

SPRINT 4

TEAM ID	PNT2022TMID22144
PROJECT NAME	Skill Based Job Recommender

FINAL DELIVERY:

CODE:

SOURCE CODE(HTML)

```
% extends 'layouts/base-presentation.html' %}

{% block title %} Presentation {% endblock title %}

<!-- Specific CSS goes HERE -->
{% block stylesheets %}{% endblock stylesheets %}

{% block body_class %} index-page {% endblock body_class %}

{% block content %}

    <header class="header-2">
        <div class="page-header section-height-75 relative" style="background-
image: url('/static/assets/img/curved-images/curved.jpg')">
            <div class="container">
                <div class="row">
                    <div class="col-lg-7 text-center mx-auto">
                        <h1 class="text-white pt-3 mt-n5">
                            <a class="text-white" target="_blank"
                                href="">Job Recommender</a>
                        </h1>
                        <p class="lead text-white mt-3">
                            recommends using closest near neighbours
                            <br />
                            <a class="text-white" target="_blank" href=""
                                target="_blank"></a>.
                        </p>
                    </div>
                </div>
            </div>
            <div class="position-absolute w-100 z-index-1 bottom-0">
                <svg class="waves" xmlns="http://www.w3.org/2000/svg"
                    xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 24 150 40"
                    preserveAspectRatio="none" shape-rendering="auto">
```

```

        <defs>
            <path id="gentle-wave" d="M-160 44c30 0 58-18 88-18s 58 18 88 18
58-18 88-18 58 18 88 18 v44h-352z" />
        </defs>
        <g class="moving-waves">
            <use xlink:href="#gentle-wave" x="48" y="-1"
fill="rgba(255,255,255,0.40)" />
            <use xlink:href="#gentle-wave" x="48" y="3"
fill="rgba(255,255,255,0.35)" />
            <use xlink:href="#gentle-wave" x="48" y="5"
fill="rgba(255,255,255,0.25)" />
            <use xlink:href="#gentle-wave" x="48" y="8"
fill="rgba(255,255,255,0.20)" />
            <use xlink:href="#gentle-wave" x="48" y="13"
fill="rgba(255,255,255,0.15)" />
            <use xlink:href="#gentle-wave" x="48" y="16"
fill="rgba(255,255,255,0.95)" />
        </g>
    </svg>
</div>
</div>
</header>

<script>
    window.watsonAssistantChatOptions = {
        integrationID: "0bb96b92-4e98-44c7-9dab-3a5fe2ff8562", // The ID of this
integration.
        region: "au-syd", // The region your integration is hosted in.
        serviceInstanceID: "e5babddc-2ad5-4eac-a0c5-6dad126622cb", // The ID of
your service instance.
        onLoad: function(instance) { instance.render(); }
    };
    setTimeout(function(){
        const t=document.createElement('script');
        t.src="https://web-
chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
        document.head.appendChild(t);
    });
</script>

{% for items in product %}

<section class="my-5 py-5">
    <div class="container">
        <div class="row align-items-center">
            <div class="col-lg-4 ms-auto me-auto p-lg-4 mt-lg-0 mt-4">

```

```

        <div class="card card-background " data-tilt>
            <div class="full-background" style="background-image:
url('https://images.unsplash.com/photo-1567095761054-7a02e69e5c43?ixlib=rb-
4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w
=687&q=80')"></div>
            <div class="card-body pt-7 text-center">
                <h2 class="text-white up mb-0">{{product[items].Position}} <br
/> {{product[items].Company}} <br> {{product[items].location}}</h2>
                <a href="{{product[items].url}}" class="btn btn-outline-white
mt-5 up btn-round">Apply</a>
            </div>
        </div>
    </div>
</div>
</section>

{% endfor%}

{% endblock content %}

<!-- Specific JS goes HERE -->
{% block javascripts %}

    <script src="/static/assets/js/plugins/countup.min.js"></script>
    <script src="/static/assets/js/plugins/choices.min.js"></script>
    <script src="/static/assets/js/plugins/rellax.min.js"></script>
    <script src="/static/assets/js/plugins/tilt.min.js"></script>
    <script src="/static/assets/js/plugins/choices.min.js"></script>

    <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDTTfWur0PDbZWPr7Pmq8K3j
iDp0_xUziI"></script>
    <script src="/static/assets/js/soft-design-system.min.js?v=1.0.1"
type="text/javascript"></script>
    <script type="text/javascript">
        if (document.getElementById('state1')) {
            const countUp = new CountUp('state1',
document.getElementById("state1").getAttribute("countTo"));
            if (!countUp.error) {
                countUp.start();
            } else {
                console.error(countUp.error);
            }
        }
        if (document.getElementById('state2')) {
            const countUp1 = new CountUp('state2',
document.getElementById("state2").getAttribute("countTo"));

```

```

        if (!countUp1.error) {
            countUp1.start();
        } else {
            console.error(countUp1.error);
        }
    }
    if (document.getElementById('state3')) {
        const countUp2 = new CountUp('state3',
document.getElementById("state3").getAttribute("countTo"));
        if (!countUp2.error) {
            countUp2.start();
        } else {
            console.error(countUp2.error);
        };
    }
}
</script>

{% endblock javascripts %}

```

SOURCE CODE(FLASH PYTHON)

```

# @blueprint.route('/search',methods=['GET','POST'])
# def recommender():
#     with open('G:\\SMART JOB Recommender\\flask-pixel\\csv') as f1:
#         stopw = set(stopwords.words('english'))
#         df11 = pd.read_csv(f1)
#         df11['test']=df11['gender','category','type','brand','description'].
apply(lambda x: ' '.join([word for word in str(x).split() if len(word)>2 and
word not in (stopw)]))

#         def ngrams(string, n=3):
#             string = fix_text(string) # fix text
#             string = string.encode("ascii", errors="ignore").decode()
# remove non ascii chars
#             string = string.lower()
#             chars_to_remove = [")","(",",",".",",","|","[","]","{","}",'"','"']
#             rx = '[' + re.escape(''.join(chars_to_remove)) + ']'
#             string = re.sub(rx, '', string)
#             string = string.replace('&', 'and')
#             string = string.replace(',', ' ')
#             string = string.replace('-', ' ')
#             string = string.title() # normalise case - capital at start of
each word
#             string = re.sub(' +',' ',string).strip() # get rid of multiple
spaces and replace with a single
#             string = ' ' + string + ' ' # pad names for ngrams...

```

```

#         string = re.sub(r'[,.-/]|\\sBD',r'', string)
#         ngrams = zip(*[string[i:] for i in range(n)])
#         return [''.join(ngram) for ngram in ngrams]

#         vectorizer = TfidfVectorizer(min_df=1, analyzer=ngrams,
lowercase=False)
#         tfidf = vectorizer.fit_transform(org_name_clean)

#         def getNearestN(query):
#             queryTFIDF_ = vectorizer.transform(query)
#             distances, indices = nbrs.kneighbors(queryTFIDF_)
#             return distances, indices

#         nbrs = NearestNeighbors(n_neighbors=1, n_jobs=-1).fit(tfidf)
#         unique_org = (df11['test'].values)
#         distances, indices = getNearestN(unique_org)
#         unique_org = list(unique_org)
#         matches = []
#         for i,j in enumerate(indices):
#             dist=round(distances[i][0],2)

#             temp = [dist]
#             matches.append(temp)
#         matches = pd.DataFrame(matches, columns=['Match confidence'])
#         df11['match']=matches['Match confidence']
#         df111=df11.sort_values('match')
#         df22=df111[['Product_url',
'image_url','type','category','description','brand']].head(30).reset_index()
#         return render_template('includes/card-each.html', sproduct = df22,
segment='index')

@blueprint.route('/<template>')
@login_required
def route_template(template):

    try:

        if not template.endswith('.html'):
            template += '.html'

        # Detect the current page
        segment = get_segment(request)

        # Serve the file (if exists) from app/templates/home/FILE.html
        return render_template("home/" + template, segment=segment)

    except TemplateNotFound:

```

```

        return render_template('home/page-404.html'), 404

    except:
        return render_template('home/page-500.html'), 500

# Helper - Extract current page name from request
def get_segment(request):

    try:

        segment = request.path.split('/')[1]

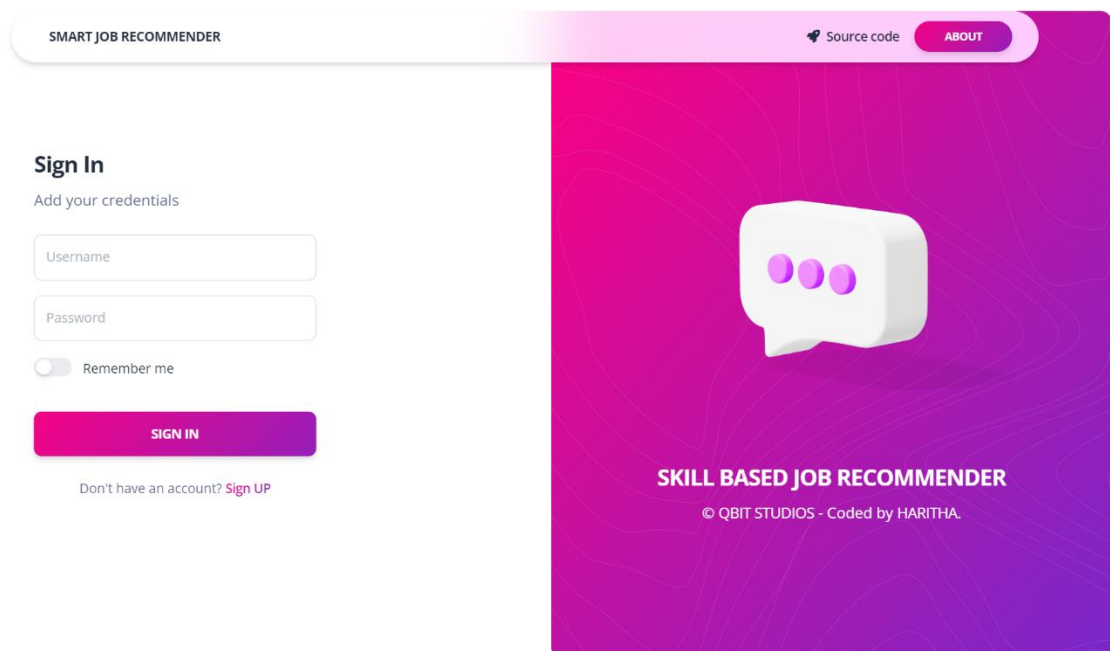
        if segment == '':
            segment = 'index'

        return segment

    except:
        return None

```

OUTPUT



The screenshot displays the web application interface for the 'SMART JOB RECOMMENDER'. The header features the title 'SMART JOB RECOMMENDER' on the left and navigation links for 'Source code' and 'ABOUT' on the right. The main content area is split into two sections. On the left, the 'Sign In' section prompts users to 'Add your credentials' and includes input fields for 'Username' and 'Password', a 'Remember me' checkbox, and a prominent 'SIGN IN' button. Below this, a link for 'Sign UP' is provided for users without an account. On the right, a large, vibrant graphic with a pink-to-purple gradient background features a 3D white speech bubble icon. Below the graphic, the text 'SKILL BASED JOB RECOMMENDER' is displayed, followed by the copyright notice '© QBIT STUDIOS - Coded by HARITHA.'.

SMART JOB RECOMMENDER

[Source code](#)[ABOUT](#)

Add your credentials

Username


Email

Password

☐ Agree with terms

REGISTER

Have an account? [Sign IN](#)



SKILL BASED JOB RECOMMENDER

© QBIT STUDIOS - Coded by HARITHA.

