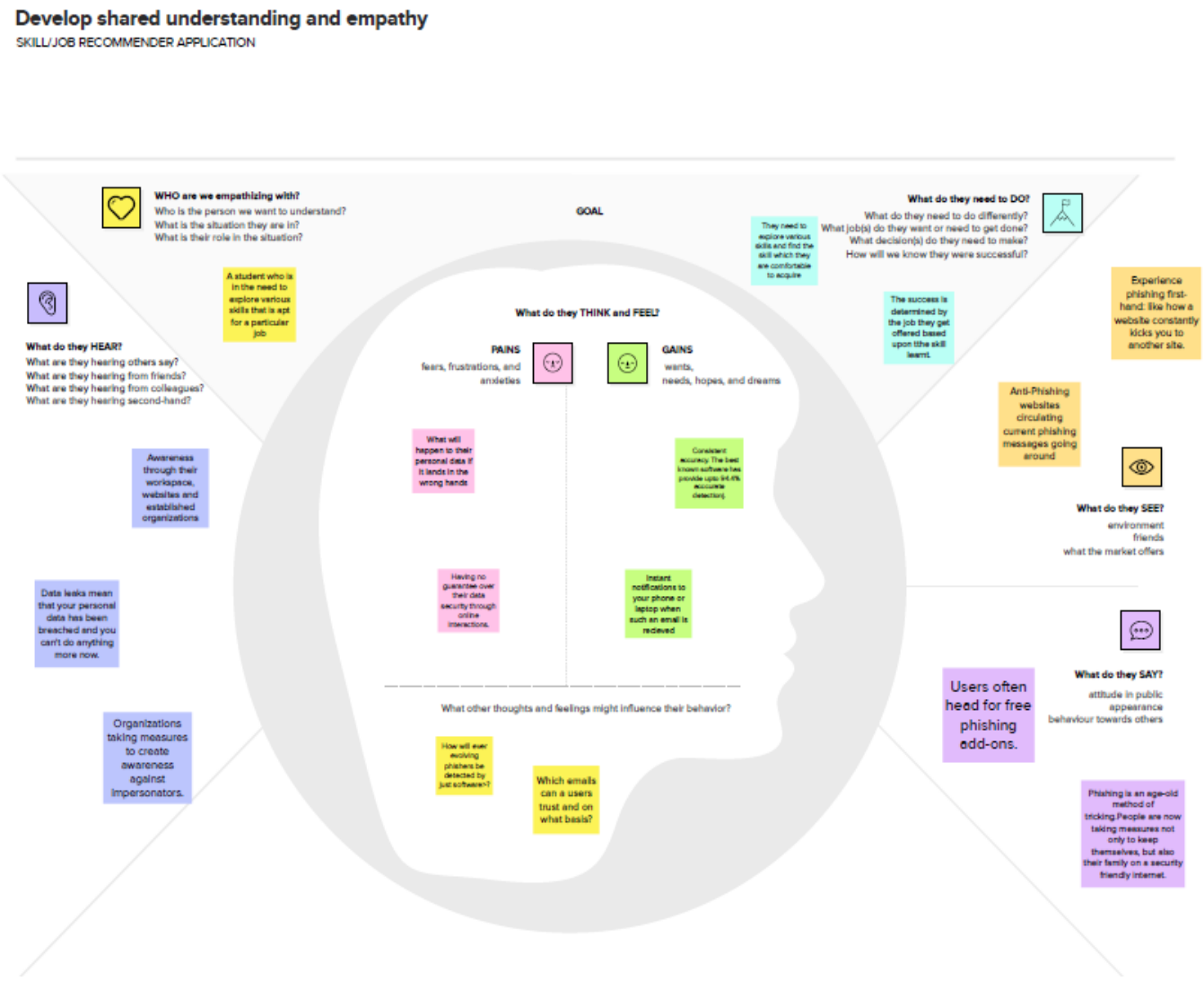
Problem Statement Definition

How can we customise job searches and recommend jobs based on the user's skill set while securely storing user and recruiter data

3. IDEATION & PROPOSED SOLUTION

Empathy Map Canvas

An empathy map is a template that organizes a user's behaviors and feelings to create a sense of empathy between the user and developer. The empathy map represents a principal user and helps teams better understand their motivations, concerns, and user experience.



Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

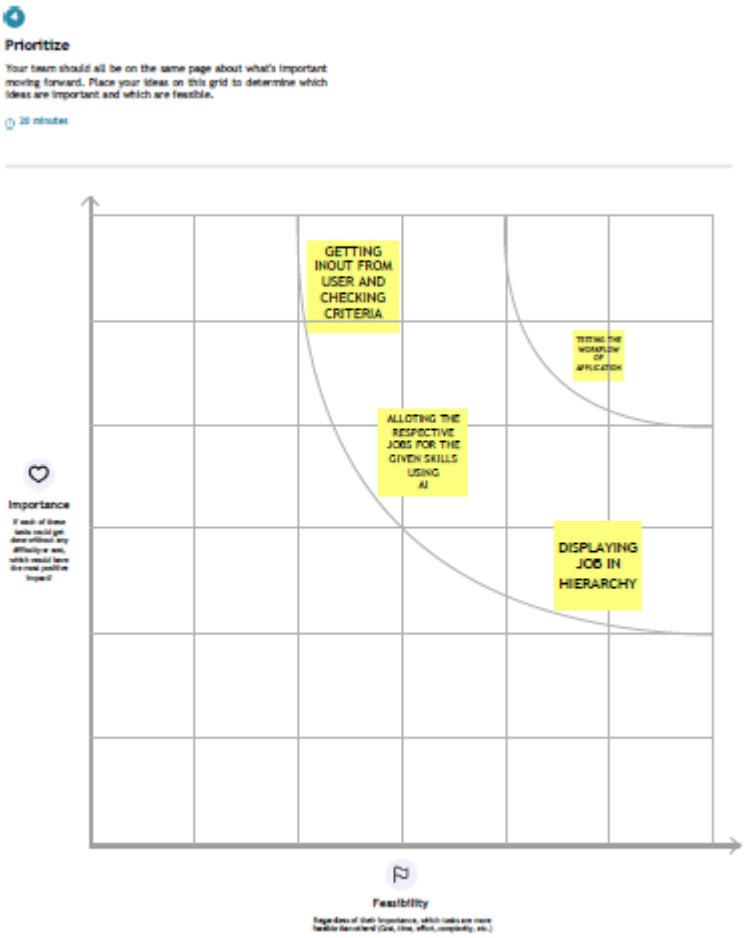
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Step-1: Team Gathering, Collaboration and Select the Problem Statement.

Step-2: Brainstorm

Step-3: Group Ideas

Step-4: Idea Prioritization



Proposed Solution

Based on the user's skill set and preferences, the system customises and only displays recommended jobs (Using graphql api).

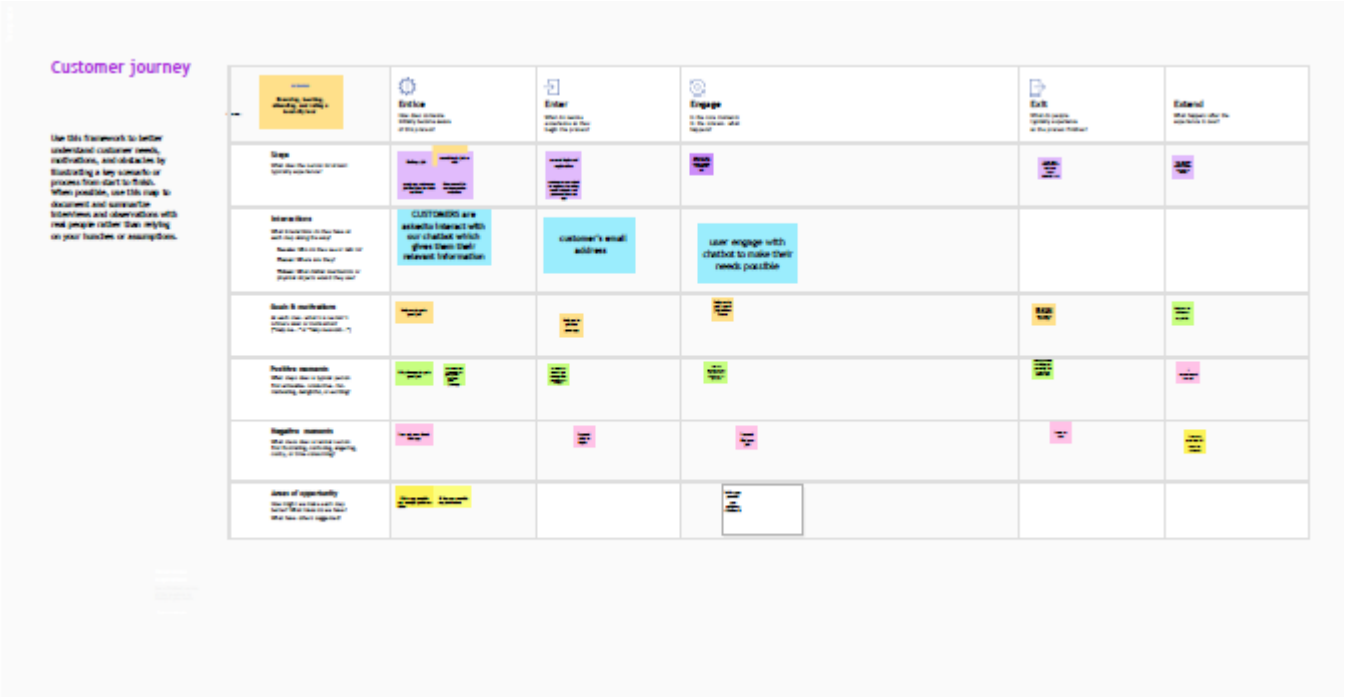
Similarly, the same recommendation system assists job recruiters in finding the most qualified candidates for their organisation.

All critical data, including personal information from job seekers and hosts, must be stored safely and securely. Using a SQL database is the simplest, safest, and most convenient method.

In some cases, such as when information is shared with the host while applying for a job, data must also be kept private.

Problem Solution fit

Customer Problem Statement



4. REQUIREMENT ANALYSIS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Utilizing a Form for Registration signing up with Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Chat Bot	A chatbot will be available on the website to address user concerns and issues about job applications, job searches, and much more.
FR-4	User Login	Log in using the Register credentials
FR-5	User Search	Job exploration using suggested skills and job filters.
FR-6	User Profile	The login credentials are used to update the user profile.
FR-7	User Acceptance	Confirmation of the Job.

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

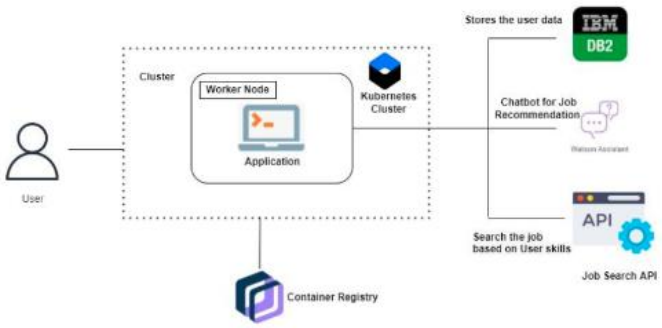
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Job searchers can log in and search for jobs based on their skill sets using this programme.
NFR-2	**Security**	This application has separate logins for job recruiters and job seekers, making it secure.
NFR-3	**Reliability**	You can use this application for free and without having to pay anything because it is open-source. All job seekers will have unlimited access to the massive employment postings..
NFR-4	**Performance**	This application responds more quickly and completes tasks in a shorter amount of time.
NFR-5	**Availability**	This programme advises skills for specific job vacancies and offers jobs.
NFR-6	**Scalability**	The Response time of the application is quite faster compared to any other application.

5. PROJECT DESIGN

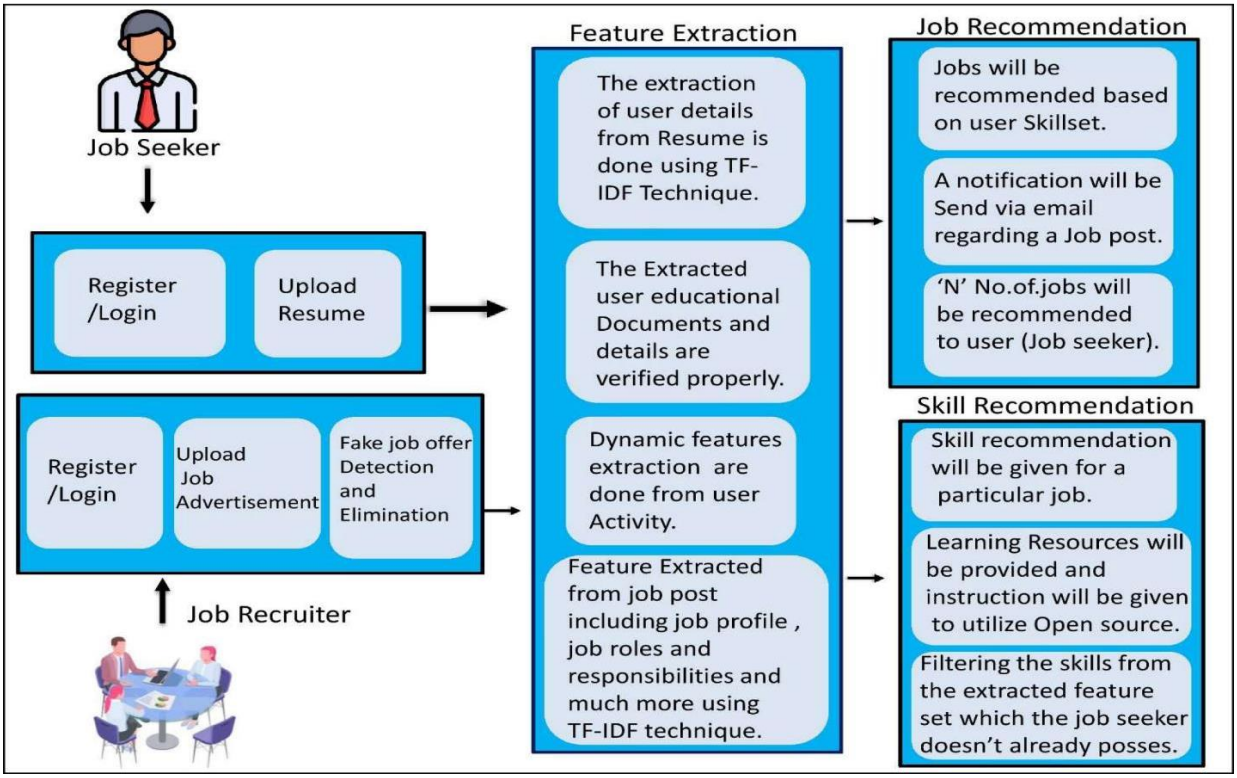
Data Flow Diagrams

Data Flow Diagrams:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.



Solution & Technical Architecture



User Stories

Functional Requirement (Epic)	User Story / Task
UI Design & Frontend Development	As a user I can expect to experience a cool user interface and smooth user experience
Home	As a user, I will land on the landing page of the website
Database	As a user my data will be stored in database for further use
Registration	As a user, I can register for the application as a Job seeker or Recruiter.
	As a user, I will receive verification email once I have registered for the application
	As a user, I can register for the application through Google Signup
	As a user, I can register for the application through Sign in with LinkedIn
Login	As a user, I can log into the application as Jobseeker or Recruiter by entering registered email & correct password
	As a user, I can log into the application using google sign in option
	As a user, I can log into the application using LinkedIn Login
Profile Setup	As a fresh user I need to setup my profile initially by filling required details which can be modified later
	As a fresh recruiter I need to setup profile for my company by filling required details which can be modified later
Cloud Storage	As a user I can upload my Image, Resume and much more in the website
Posting	As a Recruiter I can post various job openings
Job Listing	As a user I can access jobs posted by recruiters and Google Job Search API
Applying	As a Job Seeker I can view all Job openings in the home page and also, I can search for specific jobs and apply for the same
Shortlisting	As a Recruiter I can view applied candidates and shortlist few among them.
Chatbot	As a User I can access chatbot to avail any kind of guidance in the website
Notification (SendGrid)	As a User, I can get notification on new Job openings via email using SendGrid service
Courses & Webinars	As an administrator I can suggest users' various courses f rom famous websites like Udemy, Coursera based on their skillset to improve their skills
Interviews	As a recruiter I can schedule face to face Interview with shortlisted candidates using WebRTC framework
	As a shortlisted candidate I can join the scheduled interview using the meeting link
System testing	As a user I can access my website without any fault or malfunction
Docker	As a user I can access my containerized application in any device
Kubernetes	As a user I can access my containerized application in any device with greater security

6. PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	18	6 Days	24 Oct 2022	29 Oct 2022	18	29 Oct 2022
Sprint-2	27	6 Days	31 Oct 2022	05 Nov 2022	27	05 Nov 2022
Sprint-3	29	6 Days	07 Nov 2022	12 Nov 2022	29	12 Nov 2022
Sprint-4	14	6 Days	14 Nov 2022	19 Nov 2022	14	19 Nov 2022

Sprint Delivery Schedule

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	UI Creation Creating Registration page, Login page	10	Medium	KAVITHA LOGAVARSHINI BOLLU GEETHA ARCHANA
Sprint-1	Database Connectivity	USN-2	Viewing and applying jobs Connecting UI with Database	10	High	KAVITHA LOGAVARSHINI BOLLUGEETHA ARCHANA
Sprint-2	SendGrid Integration	USN-3	SendGrid Integration with Python Code	10	Low	KAVITHA LOGAVARSHINI BOLLUGEETHA ARCHANA
Sprint-2	Chatbot Development	USN-4	Building a chatbot	10	High	KAVITHA LOGAVARSHINI BOLLUGEETHA ARCHANA
Sprint-3	Integration and Containerisation	USN-5	Integrating chatbot to the HTML page and containerizing the app.	20	Medium	KAVITHA LOGAVARSHINI BOLLUGEETHA ARCHANA
Sprint-4	Upload Image and deployment	USN-6	Upload the image to the IBM Registry and deploy it in the Kubernetes Cluster.	20	High	KAVITHA LOGAVARSHINI BOLLUGEETHA ARCHANA

7. CODING&SOLUTIONING

Feature 1

Skill based job recommendation – Jobs are recommended based on job seeker's skill and requirements. This also brings in custom list of jobs that's different for different job seekers.

Feature 2

Hosting jobs– Job hoster can easily host jobs that can be accessed by a varied range of applicants. Additional feature – filtering through jobs based on skill, location, salary/stipend, mode of job (for both applying and hosting jobs).

Code

```
import ibm_db

from flask import Flask, url_for, render_template, request, session, redirect, flash,
send_file
from authlib.integrations.flask_client import OAuth
import traceback
from datetime import date
from io import BytesIO

app = Flask(_name_)
oauth = OAuth(app)
arr2=[]

def connection():
    try:
        #jesima db2 credential
        conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-a1f5-
23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;\
            PORT=32716;PROTOCOL=TCPIP;UID=rmy92863;PWD=DDoUqjA0drfzoKCm;SECURITY=SSL;SS
LServiceCertificate=DigiCertGlobalRootCA.crt", "", "")
        print("CONNECTED TO DATABASE")
```

```
        return conn
    except:
        print(ibm_db.conn_errormsg())
        print("CONNECTION FAILED")

@app.route('/google')
def google():
    GOOGLE_CLIENT_ID = '367786402665-
skc738qj1tacaf0kkrkcgolap5775qia.apps.googleusercontent.com'
    GOOGLE_CLIENT_SECRET = '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'

    CONF_URL = 'https://accounts.google.com/.well-known/openid-configuration'
    oauth.register(
        name='google',
        client_id=GOOGLE_CLIENT_ID,
        client_secret=GOOGLE_CLIENT_SECRET,
        server_metadata_url=CONF_URL,
        client_kwargs={
            'scope': 'openid email profile'
        }
    )
    # Redirect to google_auth function
    redirect_uri = url_for('google_auth', _external=True)
    return oauth.google.authorize_redirect(redirect_uri)

@app.route('/google/auth')
def google_auth():
    token = oauth.google.authorize_access_token()
    user = oauth.google.parse_id_token(token,None)
    print(" Google User ", user)
    try:
        conn=connection()
        sql="INSERT INTO USERS VALUES(?,?)"
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt, 1, user['name'])
        ibm_db.bind_param(stmt, 2,user['email'])

        out=ibm_db.execute(stmt)
    except Exception as e:
        print(e)
    return render_template('index.html')

#Home Page
@app.route("/")
def home():
```

```
    return render_template('index.html')

#Logout
@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('username',  None)
    return render_template("index.html")

#Filter Jobs
@app.route('/FilteredJobs',methods=['POST','GET'])
def FilteredJobs():
    #arr=[]
    if request.method == "POST":
            data = {}
            data['role'] = request.json['role']
            data['loc'] = request.json['loc']
            data['type'] = request.json['type']

            try:
                conn=connection()
                sql ="SELECT * FROM JOBS WHERE (LOCATION = ? AND JOBTYPE = ?) AND ROLE
= ? "
                stmt = ibm_db.prepare(conn,sql)
                ibm_db.bind_param(stmt, 1, data['loc'])
                ibm_db.bind_param(stmt,2,data['type'])
                ibm_db.bind_param(stmt,3,data['role'])
                out=ibm_db.execute(stmt)
                while ibm_db.fetch_row(stmt)  !=  False:
                     inst={}
                     inst['COMPANY']=ibm_db.result(stmt,1)
                     inst['ROLE']=ibm_db.result(stmt,3)
                     inst['SALARY']=ibm_db.result(stmt,11)
                     inst['LOCATION']=ibm_db.result(stmt,10)
                     inst['JOBTYPE']=ibm_db.result(stmt,5)
                     inst['POSTEDDATE']=ibm_db.result(stmt,16)

                     arr2.append(inst)
                     print(arr2)

            except Exception as e:
                print(e)

    return render_template('job_listing.html',arr=arr2)
```

```
@app.route('/filter')
def filter():
    return render_template('job_listing.html',arr=arr2)

#Job Listing - Seeker Home Page
@app.route('/job_listing')
def job_listing():
    try:
        conn=connection()
        arr=[]
        sql="SELECT * FROM JOBS"
        stmt = ibm_db.exec_immediate(conn, sql)
        dictionary = ibm_db.fetch_both(stmt)
        while dictionary != False:
             inst={}
             inst['JOBID']=dictionary['JOBID']
             inst['COMPANY']=dictionary['COMPANY']
             inst['ROLE']=dictionary['ROLE']
             inst['SALARY']=dictionary['SALARY']
             inst['LOCATION']=dictionary['LOCATION']
             inst['JOBTYPE']=dictionary['JOBTYPE']
             inst['POSTEDDATE']=dictionary['POSTEDDATE']
             inst['LOGO']=BytesIO(dictionary['LOGO'])
             arr.append(inst)
             dictionary = ibm_db.fetch_both(stmt)
    except Exception as e:
        print(e)
    return render_template('job_listing.html',arr=arr)

#Register
@app.route("/register",methods=["GET","POST"])
def registerPage():
    if request.method=="POST":
        conn=connection()
        try:
            role=request.form["urole"]
            if role=="seeker":
                sql="INSERT INTO SEEKER
VALUES('{}','{}','{}','{}','{}','{}')".format(request.form["uemail"],request.form["upas
s"],request.form["uname"],request.form["umobileno"],request.form["uworkstatus"],request
.form["uorganisation"])
            else:
                sql="INSERT INTO RECRUITER
VALUES('{}','{}','{}','{}','{}')".format(request.form["uemail"],request.form["upass"],r
equest.form["uname"],request.form["umobileno"],request.form["uorganisation"])
```

```
            ibm_db.exec_immediate(conn,sql)
            return render_template('index.html')
        except Exception as error:
            print(error)
            return render_template('register.html')
    else:
        return render_template('register.html')

#Seeker Login
@app.route("/login_seeker",methods=["GET","POST"])
def loginPageSeeker():
    if request.method=="POST":
        conn=connection()
        useremail=request.form["lemail"]
        password=request.form["lpass"]
        sql="SELECT COUNT(*) FROM SEEKER WHERE EMAIL=? AND PASSWORD=?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,useremail)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        res=ibm_db.fetch_assoc(stmt)
        if res['1']==1:
            session['loggedin']= True
            session['user'] = useremail
            return redirect(url_for('job_listing'))
        else:
            print("Wrong Username  or  Password")
            return render_template('loginseeker.html')
    else:
        return render_template('loginseeker.html')

#Recruiter Login
@app.route("/login_recruiter",methods=["GET","POST"])
def loginPageRecruiter():
    if request.method=="POST":
        conn=connection()
        useremail=request.form["lemail"]
        password=request.form["lpass"]
        sql="SELECT COUNT(*) FROM RECRUITER WHERE EMAIL=? AND PASSWORD=?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,useremail)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        res=ibm_db.fetch_assoc(stmt)
        if res['1']==1:
```

```
            session['loggedin']= True
            session['user'] = useremail
            return render_template("recruitermenu.html")
        else:
            print("Wrong Username or Password")
            return render_template('loginrecruiter.html')
    else:
        return render_template('loginrecruiter.html')

#Display Job Description
@app.route("/JobDescription",methods=["GET","POST"])
def JobDescPage():
    if request.method=="POST":
        conn=connection()
        try:
            sql="SELECT * FROM JOBS WHERE JOBID={}".format(request.form['jobidname'])
            #sql="SELECT * FROM JOBS WHERE JOBID=101" #should be replaced with the
jobid variable
            stmt = ibm_db.exec_immediate(conn,sql)
            dictionary = ibm_db.fetch_both(stmt)
            if dictionary != False:
                print ("COMPANY: ", dictionary["COMPANY"])
                print ("ROLE: ",  dictionary["ROLE"])
                print ("SALARY: ", dictionary["SALARY"])
                print ("LOCATION: ",  dictionary["LOCATION"])
                print ("JOBDESCRIPTION: ",  dictionary["JOBDESCRIPTION"])
                print ("POSTEDDATE: ",  dictionary["POSTEDDATE"])
                print ("APPLICATIONDEADLINE: ",  dictionary["APPLICATIONDEADLINE"])
                print ("JOBID: ",  dictionary["JOBID"])
                print ("JOBTYPE: ",  dictionary["JOBTYPE"])
                print ("EXPERIENCE: ", dictionary["EXPERIENCE"])
                print ("KEYSKILLS: ",  dictionary["KEYSKILLS"])
                print ("BENEFITSANDPERKS: ",  dictionary["BENEFITSANDPERKS"])
                print ("EDUCATION: ",  dictionary["EDUCATION"])
                print ("NOOFVACANCIES: ",  dictionary["NUMBEROFVACANCIES"])
                print ("DOMAIN: ",  dictionary["DOMAIN"])
                print ("RECRUITERMAIL: ", dictionary["RECRUITERMAIL"])
                fields=['JOBID','COMPANY','RECRUITER MAIL','ROLE','DOMAIN','JOB
TYPE','JOB DESCRIPTION','EDUCATION','KEY
SKILLS','EXPERIENCE','LOCATION','SALARY','BENEFITS AND PERKS','APPLICATION
DEADLINE','LOGO','NUMBER OF VACANCIES','POSTED DATE']
                today = date.today()
                if today > dictionary['APPLICATIONDEADLINE'] or
dictionary["NUMBEROFVACANCIES"]<=0:
                    disable=True
```

```
                else:
                    disable=False
                return
render_template('JobDescription.html',data=dictionary,fields=fields,disable=disable)
            else:
                print("INVALID JOB ID")
                return render_template('sample.html')
        except:
            print("SQL QUERY NOT EXECUTED")
            return render_template('sample.html')
    else:
        return render_template('sample.html')

#Apply Jobs
@app.route("/JobApplicationForm",methods=["GET","POST"])
def loadApplForm():
    if request.method=="POST":
        jobid=request.form["Applbutton"]
        print(jobid)
        return render_template('JobApplication.html',jobid=jobid)
    else:
        return render_template("sample.html")

#Apply Job Status Page
@app.route("/JobApplicationSubmit",methods=["GET","POST"])
def jobApplSubmit():
    if request.method=="POST":
        try:
            uploaded_file = request.files['uresume']
            if uploaded_file.filename != '':
                contents=uploaded_file.read()
                print(contents)
                try:
                    conn=connection()
                    sql="INSERT INTO APPLICATIONS
(JOBID,FIRSTNAME,LASTNAME,EMAILID,PHONENO,DOB,GENDER,PLACEOFBIRTH,CITIZENSHIP,PALINE1,P
ALINE2,PAZIPCODE,PACITY,PASTATE,PACOUNTRY,CURLINE1,CURLINE2,CURZIPCODE,CURCITY,CURSTATE
,CURCOUNTRY,XBOARD,XPERCENT,XYOP,XIIBOARD,XIIPERCENT,XIIYOP,GRADPERCENT,GRADYOP,MASTERS
PERCENT,MASTERSYOP,WORKEXPERIENCE,RESUME)
VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
                    stmt = ibm_db.prepare(conn, sql)
                    ibm_db.bind_param(stmt, 1, request.form["jobidname"])
                    ibm_db.bind_param(stmt, 2, request.form["ufname"])
                    ibm_db.bind_param(stmt, 3, request.form["ulname"])
                    ibm_db.bind_param(stmt, 4, request.form["uemail"])
```

```
                    ibm_db.bind_param(stmt, 5, request.form["uphone"])
                    ibm_db.bind_param(stmt, 6, request.form["udob"])
                    ibm_db.bind_param(stmt, 7, request.form["ugender"])
                    ibm_db.bind_param(stmt, 8, request.form["upob"])
                    ibm_db.bind_param(stmt, 9, request.form["uciti"])
                    ibm_db.bind_param(stmt, 10, request.form["pAL1"])
                    ibm_db.bind_param(stmt, 11, request.form["pAL2"])
                    ibm_db.bind_param(stmt, 12, request.form["pzip"])
                    ibm_db.bind_param(stmt, 13, request.form["pcity"])
                    ibm_db.bind_param(stmt, 14, request.form["pstate"])
                    ibm_db.bind_param(stmt, 15, request.form["pcntry"])
                    ibm_db.bind_param(stmt, 16, request.form["curAL1"])
                    ibm_db.bind_param(stmt, 17, request.form["curAL2"])
                    ibm_db.bind_param(stmt, 18, request.form["curzip"])
                    ibm_db.bind_param(stmt, 19, request.form["curcity"])
                    ibm_db.bind_param(stmt, 20, request.form["curstate"])
                    ibm_db.bind_param(stmt, 21, request.form["curcntry"])
                    ibm_db.bind_param(stmt, 22, request.form["Xboard"])
                    ibm_db.bind_param(stmt, 23, request.form["XPercent"])
                    ibm_db.bind_param(stmt, 24, request.form["XYOP"])
                    ibm_db.bind_param(stmt, 25, request.form["XIIboard"])
                    ibm_db.bind_param(stmt, 26, request.form["XIIPercent"])
                    ibm_db.bind_param(stmt, 27, request.form["XIIYOP"])
                    ibm_db.bind_param(stmt, 28, request.form["GradPercent"])
                    ibm_db.bind_param(stmt, 29, request.form["GradYOP"])
                    ibm_db.bind_param(stmt, 30, request.form["MastersPercent"])
                    ibm_db.bind_param(stmt, 31, request.form["MastersYOP"])
                    ibm_db.bind_param(stmt, 32, request.form["work"])
                    ibm_db.bind_param(stmt, 33, contents)
                    ibm_db.execute(stmt)
                    uemail=request.form["uemail"]

                    #REDUCE THE NO OF VACANCIES BY 1
                    sql2="UPDATE JOBS SET NUMBEROFVACANCIES = NUMBEROFVACANCIES-1 WHERE
JOBID='{}'".format(request.form["jobidname"])
                    stmt = ibm_db.exec_immediate(conn,sql2)
                    return render_template("JobApplicationSuccess.html",uemail=uemail)
                except:
                    print("SQL QUERY FAILED")
                    traceback.print_exc()
                    return render_template('sample.html')
        except:
            print("FILE UPLOAD FAILED")
            return render_template("sample.html")
    else:
```

```
        return render_template("sample.html")

#Download Resume
@app.route("/ResumeDownload",methods=["GET","POST"])
def downloadResume():
    if request.method=="POST":
        try:
            conn=connection()
            sql="SELECT * FROM APPLICATIONS WHERE
EMAILID='{}'".format(request.form["uemail"])
            stmt = ibm_db.exec_immediate(conn,sql)
            dictionary = ibm_db.fetch_both(stmt)
            return send_file(BytesIO(dictionary["RESUME"]),download_name="resume.pdf",
as_attachment=True)
        except:
            print("SELECT QUERY FAILED")
            traceback.print_exc()
            return render_template('sample.html')
    else:
        return render_template("sample.html")

#Recruiter Menu
@app.route('/recruitermenu', methods =["GET","POST"])
def recruitermenu():
    return render_template('recruitermenu.html')

#Post Job
@app.route('/postjob', methods =["GET","POST"])
def postjob():
    try:
        if request.method=="POST":
            conn=connection()

            sql1="SELECT ORGANISATION FROM RECRUITER WHERE EMAIL=?"
            stmt = ibm_db.prepare(conn, sql1)
            ibm_db.bind_param(stmt, 1, session['user'])
            ibm_db.execute(stmt)
            company = ibm_db.fetch_assoc(stmt)

            sql = "INSERT INTO JOBS(COMPANY, RECRUITERMAIL, ROLE, DOMAIN, JOBTYPE,
JOBDESCRIPTION, EDUCATION, KEYSKILLS, \
                EXPERIENCE, LOCATION, SALARY, BENEFITSANDPERKS, APPLICATIONDEADLINE,
LOGO, NUMBEROFVACANCIES, POSTEDDATE) \
                    values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)"
            stmt = ibm_db.prepare(conn, sql)
```

```
            ibm_db.bind_param(stmt, 1, list(company.values())[0])
            ibm_db.bind_param(stmt, 2, session['user'])
            ibm_db.bind_param(stmt, 3, request.form["role"])
            ibm_db.bind_param(stmt, 4, request.form["domain"])
            ibm_db.bind_param(stmt, 5, request.form["jobtype"])
            ibm_db.bind_param(stmt, 6, request.form["jobdes"])
            ibm_db.bind_param(stmt, 7, request.form["education"])
            ibm_db.bind_param(stmt, 8, request.form["skills"])
            ibm_db.bind_param(stmt, 9, request.form["experience"])
            ibm_db.bind_param(stmt, 10, request.form["location"])
            ibm_db.bind_param(stmt, 11, request.form["salary"])
            ibm_db.bind_param(stmt, 12, request.form["benefits"])
            ibm_db.bind_param(stmt, 13, request.form["deadline"])
            ibm_db.bind_param(stmt, 14, request.files["logo"].read())
            ibm_db.bind_param(stmt, 15, (int)(request.form["vacancies"]))
            ibm_db.bind_param(stmt, 16, date.today())
            ibm_db.execute(stmt)

            flash("Job Successfully Posted!")
            return render_template('recruitermenu.html')
        else:
            return render_template('postjob.html')
    except:
        traceback.print_exc()

if_name_=='_main_':
    app.config['SECRET_KEY']='super secret key'
    app.config['SESSION_TYPE']='filesystem'
    app.run(debug=True)
```

```
#dropbtn {

    border-color: #f03768;
    background: white;
    color: black;
    padding: 5px 20px;
```

```
    margin: 5px;
    min-width: 130px;

}

.dropdown  {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f1f1f1;
  min-width: 130px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}

.dropdown-content a {
  color:  black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
}

.dropdown-content a:hover {background-color: #ddd;}

.dropdown:hover .dropdown-content {display: block;}

.dropdown:hover #dropbtn {background-color: #f03768;}

.wrapper{
    width: 600px;
    border: 3px solid rgb(3, 3, 69);
    padding: 30px;
    margin: 50px;
}

.Header{
    text-align: center;
}

.frm
{
```

```
    display: block;
    text-align: center;
}
form
{
    display: inline-block;
    margin-left: auto;
    margin-right: auto;
    text-align: left;
}
.label{
    width : 150px;
}

.ans{
    width : 250px;
}
```

```
body{
    background-color:#FFF;
    font-family: "Lato", sans-serif;
    font-weight: 300;
    font-size:16px;
    line-height:18px;
    color:#777;
}

p{
    margin:0;
}

a {
    color: #aa0a5f;
}

a, a:hover, a:focus{
    text-decoration: none;
    outline: 0;
}
h1, h2, h3,
h4, h5, h6 {
    color: #333;
    margin:0;
```

```
    font-weight: normal;
}

h1 { font-size: 28px;}
h2 { font-size: 24px;}
h3 { font-size: 18px;}
h5 { font-size: 16px;}
h6 { font-size: 14px;}

strong {
  letter-spacing: 1px;
}

code {
  background-color: #ddd;
  border-radius: 3px;
  color: #000;
  font-size: 85%;
  margin: 0;
  padding: 5px 10px;
}

code a {
    color: #333;
}

code a:hover {
    text-decoration: underline;
}

section p {
  line-height: 28px;
}

.clearfix:before,
.clearfix:after {
    content: " ";
    display: table;
}
.clearfix:after {
    clear: both;
}
.clearfix {
    *zoom: 1;
}
```

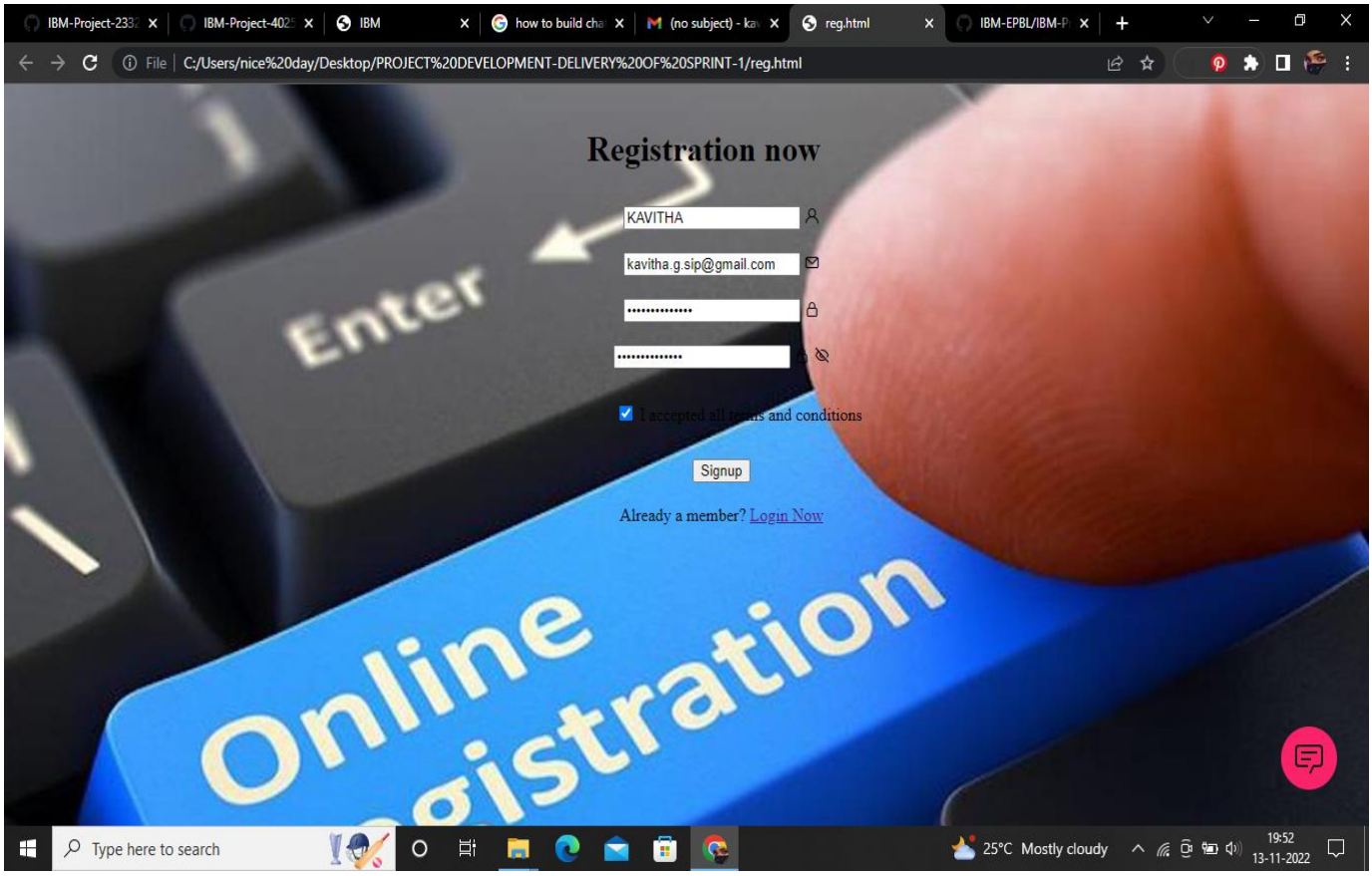
```
.tan {
  margin-bottom: 10px;
}

.fifteen{
    margin-bottom: 15px;
}

.twenty{
    margin-bottom: 20px;
}

.center {
    text-align: center;
}

.title {
    margin: 50px 0 30px;
}

.syntaxhighlighter {
    border: 1px solid #efefef;
    max-height: 100% !important;
    padding: 20px 0;
}


.main-content section {
  margin: 0 5%;
}


.left-sidebar {
    background-color: #ff2424;
    float: left;
    min-height: 100%;
    position: fixed;
    width: 18%;
}

.logo {
    padding-bottom: 30px;
    padding-left: 30px;
    padding-top: 70px;
```

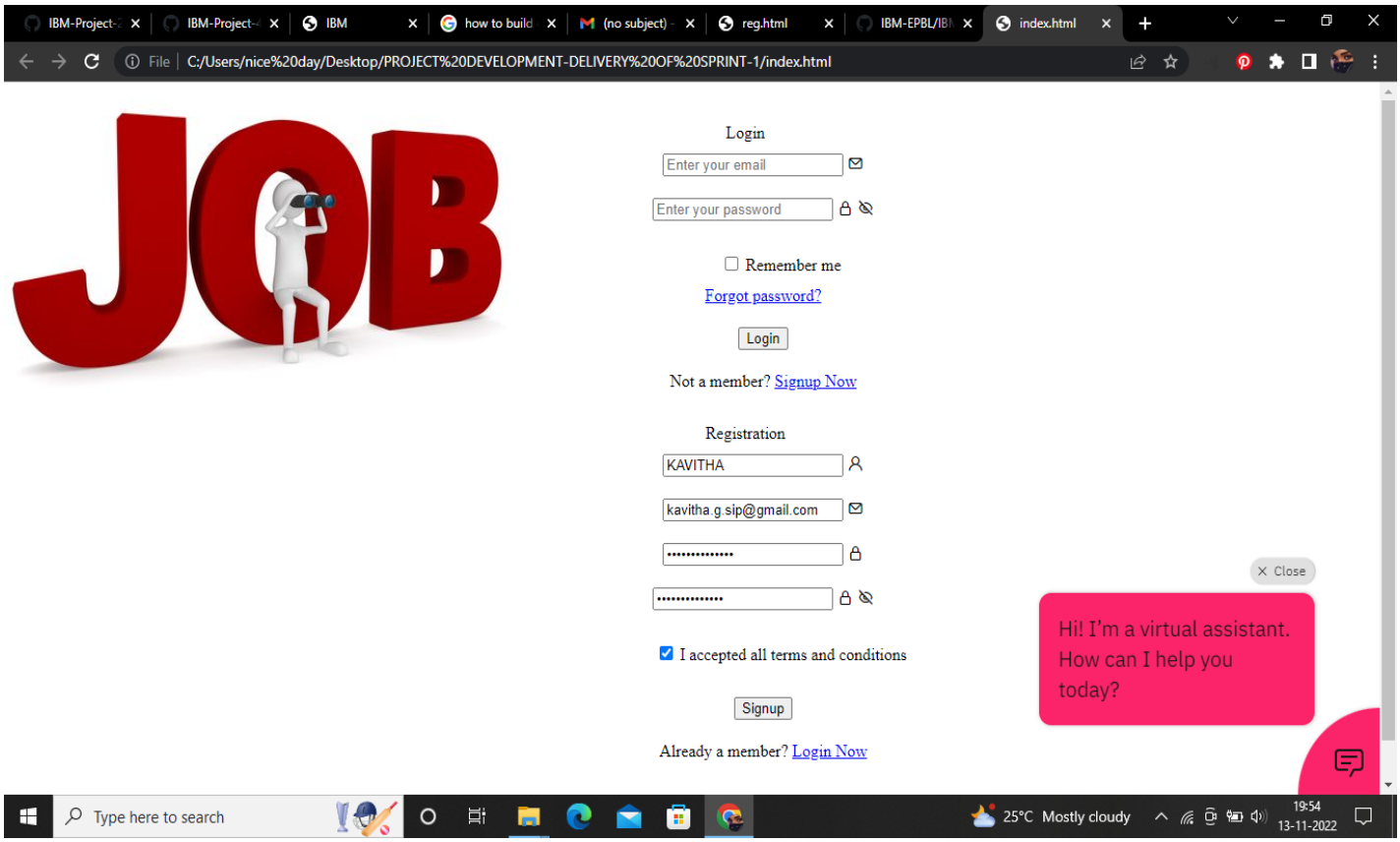
```
}
.logo h1{
    color: #fff;
    font-weight: 700;
    text-transform: uppercase;
}
.left-nav ul {
    margin: 0;
    padding: 0;
    font-size: 14px;
}

.left-nav ul li a {
    color: #fff;
    display: block;
    padding: 10px 35px;
    -webkit-transition: all 0.3s ease-in 0s;
       -moz-transition: all 0.3s ease-in 0s;
        -ms-transition: all 0.3s ease-in 0s;
         -o-transition: all 0.3s ease-in 0s;
            transition: all 0.3s ease-in 0s;
}

.left-nav ul li a:hover, .left-nav ul li .current {
    background-color: #fff;
    color: #000;
}


#main-wrapper {
    float: left;
    margin-left: 18%;
    width: 82%;
}

.content-header {
    border-bottom: 1px solid #ddd;
    border-top: 1px solid #ddd;
    margin-top: 30px;
    padding: 30px 0 35px;
    text-align: center;
}

.welcome {
    font-size: 16px;
```

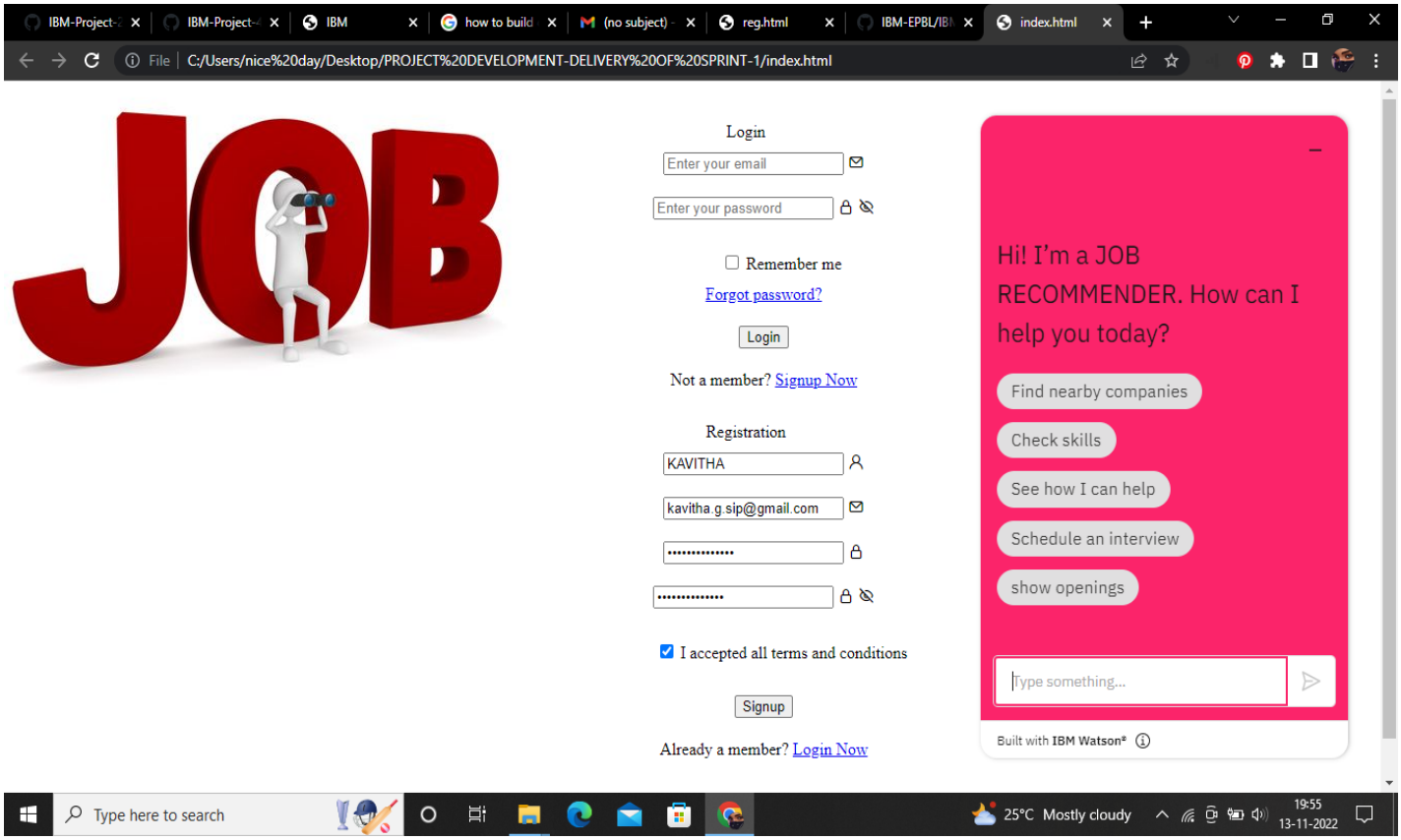
```
    line-height: 26px;
    margin: 35px auto 0;
}

.features {
    margin-top: 50px;
}

.features ul li {
    list-style: square outside none;
    margin-bottom: 15px;
}

.features > ul {
    padding-left: 18px;
}

.author {
    border-bottom: 1px solid #ddd;
    border-top: 1px solid #ddd;
    margin-top: 50px;
    padding: 30px 0;
}

.author-info {
    font-size: 18px;
    line-height: 28px;
    margin: 0 auto;
    width: 50%;
}

.section-content {
    font-size: 16px;
    line-height: 25px;
}

.section-content li {
    margin-bottom: 15px;
}

.section-content a:hover {
    text-decoration: underline;
}

.script-source li {
```

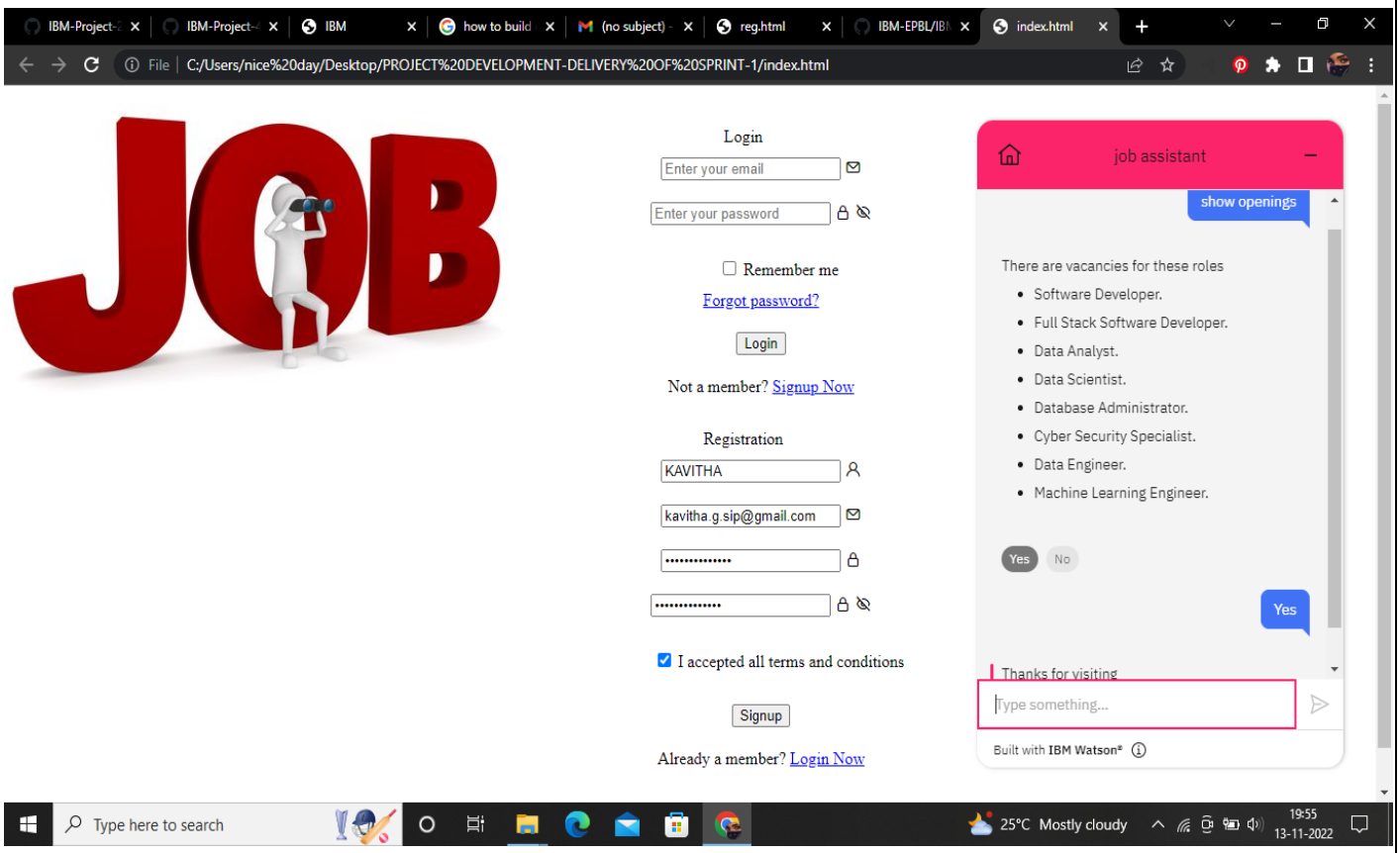
```
    list-style: square outside none;
    margin-bottom: 10px;
}

#twitter-feed li,
#flickr li {
    line-height: 25px;
    margin-bottom: 10px;
}

#twitter-feed  img  {
    border: 1px solid #ddd;
    box-shadow: 2px 3px 3px #ddd;
    height: auto;
    margin-top: 10px;
    max-width: 100%;
}
```

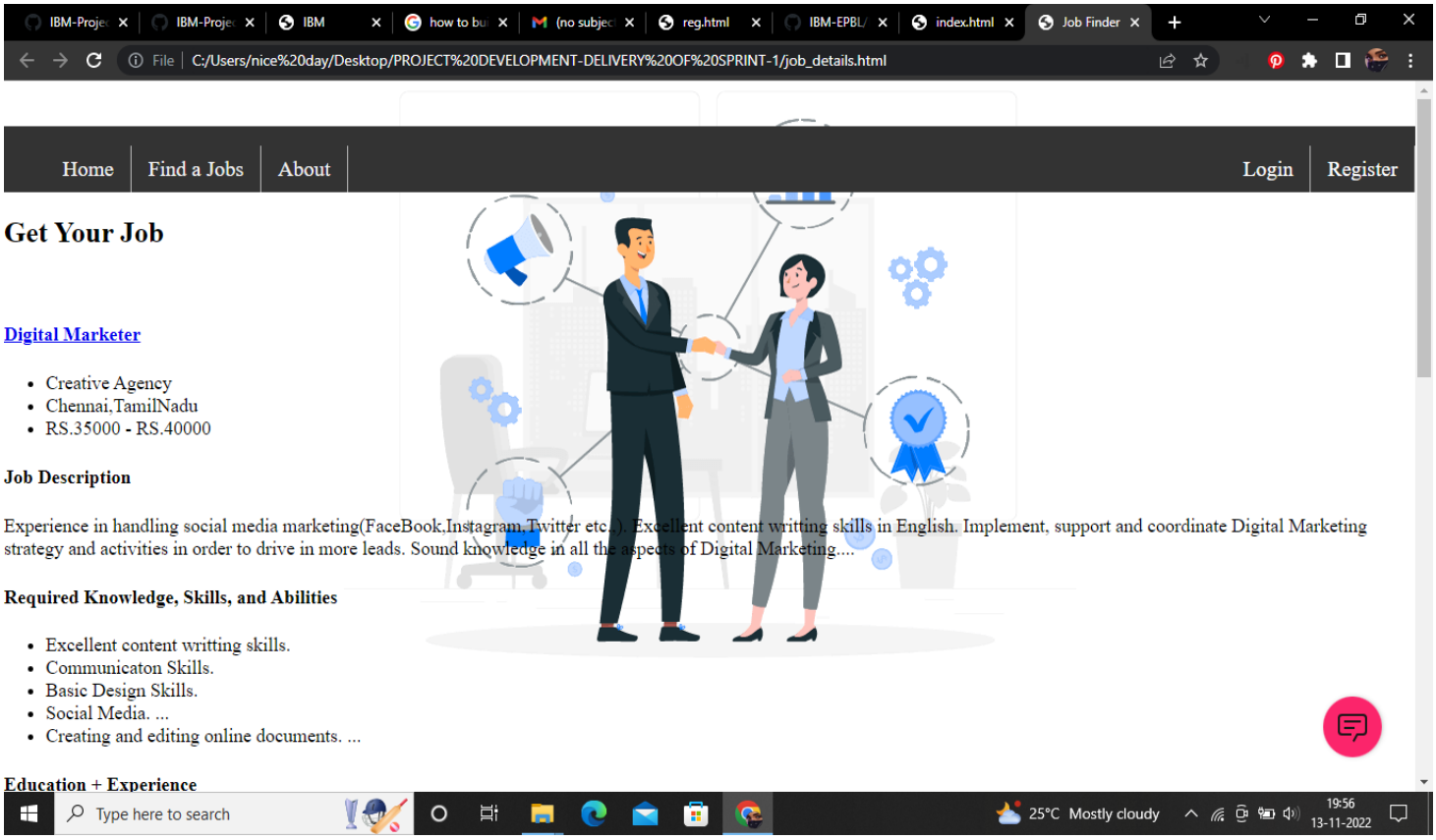
Screenshot

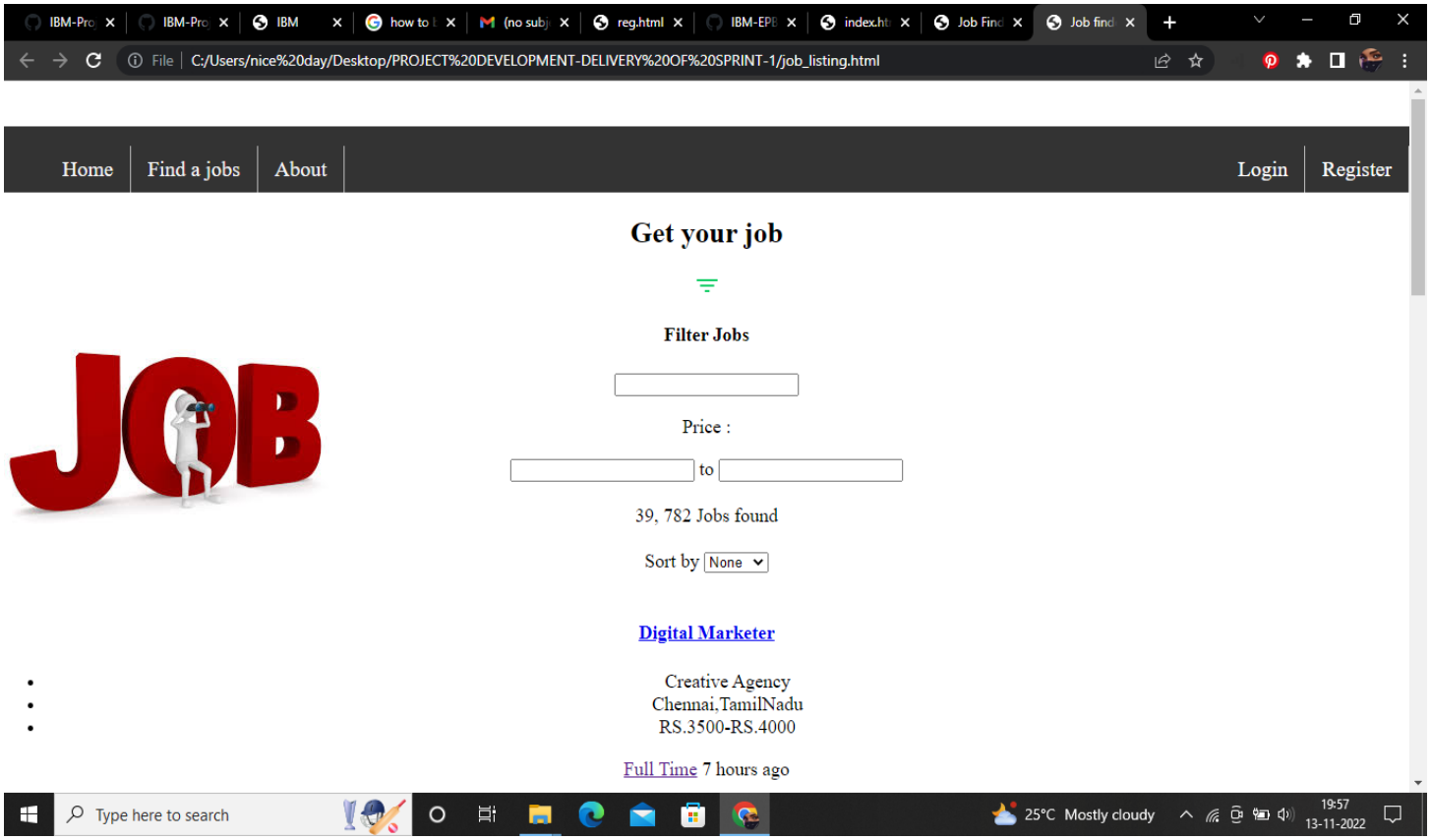

IBM-Project-2332
IBM-Project-4025
IBM
how to build cha
(no subject) - ka
reg.html
IBM-EPBL/IBM-P
File | C:/Users/nice%20day/Desktop/PROJECT%20DEVELOPMENT-DELIVERY%20OF%20SPRINT-1/reg.html
Registration now
KAVITHA
kavitha.g.sip@gmail.com
I accepted all terms and conditions
Signup
Already a member? Login Now
Enter
Online
Type here to search
25°C Mostly cloudy
19:52
13-11-2022


IBM-Project-
IBM-Project-
IBM
how to build
(no subject)
reg.html
IBM-EPBL/IBM
index.html
File
C:/Users/nice%20day/Desktop/PROJECT%20DEVELOPMENT-DELIVERY%20OF%20SPRINT-1/index.html
JOB
Login
Enter your email
Enter your password
Remember me
Forgot password?
Login
Not a member? Signup Now
Registration
KAVITHA
kavitha.g.sip@gmail.com
I accepted all terms and conditions
Signup
Already a member? Login Now
Close
Hi! I'm a virtual assistant. How can I help you today?
Type here to search
25°C Mostly cloudy
19:54
13-11-2022

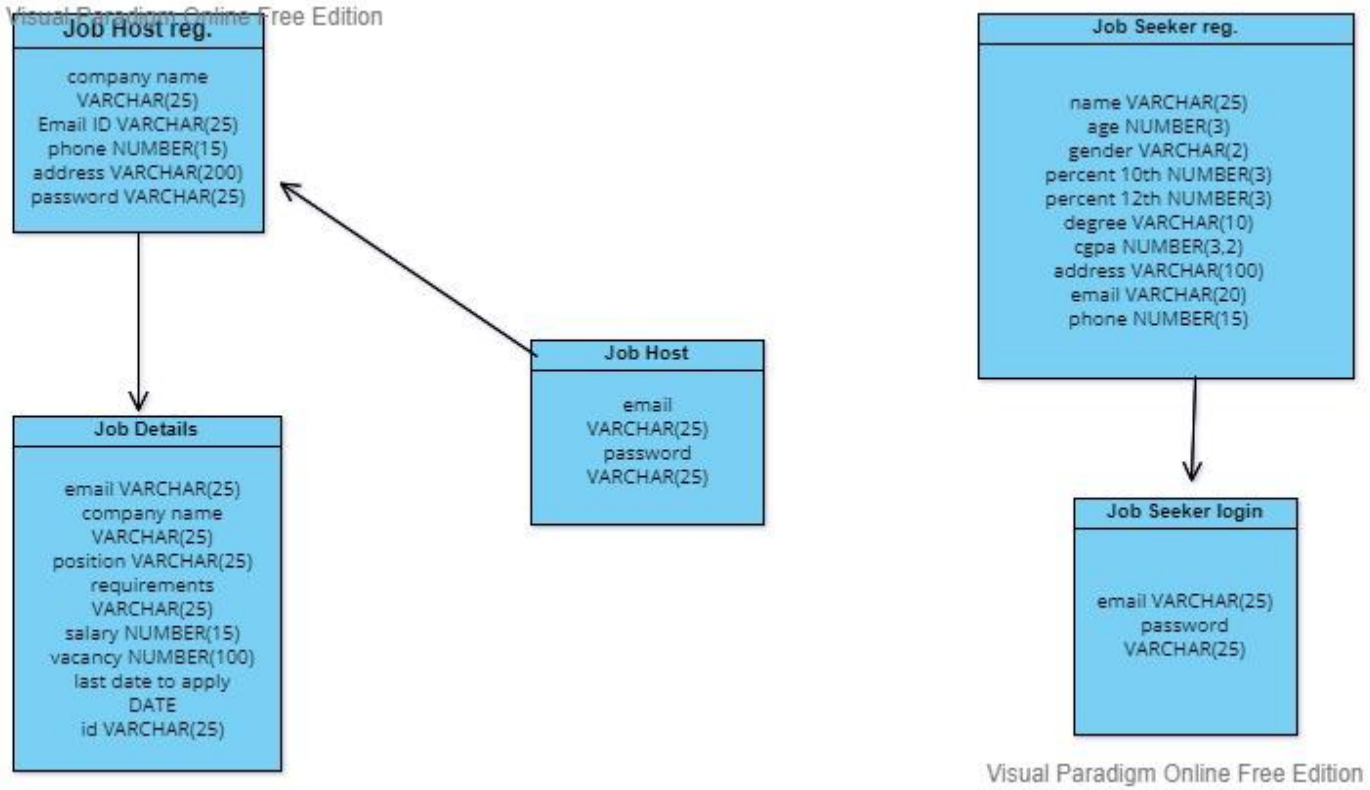

IBM-Project-
IBM-Project-
IBM
how to build
(no subject) -
reg.html
IBM-EPBL/IBM
index.html
File
C:/Users/nice%20day/Desktop/PROJECT%20DEVELOPMENT-DELIVERY%20OF%20SPRINT-1/index.html
JOB
Login
Enter your email
Enter your password
Remember me
Forgot password?
Login
Not a member? Signup Now
Registration
KAVITHA
kavitha.g.sip@gmail.com
I accepted all terms and conditions
Signup
Already a member? Login Now
Hi! I'm a JOB RECOMMENDER. How can I help you today?
Find nearby companies
Check skills
See how I can help
Schedule an interview
show openings
Type something...
Built with IBM Watson®
Type here to search
25°C Mostly cloudy
19:55
13-11-2022


IBM-Project-
IBM-Project-
IBM
how to build
(no subject) -
reg.html
IBM-EPBL/IB
index.html
File
C:/Users/nice%20day/Desktop/PROJECT%20DEVELOPMENT-DELIVERY%20OF%20SPRINT-1/index.html
JOB
Login
Enter your email
Enter your password
Remember me
Forgot password?
Login
Not a member? Signup Now
Registration
KAVITHA
kavitha.g.sip@gmail.com
I accepted all terms and conditions
Signup
Already a member? Login Now
job assistant
show openings
There are vacancies for these roles
Software Developer.
Full Stack Software Developer.
Data Analyst.
Data Scientist.
Database Administrator.
Cyber Security Specialist.
Data Engineer.
Machine Learning Engineer.
Yes
No
Yes
Thanks for visiting
Type something...
Built with IBM Watson®
Type here to search
25°C Mostly cloudy
19:55
13-11-2022


IBM-Proje
IBM-Proje
IBM
how to bu
(no subjec
reg.html
IBM-EPBL/
index.html
Job Finder
File | C:/Users/nice%20day/Desktop/PROJECT%20DEVELOPMENT-DELIVERY%20OF%20SPRINT-1/job_details.html
Home
Find a Jobs
About
Login
Register
Get Your Job
Digital Marketer
Creative Agency
Chennai,TamilNadu
RS.35000 - RS.40000
Job Description
Experience in handling social media marketing(FaceBook,Instagram,Twitter etc.). Excellent content writting skills in English. Implement, support and coordinate Digital Marketing strategy and activities in order to drive in more leads. Sound knowledge in all the aspects of Digital Marketing....
Required Knowledge, Skills, and Abilities
Excellent content writting skills.
Communicaton Skills.
Basic Design Skills.
Social Media. ...
Creating and editing online documents. ...
Education + Experience
Type here to search
25°C Mostly cloudy
19:56
13-11-2022


C:/Users/nice%20day/Desktop/PROJECT%20DEVELOPMENT-DELIVERY%20OF%20SPRINT-1/job_listing.html
Home
Find a jobs
About
Login
Register
Get your job
Filter Jobs
Price :
to
39, 782 Jobs found
Sort by None
Digital Marketer
Creative Agency
Chennai,TamilNadu
RS.3500-RS.4000
Full Time 7 hours ago
Type here to search
25°C Mostly cloudy
19:57
13-11-2022

DATABASE SCHEMA:



8 RESULTS

Performance Metrics

Based on the person-job fit premise, we propose a framework for job recommendation based on professional skills of job seekers. We automatically extracted the skills from the job seeker profiles using a variety of text processing techniques. Therefore, we perform the job recommendation using TF-IDF and four different configurations of Word2vec over a dataset of job seeker profiles and job vacancies collected by us. Our experimental results show the performances of the evaluated methods and configurations and can be used as a guide to choose the most suitable method and configuration for job recommendation.

7 ADVANTAGES & DISADVANTAGES

- Sourcing candidates requires a lot of effort, which means it can cost a company both time and money. It was found in one study that referred candidates are 55% faster to hire, compared with employees sourced through career sites. An advantage of employee referrals is that your current team member makes the connection and saves the recruiter that initial time of sourcing the candidate. Further, the candidate could be a better match compared to other candidates who apply externally. This will also help expedite the process and cut back on the need to find alternative options.
- Employees will want to work with someone who will improve their own output and day-to-day workload. So, in most cases, you can have more confidence in the candidate's ability to perform the necessary tasks. Further, according to research done by Zao, nearly three in ten employers have caught a fake reference on an application. So, a personal recommendation that is already within the company can instill confidence that the reference is in fact valid and reputable.
- After two years, retention of referred employees is 45% compared to 20% from job boards. Employee referrals tend to stay around longer, perhaps because they are personally connected to their peers. That's not to mention that the referrer themselves may feel more respected and valued too after their company takes their recommendation. And when an employee feels respected and valued, they can become more dedicated in turn. You may also want to give an employee referrer a bonus to show your appreciation.
- While in most cases an employee's motives should be "pure," there may be circumstances where a person wants to just work with their friend or receive the referral bonus. This can result in the candidate not being as qualified as either the referrer or referee said they were. The referrer may think that they can make up for the candidate's shortcomings or give them a crash course to level-set their skills. This can impact their own production in a negative way. And now your company may have two underperforming employees—and you may have to look to fill both of these positions in the not-so-far-off future.

8 CONCLUSION

We proposed a framework for job recommendation task. This framework facilitates the understanding of job recommendation process as well as it allows the use of a variety of text processing and recommendation methods according to the preferences of the job recommender system designer. Moreover, we also contribute making publicly available a new dataset containing job seekers profiles and job vacancies. Future directions of our work will focus on performing a more exhaustive evaluation considering a greater amount of methods and data as well as a comprehensive evaluation of the impact of each professional skill of a job seeker on the received job recommendation.

9 FUTURE SCOPE

For this system to be hybrid, content-based filtering is required, which can only recommend jobs based on the user's current profile. It cannot deliver anything surprising based on the user's past searches. This paper also uses collaborative filtering which faces well-known problems of privacy breaches and cold start. The system has a broad scope that can be used to make it more robust and foolproof. Firstly, automating the crawling process is required, when a new company is added to the database. In other words, removing the one-time configuration step/process to fetch jobs of a particular new company can be done. These models can implement techniques such as KNN in collaborative filtering. Implementing NLP in content-based filtering for better and more accurate search matching can be done. Along with this, testing and collecting more user data for better performance of the collaborative filtering module is required. Lastly, improving the cleansing process of the job description and using natural language processing are required. While using collaborative filtering, this work can be improved by giving different weights to different users based on their LinkedIn skills.

12 REFERECES

- https://ieeexplore.ieee.org/document/7944917
- https://www.researchgate.net/publication325697854Job_Recommendation_based_on_Job_Seeker_Skills_An_Empirical_Study
- https://www.quora.com/Linkedln
- https://ieeexplore.ieee.org/document/9752295

13 GITHUB ACCOUNT

https://github.com/IBM-EPBL/IBM-Project-17692-1659675303