1. INTRODUCTION

1.1.Project overview

This Project view provides an overview of the skill and job recommended for individuals interestedin a career in any fields. It discusses the important role that any field plays in businesses and the variousskills that are necessary for success in this field. It also outlines the different job opportunities availablein any field and the different types of companies thatemploy any field professionals.

1.2.Purpose

Having lots of skills but wondering which job willbest suit you ? Don’t need to worry! we have come up with a skill recommender solution through whichthe fresher or the skilled person can login and find the jobs by using search option or they can directly interact with the chatbot and get their dream job. To develop an end to end web application capable o displaying the current job openings based on the skillset of the users.The users and their information are stored in the Database. An alert is sent when there is an opening based on the user skillset. User will interact with the chatbot and can get the recommendations based on his skills.We can use job search API to get the current job openings in the market which will fetch the data directly from the webpage.

2. LITERATURE SURVEY

2.1.Existing problem

1. Students/Job seekers find their desired job based ontheir skillset.
2. Integrating Intelligent CHATBOT for job recommendation application.
3. A study of LinkedIn as an Employment Tool for Jobseeker & Recruiter.
4. Cloud storage and sharing services.

2.2. References

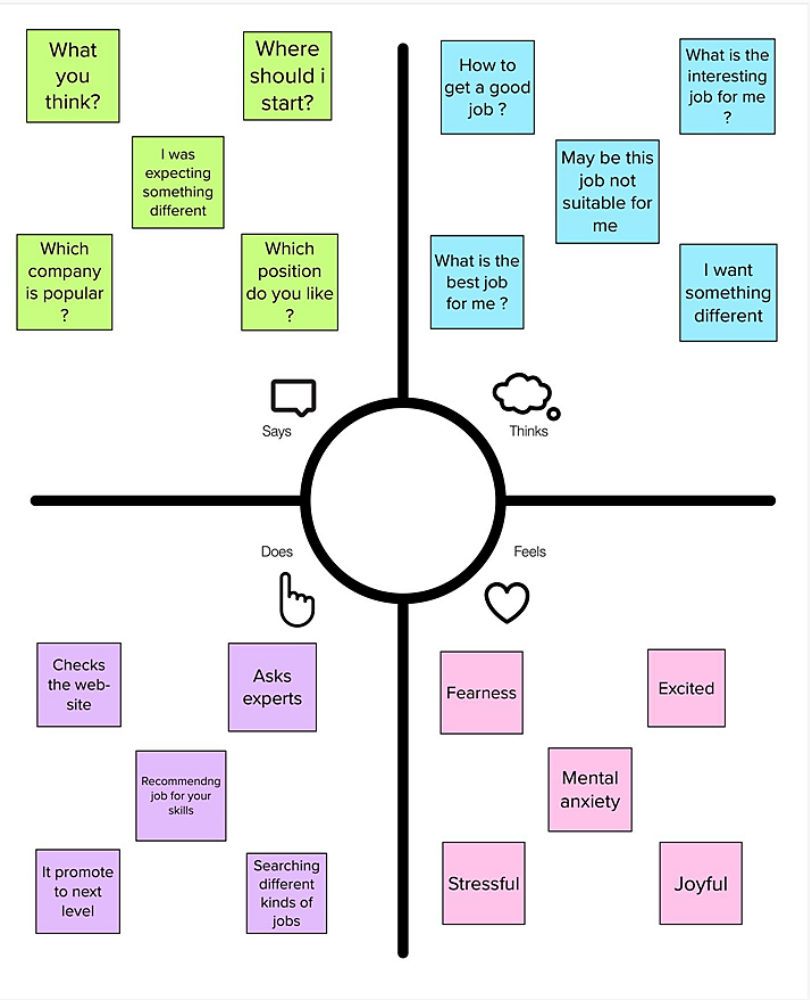
References link:

1. https://www.researchgate.net/publication/272802616_A_survey_of_job_recommender_sys tems
2. https://www.researchgate.net/publication/360820692_Intelligent_Chatbot
3. Journal homepage: http://www.ijrpr.com/ ISSN 2582-7421
4. https://www.ijresm.com/

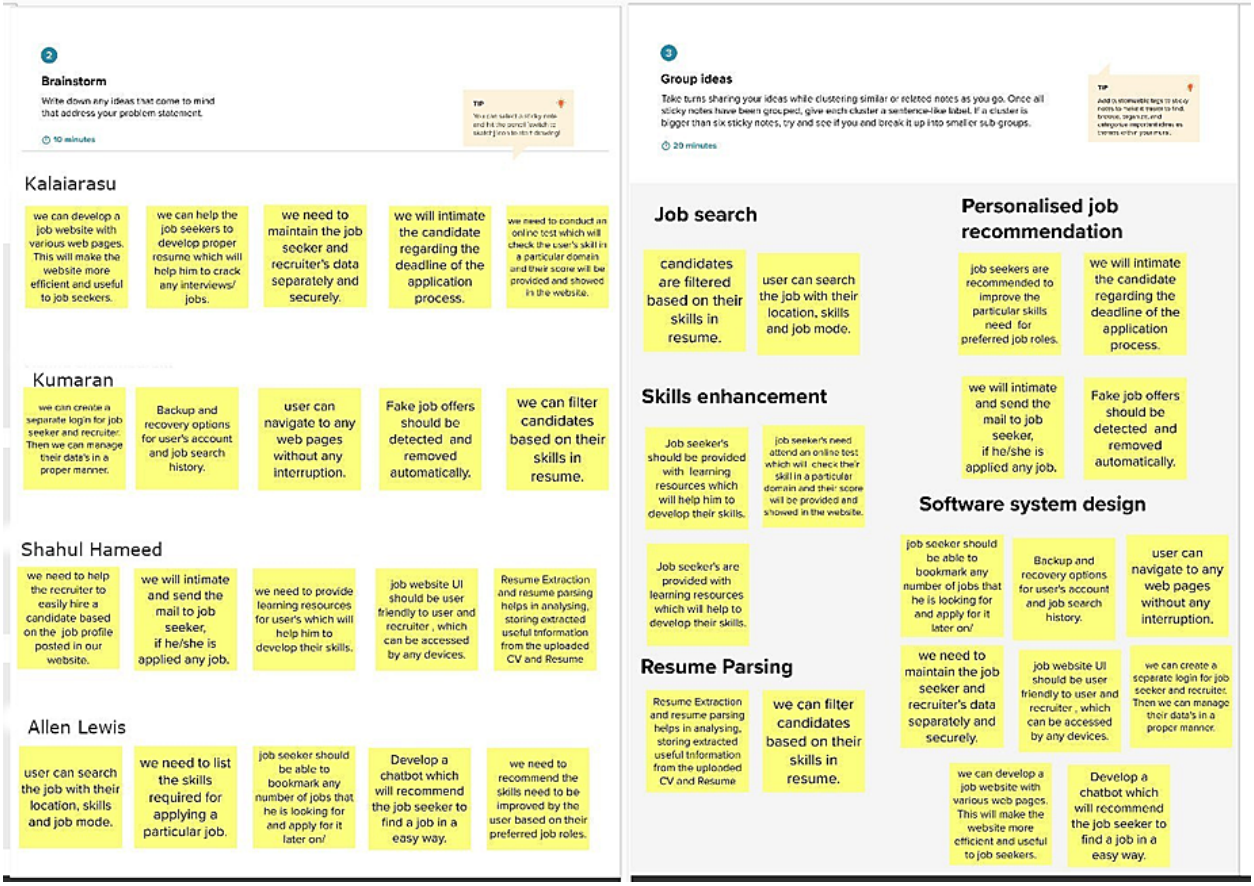
2.3. Problem Statement Definition Dealing with the enormous amount of recruiting information on the internet, a job seeker always spends hours to find useful ones. Many times, people who lack industry knowledge are unclear about what exactly they need to learn in order to get a suitable job for them. We address the problem of recommending suitable jobs to people who are seeking a new job. Job recommender technology aims to help job seekers in finding jobs that match their skills. The internet caused a substantial impact on the recruitment process through the creation of e-recruiting platforms that become a primary recruitment channel in most companies. While companies established job positions on these portals, job-seeker uses them to publishtheir profiles. E-Recruitment platforms accomplished clear advantages for both recruiters and job-seekers by reducing the recruitment time and advertisement cost .Recommender system technology aims to help users in finding items that match their preferences; it has a successful usage in a wide-range of applications to deal with problems related to information overload efficiently. In order to improve the e-recruiting functionality, many recommender system approaches have been proposed. This paper will analyze e-recruiting process and related issues for building personalized recommender system of candidates.

3. IDEATION & PROPOSED SOLUTION

3.1.Empathy Map Canvas



3.2.Ideation and Brainstroming



3.3.Proposed Solution

Team ID PNT2022TMID16689

PROJECT NAME SKILL & JOB RECOMMENDER

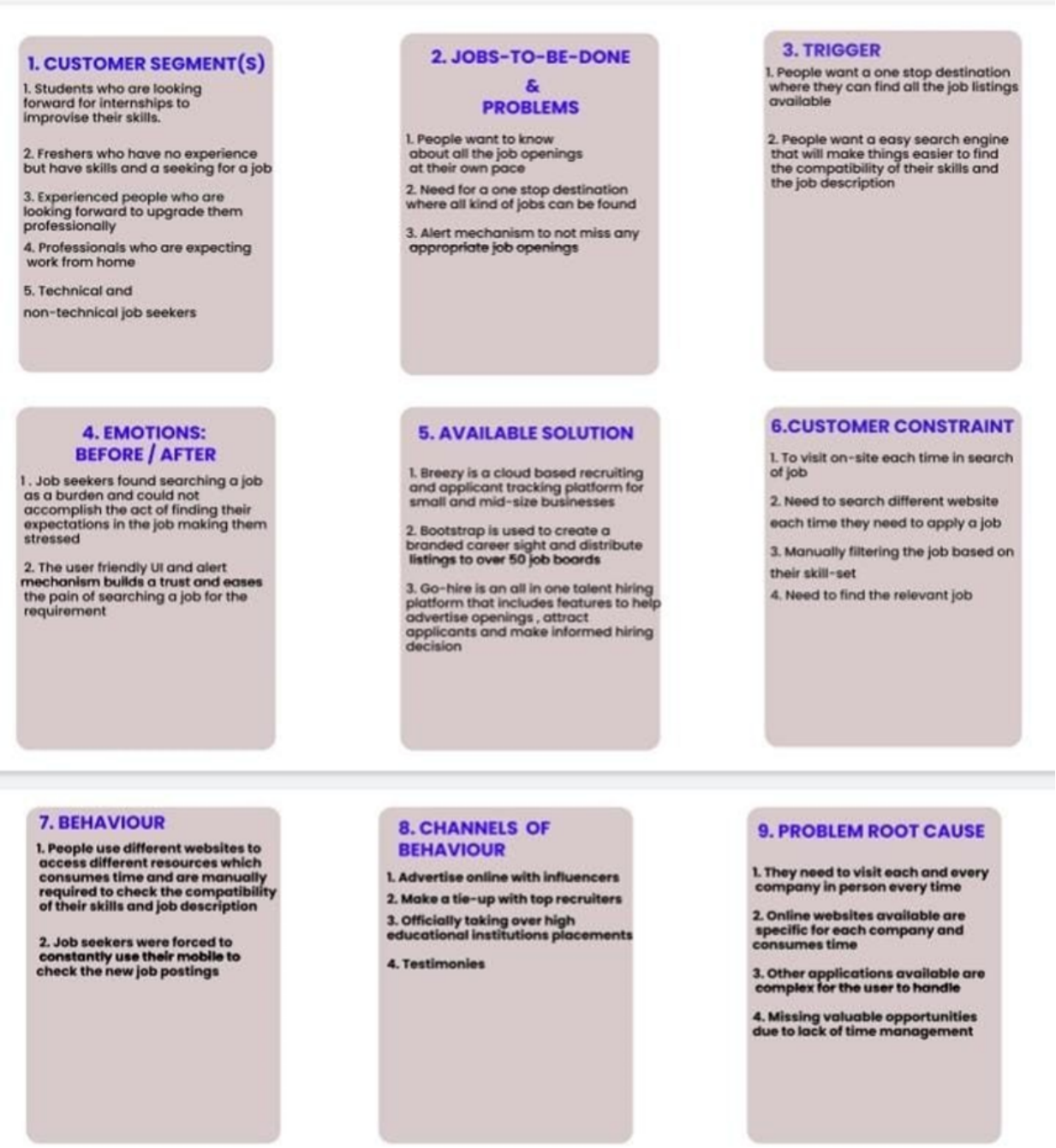
S. No	Parameter	Description
1	**Problem Statement (Problem to be solved)**	Nowadays a lot of students have great skills but unable to get a desired/appropriate job, so an end-to-end web application can be created which is capable of displayingcurrent job openings based on user skill set making it easier to hire and get hired.
2	**Idea/solution Description**	To develop an end-to-end web application which in default have a lot of current job openings through job search API out of which appropriate job will be recommended based on user skill set. At the

		sametime students can develop their skills side by side withvarious courses and webinars offered by reputed organization. In addition to this a smart chat bot will be available for 24*7 which can help users in finding the right job.
3	**Novelty/Uniqueness**	Though we have a lot of job searching applications, thisone is unique because, ❖ We have a smart chatbotbuilt with IBM Watson ❖ Our platform not only helps in getting job butalso helps in developingskills to get right job ❖ Here you can save/ bookmark jobs for

		later use and also turnon notification for company specific job alerts ❖ Add media files to your profile to showcase yourachievements ❖ It is made responsive toall screen sizes
4	**Social Impact / Customer Satisfaction**	Students will be benefited asthey will get to know which job suits them based on theirskill set and therefore Lack of Unemployment can be reduced.
5	**Business Model(Revenue Model)**	We can provide the applicationfor job seekers in a subscription based and we can share the profiles with companies and generate the revenue by providing them bestprofiles.
6	**Scalability of the Solution**	Data can be scaled up and scaled down

		according to number of current job openingsavailable.

3.4.Problem Solution Fit



4. REQUIREMENT ANALYSIS

4.1.Function Requirement

Software Required: Python, Flask, Docker

4.2.Non-Function Requirement

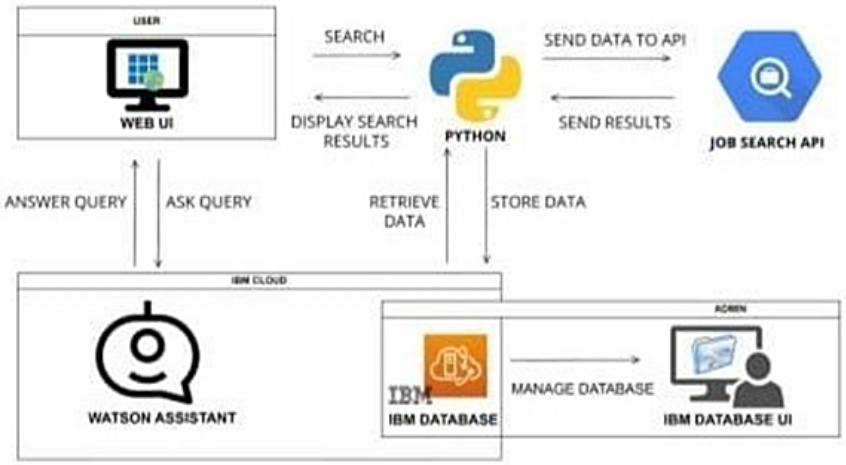
System Required: 8GB RAM, Intel Core i3, OS- Windows/Linux/MAC ,Laptop or Desktop

5. PROJECT DESIGN

5.1.Data Flow Diagrams

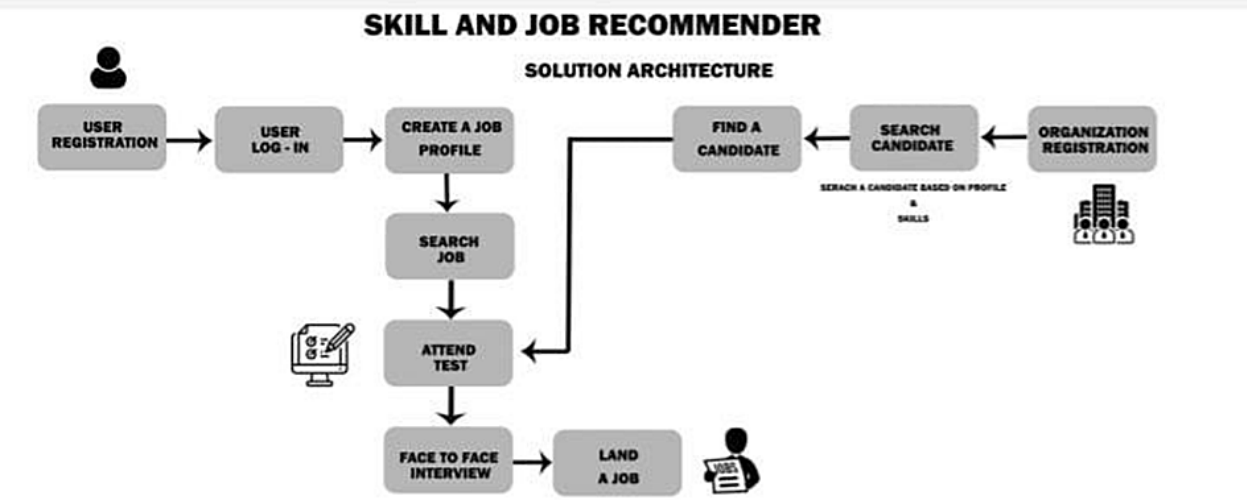
DATA FLOW DIAGRAM

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

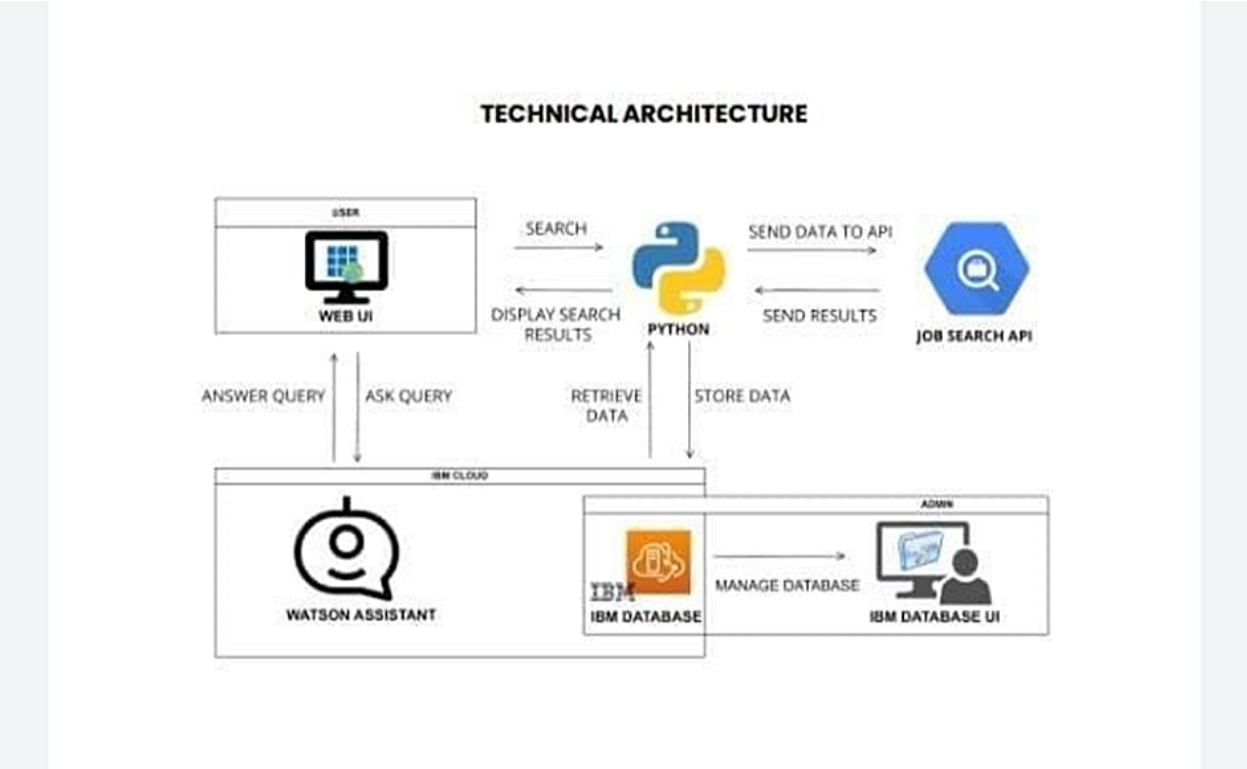


5.2.Solution & Technical Architecture

SOLUTION ARCHITECTURE



TECHNICAL ARCHITECTURE



5.3.User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
User of the Application	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-6	As a user they can enter all their information and register them	Their information is stored in database	High	Sprint-2
		USN-7	User should enter all the skills they posses		High	Sprint-2
Chat Bot		USN-8	User can interact and they can get replies to all their queries	AI bot is developed	High	Sprint-3
		USN-9	The bot requests for their skills and the job description that matches them are pulled		High	Sprint-3
		USN-10	User can find their applicable jobs for the skills they posses		High	Sprint-3
Administrator		USN-11	The jobs descriptions are stored		High	Sprint-4
		USN-12	The users queries are sorted	Queries are recognised	High	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1.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	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.2.Sprint Delivery Schedule VELOCITY:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let'scalculate the team's averagevelocity (AV) per iteration unit (story points per day)

7. CODING & SOLUTIONING

app.py:

```
from flask import Flask, render_template, g, flash,
request, redirect, url_for, session
import sqlite3
import functools
import os
from flask_mail import Mail, Message

app = Flask(__name__)
app.secret_key =
'5f21e03248d6309cfc8dae6b7f3682e22573017377f663d0'
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] =
os.environ.get('SENDGRID_API_KEY')
app.config['MAIL_DEFAULT_SENDER'] =
os.environ.get('MAIL_DEFAULT_SENDER')
mail = Mail(app)
```

```
16
17 DATABASE = 'db.db'
18
19 def loginRequired(view):
20  @functools.wraps(view)
21  def wrapped_view(**kwargs):
22       if session.get('user_id') is None:
23            return redirect(url_for("login"))
24       return view(**kwargs)
25  return wrapped_view
26
27 def adminRequired(view):
28  @functools.wraps(view)
29  def wrapped_view(**kwargs):
30       if not session.get('admin'):
31            return redirect(url_for("viewJobs"))
32       return view(**kwargs)
33  return wrapped_view
34
35 def nonAdminRequired(view):
36  @functools.wraps(view)
37  def wrapped_view(**kwargs):
38       if session.get('admin'):
39            return redirect(url_for("viewJobs"))
40       return view(**kwargs)
41  return wrapped_view
42
43 def getDb():
44  db = getattr(g, '_database', None)
45  if db is None:
46       db = g._database = sqlite3.connect(DATABASE)
47  return db
48
49 @app.teardown_appcontext
50 def closeConnection(exception):
51  db = getattr(g, '_database', None)
52  if db is not None:
```

```
53        db.close()
54
55 @app.route('/')
56 @app.route('/index/')
57 def index():
58  return redirect(url_for('login'))
59 #r
60
61 @app.route('/login/', methods=('GET', 'POST'))
62 def login():
63  return log(False)
64
65 @app.route('/login_admin/', methods=('GET', 'POST'))
66 def loginAdmin():
67  return log(True)
68
69 def log(admin):
70  error = None
71  if request.method == 'POST':
72        username = request.form['username']
73        password = request.form['password']
74        db = getDb()
75        if not username:
76             error = "Username is required"
77        elif not password:
78             error = "Password is required"
79        if error is None:
80             table = "useradmin" if admin else "user"
81             user = db.execute(f"SELECT id, password FROM
   {table} WHERE username=?",(username,)).fetchone()
82             if user is None or user[1] != password:
83                  error = 'Incorrect username or password'
84             else:
85                  session.clear()
86                  session['user_id'] = user[0]
87                  session['admin'] = admin
88                  session['username'] = username
```

```
                return redirect(url_for('viewJobs'))
 url = 'login_admin.html' if admin else 'login.html'
 return render_template(url, error = error, page = 1)

@app.route('/register/', methods=('GET', 'POST'))
def register():
 return reg(False)

@app.route('/register_admin/', methods=('GET', 'POST'))
def registerAdmin():
 return reg(True)

def reg(admin):
    error = None
    if request.method == 'POST':
         username = request.form['username']
         password = request.form['password']
         db = getDb()
         if not username:
              error = "Username is required"
         elif not password:
              error = "Password is required"
         if error is None:
              try:
                   table = "useradmin" if admin else
  "user"
                   db.execute(f"INSERT INTO
  {table}(username, password) VALUES(?, ?)", (username,
  password))
                   db.commit()
                   i = db.execute(f'select seq from
  sqlite_sequence where name="{table}"').fetchone()
                   session.clear()
                   session['user_id'] = i[0]
                   session['username'] = username
                   session['admin'] = admin
```

```
121                 except db.IntegrityError:
122                       error = f"User {username} is already
    registered"
123                 else:
124                       return redirect(url_for("viewJobs"))
125       url = 'register_admin.html' if admin else
    'register.html'
126       return render_template(url, error = error, page = 2)
127
128 @app.route('/view_jobs/', methods=('GET', 'POST'))
129 @loginRequired
130 def viewJobs():
131       applied = False
132       if request.method == 'POST':
133             db = getDb()
134             db.execute("INSERT INTO jobapplied (uid, jid)
    VALUES (?, ?)", (session['user_id'],
    request.form['job_id']))
135             db.commit()
136             jc = db.execute("SELECT job, company FROM job
    where id=?", (request.form['job_id'],)).fetchone()
137             applied = True
138             msg = Message('Confirmation of job application',
    recipients=[session['username']],
    sender='1923001@saec.ac.in')
139             msg.html = 'This is to confirm that you have
    successfully applied for the role of ' + jc[0] + ' at ' +
    jc[1]
140             mail.send(msg)
141       db = getDb()
142       jobs = db.execute('SELECT id, company, job, domain,
    salary FROM job').fetchall()
143       session['jobs'] = jobs
144       return render_template('view_jobs.html', applied =
    applied, page = 3)
145
```

```
146 @app.route('/add_jobs/', methods=('GET', 'POST'))
147 @loginRequired
148 @adminRequired
149 def addJobs():
150     if request.method == 'POST':
151         db = getDb()
152         db.execute('INSERT INTO job (company, job,
  domain, salary) VALUES (?, ?, ?, ?)', (session['username'],
  request.form['job'], request.form['domain'],
  request.form['salary']))
153         db.commit()
154         return redirect(url_for('viewJobs'))
155     return render_template('add_jobs.html')
156
157 @app.route('/applied_jobs/')
158 @loginRequired
159 @nonAdminRequired
160 def appliedJobs():
161     db = getDb()
162     jids = db.execute('SELECT jid FROM jobapplied WHERE
  uid=' + str(session['user_id'])).fetchall()
163     jobs = []
164     for jid in jids:
165         jobs.append(db.execute('SELECT id, company, job,
  domain, salary FROM job WHERE id=' +
  str(jid[0])).fetchone())
166     session['jobs'] = jobs
167     return render_template('applied_jobs.html', page = 4)
168
169 @app.route('/signout/')
170 def signout():
171     session.clear()
172     return redirect(url_for('index'))
173
174 def initDb():
175     with app.app_context():
```

```
176          db = getDb()
177          with app.open_resource('schema.sql', mode='r') as
   f:
178              db.cursor().executescript(f.read())
179          db.commit()
180
181 #initDb()
182
183 app.run()
```

schema.sql:

```
1  DROP TABLE IF EXISTS user;
2  CREATE TABLE user (
3    id INTEGER PRIMARY KEY AUTOINCREMENT,
4    username TEXT UNIQUE NOT NULL,
5    password TEXT NOT NULL
6  );
7
8  DROP TABLE IF EXISTS useradmin;
9  CREATE TABLE useradmin (
10   id INTEGER PRIMARY KEY AUTOINCREMENT,
11   username TEXT UNIQUE NOT NULL,
12   password TEXT NOT NULL
13 );
14
15 DROP TABLE IF EXISTS job;
16 CREATE TABLE job (
17   id INTEGER PRIMARY KEY AUTOINCREMENT,
18   company TEXT,
19   job TEXT,
20   domain TEXT,
21   salary TEXT
22 );
23
24 DROP TABLE IF EXISTS jobapplied;
```

```
CREATE TABLE jobapplied (
 id INTEGER PRIMARY KEY AUTOINCREMENT,
 uid INTEGER,
 jid INTEGER
);
```

style.css:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
}

li a, .dropbtn {
  display: inline-block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover:not(.active), .dropdown:hover .dropbtn {
  background-color: #111;
}

li.dropdown {
  display: inline-block;
}

```

```
.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 2);
  z-index: 1;
}

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

.dropdown-content a:hover {
  background-color: #f1f1f1;
}

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

.active {
  background-color: #04AA6D;
}

.dropdown:hover .active {
  background-color: #15BB7E;
}

button {
  font-size: 20px;
  border-radius: 6px;
  border: none;
```

```
 background: #333;
 color: white;
 margin: 5px;
}

.input_block {
 margin: 5px;
}
.input_block label {
 display: block;
 font-size: 20px;
}
.input_block input {
 width: 93%;
 padding: 0.625rem 10px;
 margin-bottom: 1.875rem;
 border: 1px solid #ccc;
 border-radius: 4px;
 font-size: 1rem;
}

.box {
 position: absolute;
 top: 50%;
 left: 50%;
 transform: translate(-50%, -50%);
 padding: 2.5rem;
 box-sizing: border-box;
 border: 1px solid #dadce0;
 border-radius: 8px;
 display: box;
}

body {
    margin: 0;
    padding: 0;
```

```
}

form {
    display: block;
    margin-top: 0em;
}

button {
    border: none;
    outline: none;
    color: #fff;
    background-color: #1a73e8;
    padding: 0.625rem 1.25rem;
    border-radius: 0.312rem;
    font-size: 1rem;
    float: right;
}

table.applyTable {
    position: absolute;
    width: 50%;
    margin-top: 50px;
    left: 25%;
    text-align: center;
    border-collapse: collapse;
}

th.applyTh {
    width: 20%;
    padding: 10px;
    border: thin solid black;
}

td.applyTd {
    width: 20%;
    padding: 10px;
    border: thin solid black;
```

```
139 }
140
141 tr.addRow td {
142     border: none;
143 }
```

add_jobs.html:

```
1  {% extends 'base.html' %}
2
3  {% block title %}Add Job {% endblock %}
4
5  {% block content %}
6   <div class="box">
7        <form method="post">
8             <h2 style="text-align:center">Add Job</h2>
9             <div class="input_block">
10                 <input name="job" placeholder="Job Title"
   required>
11            </div>
12            <div class="input_block">
13                 <input name="domain" placeholder="Domain"
   required>
14            </div>
15            <div class="input_block">
16                 <input name="salary"  placeholder="Salary"
   required>
17            </div>
18            <button type="submit">Add</button>
19       </form>
20  </div>
21 {% endblock %}
```

applied_jobs.html:

```
{% extends 'base.html' %}

{% block title %}Jobs Applied {% endblock %}

{% block content %}
 {% if applied %}
      <script>
           alert("Applied successfully");
      </script>
 {% endif %}
 <table class="applyTable">
      <tr>
           <th class="applyTh">Company</th>
           <th class="applyTh">Job Title</th>
           <th class="applyTh">Domain</th>
           <th class="applyTh">Salary</th>
      </tr>
      {% for job in session['jobs'] %}
           <tr>
                {% for i in range(1, 5) %}
                     <td class="applyTd">{{job[i]}}</td>
                {% endfor %}
           </tr>
      {% endfor %}
 </table>
{% endblock %}
```

base.html:

```
<!DOCTYPE html>
<script>
window.onpageshow = function(event){
 i
      w
 }
};
```

```
</script>
<html>
 <head>
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
      <title>{% block title %}{% endblock %} - Job
Recommender System</title>
      <link rel="stylesheet" href={{url_for("static",
filename="style.css")}}>
 </head>
 <body>
      <header>
           <ul>
<!--                  <li
endif %}><a href={{url_for("index")}}>Home</a></li>-->
                <li class="dropdown {% if page==1
%}active{% endif %}">
                     <a href="javascript:void(0)"
class="dropbtn">Login</a>
                     <div class="dropdown-content">
                          <a
href={{url_for("login")}}>Applicant</a>
                          <a
href={{url_for("loginAdmin")}}>Admin</a>
                     </div>
                </li>
                <li class="dropdown {% if page==2
%}active{% endif %}">
                     <a href="javascript:void(0)"
class="dropbtn">Register</a>
                     <div class="dropdown-content">
                          <a
href={{url_for("register")}}>Applicant</a>
                          <a
href={{url_for("registerAdmin")}}>Admin</a>
                     </div>
```

```
                </li>
                <li {% if page==3 %}class="active"{% endif %}><a href={{url_for("viewJobs")}}>View Jobs</a></li>
                <li {% if page==4 %}class="active"{% endif %}><a href={{url_for("appliedJobs")}}>Applied Jobs</a></li>
                {
                    <li style="float:right"><a href={{url_for("signout")}}>Sign out</a></li>
                {
            </ul>
        </header>
        <section class="content">
            {
        </section>
 </body>
</html>
```

index.html:

```
{% extends 'base.html' %}

{% block title %}Home {% endblock %}

{% block content %}

{% endblock %}
```

login.html:

```
{% extends 'base.html' %}

{% block title %}Login {% endblock %}

{% block content %}
 <script>
        {% if error is not none %}
```

```
            alert("{{error}}");
        {% endif %}
 </script>
 <div class="box">
        <form method="post">
            <h2 style="text-align:center">Login</h2>
            <div class="input_block">
                <input type='email' name="username"
placeholder="Username" required><br>
            </div>
            <div class="input_block">
                <input name="password" type="password"
placeholder="Password" required><br>
            </div>
            <button type="submit"
name="submit">Login</button>
        </form>
 </div>
{% endblock %}
```

login_admin.html:

```
{% extends 'base.html' %}

{% block title %}Login - Admin {% endblock %}

{% block content %}
 <script>
        {% if error is not none %}
            alert("{{error}}");
        {% endif %}
 </script>
 <div class="box">
        <form method="post">
            <h2 style="text-align:center">Login(Admin)</h2>
            <div class="input_block">
```

```
15                <input type='email' name="username"
placeholder="Username" required><br>
16            </div>
17            <div class="input_block">
18                <input name="password" type="password"
placeholder="Password" required><br>
19            </div>
20            <button type="submit"
name="submit">Login</button>
21       </form>
22  </div>
23 {% endblock %}
```

register.html:

```
1  {% extends 'base.html' %}
2
3  {% block title %}Register {% endblock %}
4
5  {% block content %}
6   <script>
7        {% if error is not none %}
8            alert("{{error}}");
9        {% endif %}
10       function validate() {
11           if(document.registerForm.password.value !=
document.registerForm.confirmPassword.value) {
12               alert("Passwords don't match");
13               return false;
14           }
15           return true;
16       }
17  </script>
18  <div class="box">
19       <form method="post" name="registerForm"
onsubmit="return validate()">
```

```
            <h2 style="text-align:center">Register</h2>
            <div class="input_block">
                 <input type='email' name="username"
placeholder="E-mail" required><br>
            </div>
            <div class="input_block">
                 <input name="password" type="password"
placeholder="Password" required><br>
            </div>
            <div class="input_block">
                 <input name="confirmPassword"
type="password" placeholder="Confirm Password"
required><br>
            </div>
            <button type="submit"
name="submit">Register</button>
       </form>
  </div>
{% endblock %}
```

register_admin.html:

```
{% extends 'base.html' %}

{% block title %}Register - Admin {% endblock %}

{% block content %}
  <script>
       {% if error is not none %}
            alert("{{error}}");
       {% endif %}
       function validate() {
            if(document.registerForm.password.value !=
document.registerForm.confirmPassword.value) {
                 alert("Passwords don't match");
                 return false;
```

```
            }
            return true;
        }
 </script>
 <div class="box">
        <form method="post" name="registerForm"
onsubmit="return validate()">
            <h2 style="text-
align:center">Register(Admin)</h2>
            <div class="input_block">
                <input type='email' name="username"
placeholder="E-mail" required><br>
            </div>
            <div class="input_block">
                <input name="password" type="password"
placeholder="Password" required><br>
            </div>
            <div class="input_block">
                <input name="confirmPassword"
type="password" placeholder="Confirm Password"
required><br>
            </div>
            <button type="submit"
name="submit">Register</button>
        </form>
 </div>
{% endblock %}
```

view_jobs.html:

```
{% extends 'base.html' %}

{% block title %}View Jobs {% endblock %}

{% block content %}
  {% if applied %}
```

```
7         <script>
8              alert("Applied successfully");
9         </script>
10   {% endif %}
11   <table class="applyTable">
12        <tr>
13             <th class="applyTh">Company</th>
14             <th class="applyTh">Job Title</th>
15             <th class="applyTh">Domain</th>
16             <th class="applyTh">Salary</th>
17             {% if not session['admin'] %}
18                  <th class="applyTh"></th>
19             {% endif %}
20        </tr>
21        {% for job in session['jobs'] %}
22             <tr>
23                  {% for i in range(1, 5) %}
24                       <td class="applyTd">{{job[i]}}</td>
25                  {% endfor %}
26                  {% if not session['admin'] %}
27                       <td class="applyTd">
28                            <form method='post'>
29                                 <input type="hidden"
   name="job_id" value="{{job[0]}}">
30                                 <button
   type="submit">Apply</button>
31                            </form>
32                       </td>
33                  {% endif %}
34             </tr>
35        {% endfor %}
36        {% if session['admin'] is true %}
37             <tr class="addRow">
38                  <td class="applyTd"></td>
39                  <td class="applyTd"></td>
40                  <td class="applyTd">
41                       <input type="button" value="Add"
```

```
   onclick="window.location='{{url_for("addJobs")}}'">
42                 </td>
43                 <td class="applyTd"></td>
44                 <td class="applyTd"></td>
45             </tr>
46         {% endif %}
47   </table>
48 {% endblock %}
```

9. RESULTS

The project has been completed as we expected. We ensured that Database was designed and well connected to our project. The Expected results were gotten

10. ADVANTAGES & DISADVANTAGE

ADVANTAGES:

1. Person who looks for a job can easily find a suitable job based on their skill set.
2. Person can check their eligibility by attending eligibility test.
3. Most of the Recruiters find the suitable person based on the scores they have gotten in the eligibility.

DISADVANTAGES

1. Person Job May get technical difficulty while taking the eligibility
2. Job seeker may have trouble to contact recruiters directly.

11. CONCLUSION

The application has been developed to make job search easier . The application that we have developed is user friendly . User can find a job based on their skillset in the short period of time. The jobseeker certainly get benefit by using this application. In the addition,Chatbot Has been implemented with the help of IBM whatson . The chatbot helps jobseeker and organization when they experience the difficulties.

12. FUTURE SCOPE

The linked in the wellknown application to find a job and stay connected with professional and organization. The job seekers and organization use linked in to find a job. In the future , There are lots of possibilities to enhance our project similar to linked in

13 Appendix

app.py:

```
1  from flask import Flask, render_template, g, flash,
   request, redirect, url_for, session
2  import sqlite3
3  import functools
4  import os
5  from flask_mail import Mail, Message
6
7  app = Flask(__name__)
8  app.secret_key =
   '5f21e03248d6309cfc8dae6b7f3682e22573017377f663d0'
9  app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
10 app.config['MAIL_PORT'] = 587
11 app.config['MAIL_USE_TLS'] = True
12 app.config['MAIL_USERNAME'] = 'apikey'
13 app.config['MAIL_PASSWORD'] =
   os.environ.get('SENDGRID_API_KEY')
14 app.config['MAIL_DEFAULT_SENDER'] =
   os.environ.get('MAIL_DEFAULT_SENDER')
15 mail = Mail(app)
16
17 DATABASE = 'db.db'
18
19 def loginRequired(view):
20  @functools.wraps(view)
21  def wrapped_view(**kwargs):
22       if session.get('user_id') is None:
23            return redirect(url_for("login"))
24       return view(**kwargs)
25  return wrapped_view
26
27 def adminRequired(view):
28  @functools.wraps(view)
```

```
29  def wrapped_view(**kwargs):
30       if not session.get('admin'):
31            return redirect(url_for("viewJobs"))
32       return view(**kwargs)
33  return wrapped_view
34
35 def nonAdminRequired(view):
36  @functools.wraps(view)
37  def wrapped_view(**kwargs):
38       if session.get('admin'):
39            return redirect(url_for("viewJobs"))
40       return view(**kwargs)
41  return wrapped_view
42
43 def getDb():
44  db = getattr(g, '_database', None)
45  if db is None:
46       db = g._database = sqlite3.connect(DATABASE)
47  return db
48
49 @app.teardown_appcontext
50 def closeConnection(exception):
51  db = getattr(g, '_database', None)
52  if db is not None:
53       db.close()
54
55 @app.route('/')
56 @app.route('/index/')
57 def index():
58  return redirect(url_for('login'))
59 #r
60
61 @app.route('/login/', methods=('GET', 'POST'))
62 def login():
63  return log(False)
64
65 @app.route('/login_admin/', methods=('GET', 'POST'))
```

```python
def loginAdmin():
 return log(True)

def log(admin):
 error = None
 if request.method == 'POST':
      username = request.form['username']
      password = request.form['password']
      db = getDb()
      if not username:
           error = "Username is required"
      elif not password:
           error = "Password is required"
      if error is None:
           table = "useradmin" if admin else "user"
           user = db.execute(f"SELECT id, password FROM {table} WHERE username=?",(username,)).fetchone()
           if user is None or user[1] != password:
                error = 'Incorrect username or password'
           else:
                session.clear()
                session['user_id'] = user[0]
                session['admin'] = admin
                session['username'] = username
                return redirect(url_for('viewJobs'))
 url = 'login_admin.html' if admin else 'login.html'
 return render_template(url, error = error, page = 1)

@app.route('/register/', methods=('GET', 'POST'))
def register():
 return reg(False)

@app.route('/register_admin/', methods=('GET', 'POST'))
def registerAdmin():
 return reg(True)

def reg(admin):
```

```
102     error = None
103     if request.method == 'POST':
104          username = request.form['username']
105          password = request.form['password']
106          db = getDb()
107          if not username:
108               error = "Username is required"
109          elif not password:
110               error = "Password is required"
111          if error is None:
112               try:
113                    table = "useradmin" if admin else
   "user"
114                    db.execute(f"INSERT INTO
   {table}(username, password) VALUES(?, ?)", (username,
   password))
115                    db.commit()
116                    i = db.execute(f'select seq from
   sqlite_sequence where name="{table}"').fetchone()
117                    session.clear()
118                    session['user_id'] = i[0]
119                    session['username'] = username
120                    session['admin'] = admin
121               except db.IntegrityError:
122                    error = f"User {username} is already
   registered"
123               else:
124                    return redirect(url_for("viewJobs"))
125     url = 'register_admin.html' if admin else
   'register.html'
126     return render_template(url, error = error, page = 2)
127
128 @app.route('/view_jobs/', methods=('GET', 'POST'))
129 @loginRequired
130 def viewJobs():
131     applied = False
```

```
132     if request.method == 'POST':
133         db = getDb()
134         db.execute("INSERT INTO jobapplied (uid, jid)
  VALUES (?, ?)", (session['user_id'],
  request.form['job_id']))
135         db.commit()
136         jc = db.execute("SELECT job, company FROM job
  where id=?", (request.form['job_id'],)).fetchone()
137         applied = True
138         msg = Message('Confirmation of job application',
  recipients=[session['username']],
  sender='1923001@saec.ac.in')
139         msg.html = 'This is to confirm that you have
  successfully applied for the role of ' + jc[0] + ' at ' +
  jc[1]
140         mail.send(msg)
141     db = getDb()
142     jobs = db.execute('SELECT id, company, job, domain,
  salary FROM job').fetchall()
143     session['jobs'] = jobs
144     return render_template('view_jobs.html', applied =
  applied, page = 3)
145
146 @app.route('/add_jobs/', methods=('GET', 'POST'))
147 @loginRequired
148 @adminRequired
149 def addJobs():
150     if request.method == 'POST':
151         db = getDb()
152         db.execute('INSERT INTO job (company, job,
  domain, salary) VALUES (?, ?, ?, ?)', (session['username'],
  request.form['job'], request.form['domain'],
  request.form['salary']))
153         db.commit()
154         return redirect(url_for('viewJobs'))
155     return render_template('add_jobs.html')
```

```
156
157 @app.route('/applied_jobs/')
158 @loginRequired
159 @nonAdminRequired
160 def appliedJobs():
161     db = getDb()
162     jids = db.execute('SELECT jid FROM jobapplied WHERE
  uid=' + str(session['user_id'])).fetchall()
163     jobs = []
164     for jid in jids:
165         jobs.append(db.execute('SELECT id, company, job,
  domain, salary FROM job WHERE id=' +
  str(jid[0])).fetchone())
166     session['jobs'] = jobs
167     return render_template('applied_jobs.html', page = 4)
168
169 @app.route('/signout/')
170 def signout():
171     session.clear()
172     return redirect(url_for('index'))
173
174 def initDb():
175     with app.app_context():
176         db = getDb()
177         with app.open_resource('schema.sql', mode='r') as
  f:
178             db.cursor().executescript(f.read())
179         db.commit()
180
181 #initDb()
182
183 app.run()
```

schema.sql:

```
1  DROP TABLE IF EXISTS user;
```

```
CREATE TABLE user (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT UNIQUE NOT NULL,
  password TEXT NOT NULL
);

DROP TABLE IF EXISTS useradmin;
CREATE TABLE useradmin (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT UNIQUE NOT NULL,
  password TEXT NOT NULL
);

DROP TABLE IF EXISTS job;
CREATE TABLE job (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  company TEXT,
  job TEXT,
  domain TEXT,
  salary TEXT
);

DROP TABLE IF EXISTS jobapplied;
CREATE TABLE jobapplied (
 id INTEGER PRIMARY KEY AUTOINCREMENT,
 uid INTEGER,
 jid INTEGER
);
```

style.css:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
```

```
  background-color: #333;
}

li {
  float: left;
}

li a, .dropbtn {
  display: inline-block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover:not(.active), .dropdown:hover .dropbtn {
  background-color: #111;
}

li.dropdown {
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 2);
  z-index: 1;
}

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

```
  text-align: left;
}

.dropdown-content a:hover {
  background-color: #f1f1f1;
}

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

.active {
  background-color: #04AA6D;
}

.dropdown:hover .active {
  background-color: #15BB7E;
}

button {
  font-size: 20px;
  border-radius: 6px;
  border: none;
  background: #333;
  color: white;
  margin: 5px;
}

.input_block {
  margin: 5px;
}
.input_block label {
  display: block;
  font-size: 20px;
}
.input_block input {
  width: 93%;
```

```
  padding: 0.625rem 10px;
  margin-bottom: 1.875rem;
  border: 1px solid #ccc;
  border-radius: 4px;
  font-size: 1rem;
}

.box {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  padding: 2.5rem;
  box-sizing: border-box;
  border: 1px solid #dadce0;
  border-radius: 8px;
  display: box;
}

body {
    margin: 0;
    padding: 0;
}

form {
    display: block;
    margin-top: 0em;
}

button {
    border: none;
    outline: none;
    color: #fff;
    background-color: #1a73e8;
    padding: 0.625rem 1.25rem;
    border-radius: 0.312rem;
    font-size: 1rem;
```

```
    float: right;
}

table.applyTable {
    position: absolute;
    width: 50%;
    margin-top: 50px;
    left: 25%;
    text-align: center;
    border-collapse: collapse;
}

th.applyTh {
    width: 20%;
    padding: 10px;
    border: thin solid black;
}

td.applyTd {
    width: 20%;
    padding: 10px;
    border: thin solid black;
}

tr.addRow td {
    border: none;
}
```

add_jobs.html:

```
{% extends 'base.html' %}

{% block title %}Add Job {% endblock %}

{% block content %}
  <div class="box">
```

```
7         <form method="post">
8              <h2 style="text-align:center">Add Job</h2>
9              <div class="input_block">
10                  <input name="job" placeholder="Job Title"
   required>
11             </div>
12             <div class="input_block">
13                  <input name="domain" placeholder="Domain"
   required>
14             </div>
15             <div class="input_block">
16                  <input name="salary"  placeholder="Salary"
   required>
17             </div>
18             <button type="submit">Add</button>
19        </form>
20   </div>
21 {% endblock %}
```

applied_jobs.html:

```
1  {% extends 'base.html' %}
2
3  {% block title %}Jobs Applied {% endblock %}
4
5  {% block content %}
6    {% if applied %}
7         <script>
8              alert("Applied successfully");
9         </script>
10   {% endif %}
11   <table class="applyTable">
12        <tr>
13             <th class="applyTh">Company</th>
14             <th class="applyTh">Job Title</th>
15             <th class="applyTh">Domain</th>
```

```
            <th class="applyTh">Salary</th>
        </tr>
        {% for job in session['jobs'] %}
            <tr>
                {% for i in range(1, 5) %}
                    <td class="applyTd">{{job[i]}}</td>
                {% endfor %}
            </tr>
        {% endfor %}
  </table>
{% endblock %}
```

base.html:

```
<!DOCTYPE html>
<script>
window.onpageshow = function(event){
  i
        w
  }
};
</script>
<html>
  <head>
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
        <title>{% block title %}{% endblock %} - Job
Recommender System</title>
        <link rel="stylesheet" href={{url_for("static",
filename="style.css")}}>
  </head>
  <body>
        <header>
            <ul>
<!--                    <li
endif %}><a href={{url_for("index")}}>Home</a></li>-->
```

```
19                  <li class="dropdown {% if page==1
  %}active{% endif %}">
20                       <a href="javascript:void(0)"
  class="dropbtn">Login</a>
21                       <div class="dropdown-content">
22                            <a
  href={{url_for("login")}}>Applicant</a>
23                            <a
  href={{url_for("loginAdmin")}}>Admin</a>
24                       </div>
25                  </li>
26                  <li class="dropdown {% if page==2
  %}active{% endif %}">
27                       <a href="javascript:void(0)"
  class="dropbtn">Register</a>
28                       <div class="dropdown-content">
29                            <a
  href={{url_for("register")}}>Applicant</a>
30                            <a
  href={{url_for("registerAdmin")}}>Admin</a>
31                       </div>
32                  </li>
33                  <li {% if page==3 %}class="active"{% endif
  %}><a href={{url_for("viewJobs")}}>View Jobs</a></li>
34                  <li {% if page==4 %}class="active"{% endif
  %}><a href={{url_for("appliedJobs")}}>Applied Jobs</a></li>
35                  {
36                       <li style="float:right"><a
  href={{url_for("signout")}}>Sign out</a></li>
37                  {
38             </ul>
39        </header>
40        <section class="content">
41             {
42        </section>
43   </body>
```

```
</html>
```

index.html:

```
{% extends 'base.html' %}

{% block title %}Home {% endblock %}

{% block content %}

{% endblock %}
```

login.html:

```
{% extends 'base.html' %}

{% block title %}Login {% endblock %}

{% block content %}
 <script>
      {% if error is not none %}
           alert("{{error}}");
      {% endif %}
 </script>
 <div class="box">
      <form method="post">
           <h2 style="text-align:center">Login</h2>
           <div class="input_block">
                <input type='email' name="username"
placeholder="Username" required><br>
           </div>
           <div class="input_block">
                <input name="password" type="password"
placeholder="Password" required><br>
           </div>
           <button type="submit"
```

```
   name="submit">Login</button>
21        </form>
22   </div>
23 {% endblock %}
```

login_admin.html:

```
1  {% extends 'base.html' %}
2
3  {% block title %}Login - Admin {% endblock %}
4
5  {% block content %}
6    <script>
7         {% if error is not none %}
8              alert("{{error}}");
9         {% endif %}
10   </script>
11   <div class="box">
12        <form method="post">
13             <h2 style="text-align:center">Login(Admin)</h2>
14             <div class="input_block">
15                  <input type='email' name="username"
   placeholder="Username" required><br>
16             </div>
17             <div class="input_block">
18                  <input name="password" type="password"
   placeholder="Password" required><br>
19             </div>
20             <button type="submit"
   name="submit">Login</button>
21        </form>
22   </div>
23 {% endblock %}
```

register.html:

```
{% extends 'base.html' %}

{% block title %}Register {% endblock %}

{% block content %}
 <script>
      {% if error is not none %}
           alert("{{error}}");
      {% endif %}
      function validate() {
           if(document.registerForm.password.value != document.registerForm.confirmPassword.value) {
                alert("Passwords don't match");
                return false;
           }
           return true;
      }
 </script>
 <div class="box">
      <form method="post" name="registerForm" onsubmit="return validate()">
           <h2 style="text-align:center">Register</h2>
           <div class="input_block">
                <input type='email' name="username" placeholder="E-mail" required><br>
           </div>
           <div class="input_block">
                <input name="password" type="password" placeholder="Password" required><br>
           </div>
           <div class="input_block">
                <input name="confirmPassword" type="password" placeholder="Confirm Password" required><br>
           </div>
           <button type="submit" name="submit">Register</button>
```

```
        </form>
    </div>
{% endblock %}
```

register_admin.html:

```
{% extends 'base.html' %}

{% block title %}Register - Admin {% endblock %}

{% block content %}
    <script>
        {% if error is not none %}
            alert("{{error}}");
        {% endif %}
        function validate() {
            if(document.registerForm.password.value != document.registerForm.confirmPassword.value) {
                alert("Passwords don't match");
                return false;
            }
            return true;
        }
    </script>
    <div class="box">
        <form method="post" name="registerForm" onsubmit="return validate()">
            <h2 style="text-align:center">Register(Admin)</h2>
            <div class="input_block">
                <input type='email' name="username" placeholder="E-mail" required><br>
            </div>
            <div class="input_block">
                <input name="password" type="password" placeholder="Password" required><br>
```

```
            </div>
            <div class="input_block">
                  <input name="confirmPassword"
type="password" placeholder="Confirm Password"
required><br>
            </div>
            <button type="submit"
name="submit">Register</button>
      </form>
 </div>
{% endblock %}
```

view_jobs.html:

```
{% extends 'base.html' %}

{% block title %}View Jobs {% endblock %}

{% block content %}
 {% if applied %}
      <script>
            alert("Applied successfully");
      </script>
 {% endif %}
 <table class="applyTable">
      <tr>
            <th class="applyTh">Company</th>
            <th class="applyTh">Job Title</th>
            <th class="applyTh">Domain</th>
            <th class="applyTh">Salary</th>
            {% if not session['admin'] %}
                  <th class="applyTh"></th>
            {% endif %}
      </tr>
      {% for job in session['jobs'] %}
            <tr>
```

```
                {% for i in range(1, 5) %}
                    <td class="applyTd">{{job[i]}}</td>
                {% endfor %}
                {% if not session['admin'] %}
                    <td class="applyTd">
                        <form method='post'>
                            <input type="hidden"
name="job_id" value="{{job[0]}}">
                            <button
type="submit">Apply</button>
                        </form>
                    </td>
                {% endif %}
            </tr>
        {% endfor %}
        {% if session['admin'] is true %}
            <tr class="addRow">
                <td class="applyTd"></td>
                <td class="applyTd"></td>
                <td class="applyTd">
                    <input type="button" value="Add"
onclick="window.location='{{url_for("addJobs")}}'">
                </td>
                <td class="applyTd"></td>
                <td class="applyTd"></td>
            </tr>
        {% endif %}
 </table>
{% endblock %}
```

14 GITHUB & PROJECT DEMO LINK:

All the tasks of developing the application were uploaded on the github. The github has been uploaded below

Github: https://github.com/IBM-EPBL/IBM-Project-15848-1659605380

Demo:
https://drive.google.com/file/d/18CSen9lO3rd_SpT8wbRdCvIIPP_C2L2A/view?usp=share_link