# Project Report Format

| | |
|---|---|
| **Date** | 19 November 2022 |
| **Team ID** | PNT2022TMID16344 |
| **Project Name** | Skill/ Job Recommender Application |
| **Team Members** | ❖ Varshini J (Team Leader)<br>❖ Pradhiksha R<br>❖ Savitha G<br>❖ Sneha R L |
| **Team Mentor** | Umamageswaran J |
| **College and Department** | R.M.K. Engineering College / Information Technology |


1. **INTRODUCTION**
   - 1.1 Project Overview
   - 1.2 Purpose
2. **LITERATURE SURVEY**
   - 2.1 Existing problem
   - 2.2 References
   - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   - 3.1 Empathy Map Canvas
   - 3.2 Ideation & Brainstorming
   - 3.3 Proposed Solution
   - 3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   - 4.1 Functional requirement
   - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   - 5.1 Data Flow Diagrams
   - 5.2 Solution & Technical Architecture
   - 5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
   - 6.1 Sprint Planning & Estimation
   - 6.2 Sprint Delivery Schedule
   - 6.3 Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   - 7.1 Feature 1
   - 7.2 Feature 2
   - 7.3 Database Schema (if Applicable)
8. **TESTING**

8.1 Test Cases
8.2 User Acceptance Testing

**9. RESULTS**

9.1 Performance Metrics

**10. ADVANTAGES & DISADVANTAGES**

**11. CONCLUSION**

**12. FUTURE SCOPE**

**13. APPENDIX**

13.1 Source Code

13.2 GitHub & Project Demo Link

## TABLE OF CONTENTS



| *S.NO* | *TOPIC* | *PAGE NO* |
|---|---|---|
| **01** | **INTRODUCTION** | |
| 1.1 | Project Overview | |
| 1.2 | Project Purpose | |
| **02** | **LITERATURE SURVEY** | |
| 2.1 | Existing Problem | |
| 2.2 | References | |
| 2.3 | Problem Statement and Definition | |
| **03** | **IDEATION AND PROPOSED SOLUTION** | |

| | | |
|---|---|---|
| 3.1 | Empathy Map Canvas | |
| 3.2 | Ideation and Brainstorming | |
| 3.3 | Proposed Solution | |
| 3.4 | Problem Solution Fit | |
| **04** | **REQUIREMENT ANALYSIS** | |
| 4.1 | Functional Requirement | |
| 4.2 | Non-Functional Requirement | |
| **05** | **PROJECT DESIGN** | |
| 5.1 | Data Flow Diagrams | |
| 5.2 | Solution and Technical Architecture | |
| 5.3 | User Stories | |
| **06** | **PROJECT PLANNING AND SECHEDULE** | |
| 6.1 | Sprint Planning and Estimation | |
| 6.2 | Sprint Delivery Schedule | |
| 6.3 | Reports from JIRA | |

| | | |
|---|---|---|
| 07 | CODING and SOLUTIONING (Explain the features added in the project along with code) | |
| 7.1 | Feature – 1 | |
| 7.2 | Feature – 2 | |
| **08** | **TESTING** | |
| 8.1 | Test Cases | |
| 8.2 | User Acceptance Testing | |
| **09** | **RESULTS** | |
| 9.1 | Performance Metrics | |
| **10** | **ADVANTAGES AND DISADVANTAGES** | |
| **11** | **CONCLUSION** | |
| **12** | **FUTURE SCOPE** | |
| **13** | **APPENDIX** | |
| 13.1 | Source Code | |
| 13.2 | Project Demo Link and GitHub | |

# Introduction:

## Project Overview:

Recently, most job-seekers exploit the internet to look for jobs online recruiting systems. However, with the rapid growth of information on these systems causes job searching become a time-consuming task. In other words, applicants have been dealing with the problem of information overload. Therefore, there is a need to develop job recommender systems that automatically suggest a ranked list of jobs matching users' particular interest. Nobody can deny the fact that these systems play a significant role in connecting between employees and employers. Normally, job recommender systems do not only serve for employers but also serve for applicants. Automatically, applicants obtain a ranked list of recommended jobs associated with their preference, while employers receive a ranked list of potential candidates matching their recruiting need. In this paper, we concentrate on recommending associated jobs for applicants. The question is which method should be applied for job recommender system. In order to choose a suitable algorithm, the aim of this study is to conduct a comparison study of popular methods for job recommendation. The key contributions of this study are summarized as follows: Building an experimental dataset for the problem of job recommendation. Providing a comparison study of popular methods (CB, CF, Linear-Hybrid) for job recommendation. The rest of the paper is organized as follows. In section II, formal definition of job recommendation is presented.

With the rapid development of the information society, a vast amount of information is now available from many diverse sources. To deal with this information overload problem, recommender systems, which assist users to discover relevant information, products and services, have now been successfully deployed in many domains [1, 2]. In this paper, we consider the task of job recommendation, where suitable jobs are recommended to users based on their past job application history. Recent research has also focused on this task. For example, in [3] jobs are recommended based on a graph constructed from the previously observed job transition patterns of users. These patterns were based on various features relating to employer sector and size, employee experience and education, etc. A supervised machine learning approach was then adopted to recommend suitable new jobs to users. In [4], a graph-based hybrid recommender is proposed which considers both content-based user and job profile similarities and interaction-based activities (e.g., applying to or liking a job). Personalized recommendations of candidates and jobs are then generated using a PageRank-style ranking algorithm. Further, a collaborative filtering approach based on implicit profiling techniques was proposed in [5] to deliver personalized, query-less job recommendations to users. For other work in this area, see [6, 7, 14]. To date, research in the area of job recommendation has not considered an in-depth analysis of traditional content-based or case-based approaches to recommendation. Thus, the core contributions of this work are as follows. Firstly, a number of content-based approaches to job recommendation are proposed.

## Literature Survey:

There are lots of paper works and a few projects done on Skill/Job recommender and the problems that are associated with the existing solution is to be detected and the references are mentioned while finding problems in one of the papers that is as follows.

## Existing Solution:

"The hybrid job recommendation approaches presented combined two or more techniques to overcome the problems that suffer from using each technique separately. For example , while the probability hybrid approaches in paragraph A realized a bidirectional recommendation and tried to cover different selection dimensions, they need to enhance by including more features for individuals and extending by various relational aspects other than trust. Additionally, they only adopted the binary representation with Yes and No when state user preferences, and it cannot measure the degree of users preferences for each index well, so the quality of recommendation is not high (Yu et al., 2011). As for the content-based job recommender systems, it is presented some approaches and systems based on CBF techniques. As mentioned in the CBF, it is limited by the features that explicitly associated with recommended objects. Therefore, since the applicants" resumes are usually represented by their most important features using some key words, CBF systems cannot distinguish between different keywords meaning. In addition, the problem usually associated with the pure CBF systems; it cannot recommend jobs that are different from anything the user has seen before. Jobs will be recommended if they are similar to other jobs that the applicant has already interested. Thus, the applicants have to rate a sufficient number of jobs before a CBF recommender system can really understand the applicants preferences and present reliable recommendations. For example, the machine learned recommender system in paragraph 0 builds an automated system to recommend jobs for applicants based on their past job histories. This system is used a classifier that makes a recommendation by training them on content information. It suffered from scalability and data sparsity problems (Ghazanfar and Pr¨ugel-Bennett, 2010). In addition to, this system performs the recommendation as a unary relation and ignores the person-team fit when matching candidates with jobs. Table 4 summarizes the advantages and dis-advantages of these approaches and systems. Finally, from our research and findings from existing literature, we showed the increasing importance of information technology for the recruitment process. Thus, the important challenge for most organizations as identified by the literature analysis is the low qualification of applicants, where skills of applicants do not fit with the job profile

## References:


1. Title: Job Recommendation based on Job Seeker Skills: An Empirical Study. Source: Researchgate. Author: Jorge Valverde-Rebaza. Date: March 2018 Website: https://www.researchgate.net/publication/356601605_Job_Recommender_Systems_A_Review

2. Title: Job Recommender Systems: A Review Source: ResearchGate Author: Corné de Rujit Date: November 2021 Website: https://www.researchgate.net/publication/325697854_Job_Recommendation_based_on_Job_Seeker_Skills_An_Empirical_Study

3. Title: Extracting Relations Between Sectors Source: ResearchGate Author: Atkan Kara. Date: August 2022 Website: https://www.researchgate.net/publication/363128874_Extracting_Relations_Between_Sectors

4. Title: Job Candidate Rank Approach Using Machine Learning Techniques Author: Lamiaa Mostafa Source: Date: March, 2020. Website: https://www.researchgate.net/publication/349816523_Job_Candidate_Rank_Approach_Using_Machine_Learning_Techniques

5. Title: JOB RECOMMENDATION USING TEXT PROCESSING A Project Report. Author: Dipanwita Saha. Source: ResearchGate Date: July 2022 Website: https://www.researchgate.net/publication/362889143_JOB_RECOMMENDATION_USING_TEXT_PROCESSING_A_Project_Report

6. Title: Job Recommendation Based on Extracted Skill Embeddings Author: anonymous. Source: ResearchGate Date: September 2022 Website: https://www.researchgate.net/publication/363190802_Job_Recommendation_Based_on_Extracted_Skill_Embeddings

7. Title: Job recommendation systems for enhancing e-recruitment process Author: Shaha Alotaibi Source: ResearchGate Date: February 2014 Website: https://www.researchgate.net/publication/323079153_Job_recommendation_systems_for_enhancing_e-recruitment_process

8. Title: Job recommendation systems for enhancing e-recruitment process Authors: Shaha T Alotaibi Abdulrahman A Mirza Source: ResearchGate Date: January 2012 Website: https://www.researchgate.net/publication/323078898_Job_recommendation_systems_for_enhancing_e-recruitment_process

9. Title: Job Recommendation based on Job Seeker Skills: An Empirical Study Authors: Jorge Valverde-Rebaza, Visibilia, Ricardo Puma Source: ResearchGate Date:March 2018 Website: https://www.researchgate.net/publication/325697854_Job_Recommendation_based_on_Job_Seeker_Skills_An_Empirical_Study

10. Title: A survey of job recommender systems Authors: Shaha Alotaibi Source: ResearchGate Date: July 2012 Website: https://www.researchgate.net/publication/272802616_A_survey_of_job_recommender_systems

## The Approach to the Project:

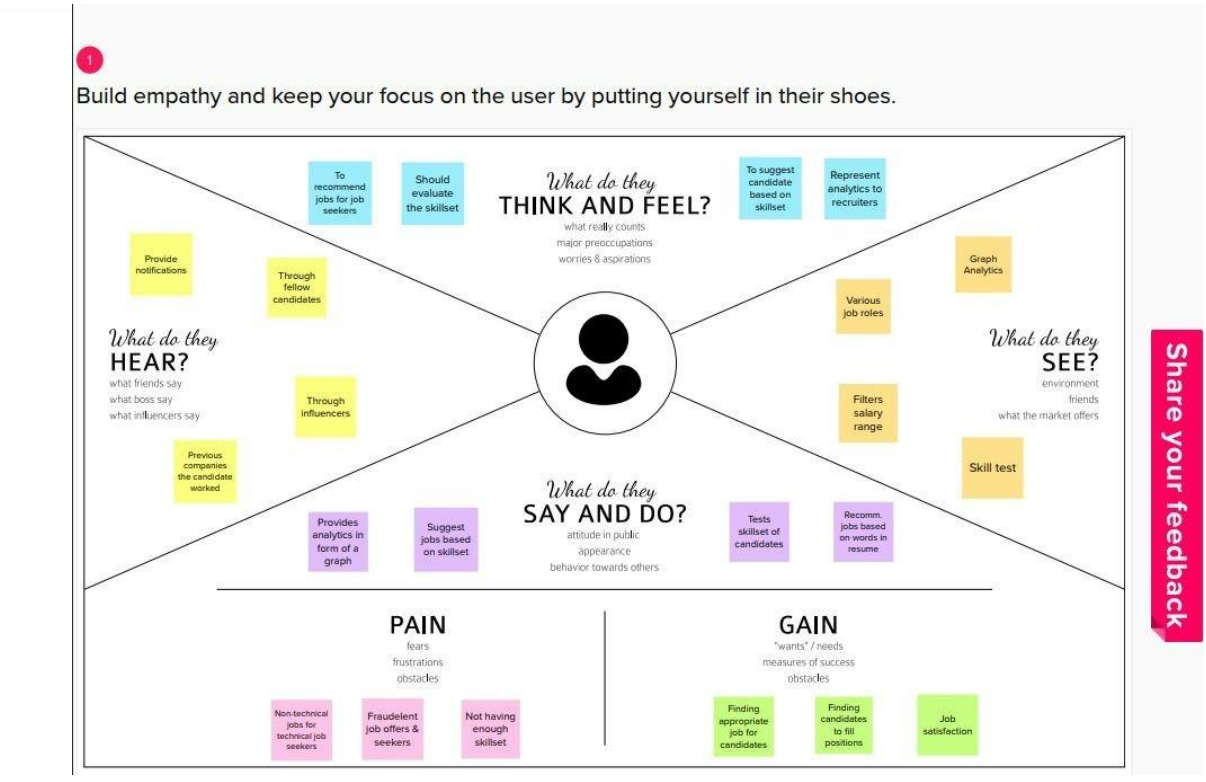
There are exceptional solutions to job recommender, but the modification needs to be applied to the existing model and a skill evaluation phase must be implemented and the users must receive job recommendations appropriately. This approach is a way to enhance the existing wide range of applications.

2.3 Problem Statement Definition

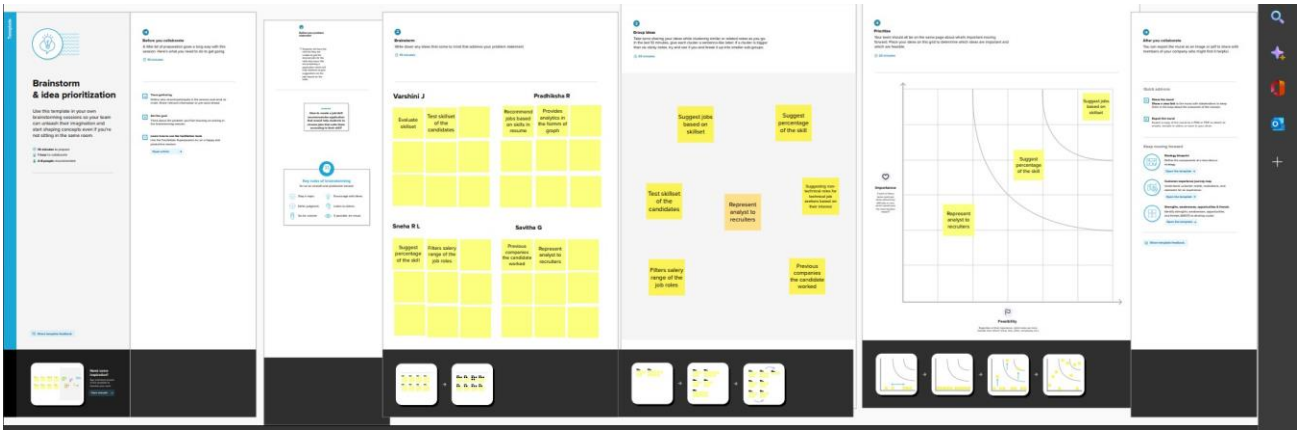
- Design a skill recommender app that can suggest new skills for users to learn based on their current skillset and career goals.
- The Skill Recommender app will allow users to input their skills and interests, and receive recommendations for other skills they might be interested in. The app will use data from the user's profile and activity to make recommendations.

## 3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



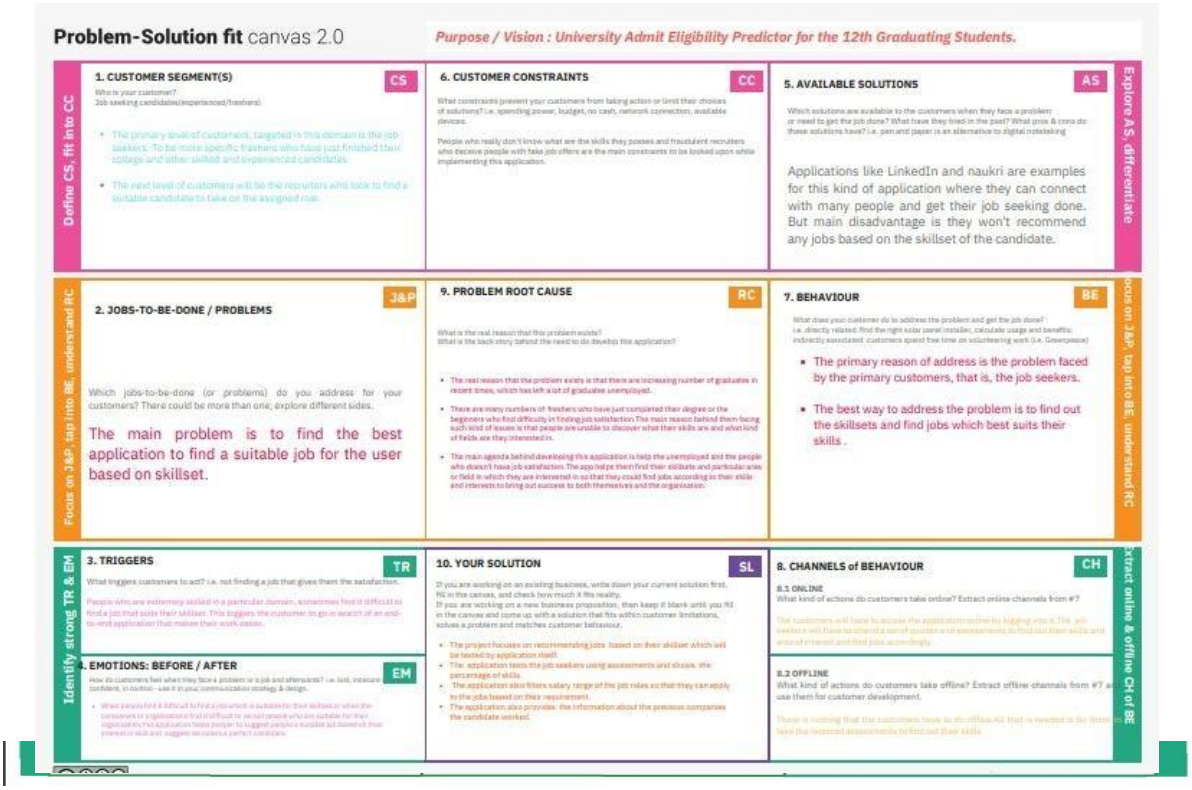
## 3.2 Ideation & Brainstorming




## 3.3 Proposed Solution

Premium policy is an issue to the users. Look for field-based jobs as searching for fields as a whole is time-consuming. Salary calculator for the estimation of the pay. Earnings estimator based on knowledge of users.Open doors for every users as there is free accessProviding recruiters with better candidatesData can be scaled up and scaled down according to the number of current job openings available.

## 3.4 Problem Solution fit




# 4. REQUIREMENT ANALYSIS

## 4.1 Functional requirement

**Functional Requirements:**

Following are the functional requirements of the proposed solution.

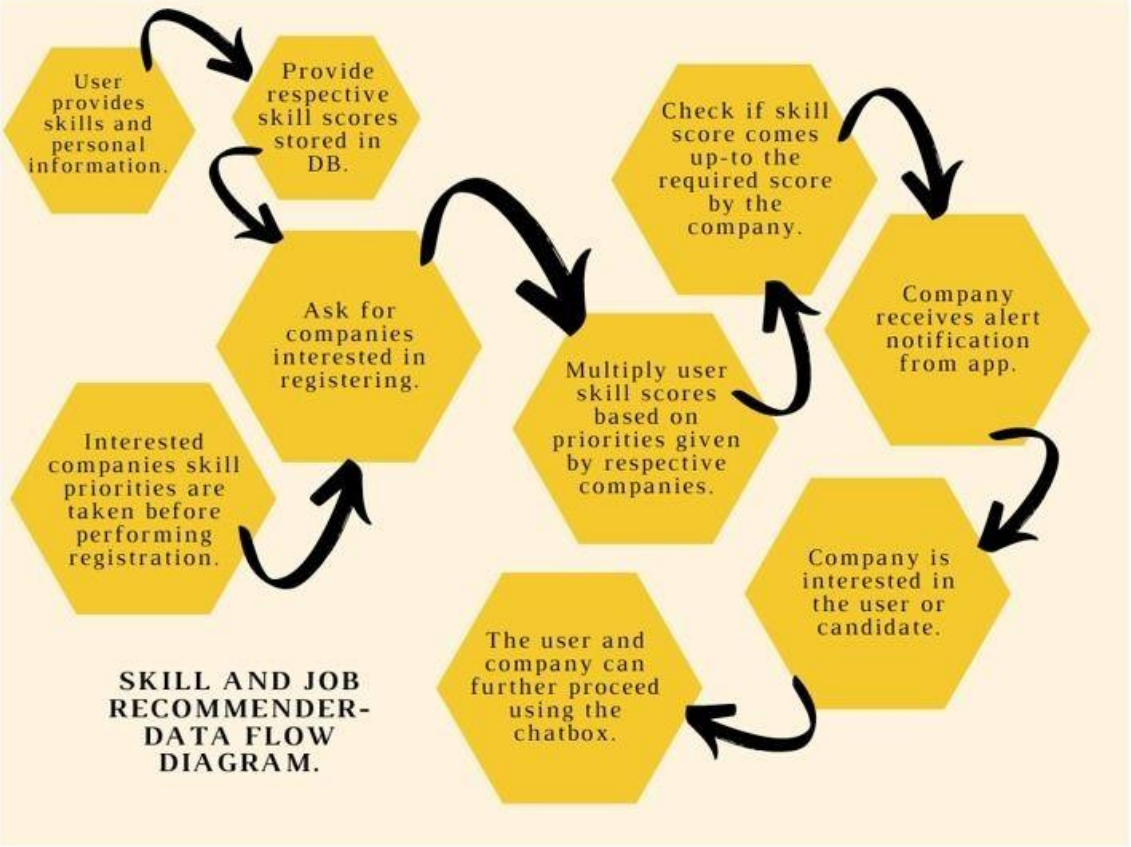
| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | User Registration | Registration through Form<br>Registration through Gmail<br>Registration through LinkedIN |
| FR-2 | User Confirmation | Confirmation via Email<br>Confirmation via OTP |
| FR-3 | | |
| FR-4 | | |
| | | |
| | | |

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | **Usability** | Once job offers and profiles are filtered, the second step is text preprocessing. In this task, we perform stopwords removal, tokenization and lemmatization for the Portuguese language. this option helps us to find the specified jobs for us. |
| NFR-2 | **Security** | where a job seeker applies to a job by creating a profile on a job portal by providing all his/her work preferences. These work preferences of each user can be collected from each user and provide job recommendations based on their preference. This helps us to avoid the fake job recommendations. |
| NFR-3 | **Reliability** | The scraper is set up to avoid duplicate job offers, thus all the job offers are unique. To making the user reliabile. |
| NFR-4 | **Performance** | Once job offers and profiles are filtered, the second step is text preprocessing. In this task, we perform |

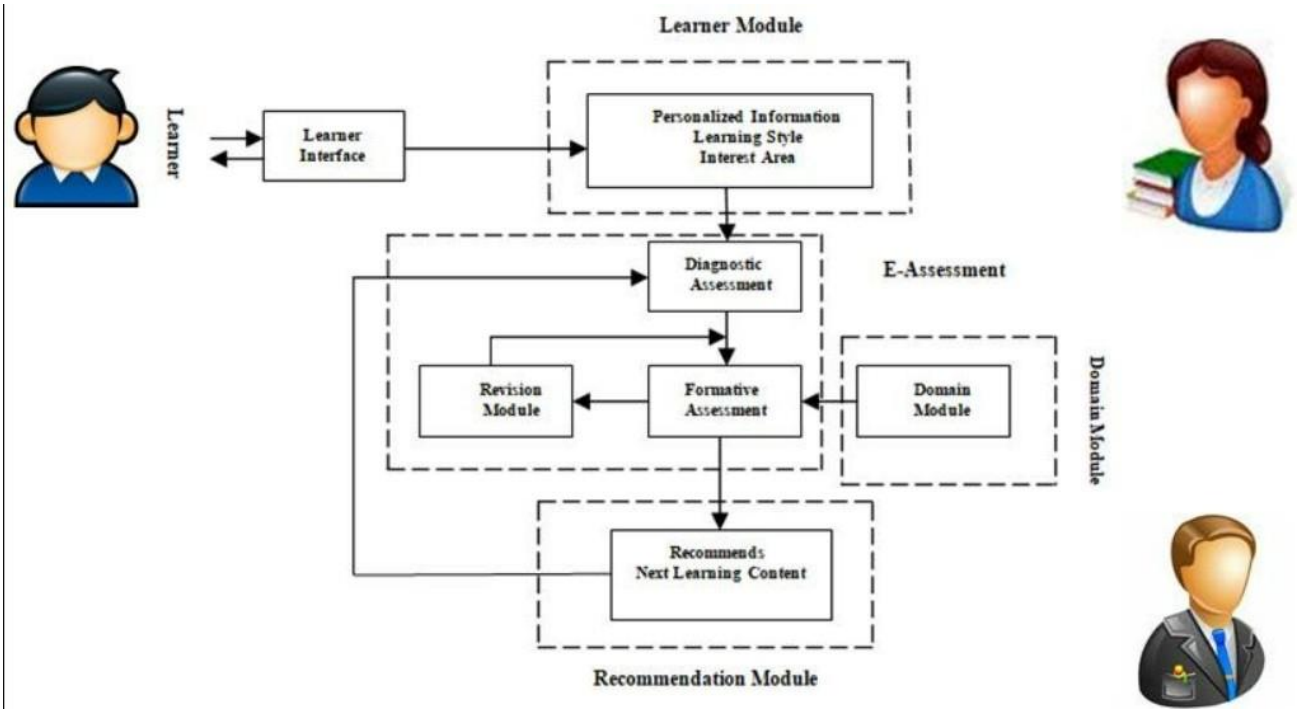
| | | |
|---|---|---|
| | | stopwords removal, tokenization and lemmatization for the Portuguese language. this option helps us to find the specified jobs for us |
| NFR-5 | **Availability** | The user can get the available resources about the job information. Also, getting notified about job availabilities. |
| NFR-6 | **Scalability** | Here a seeker looks up for the job he would find relevant to him and apply for it. As there are many job boards, applicants tend to use the tool that provides better services to them, services such as writing a CV, creating a job profile, and recommending new jobs to a job seeker |

## 5. PROJECT DESIGN

5.1 Data Flow Diagrams



## 5.2 Solution & Technical Architecture



## 5.3 User Stories

**User Stories**

Use the below template to list all the user stories for the product.

| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
|---|---|---|---|---|---|---|
| Customer (Mobile user) | 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 Linkedin | I can register & access the dashboard with Linkedin Login | Low | Sprint-2 |
| | | USN-4 | As a user, I can register for the application through Gmail | I can register and access the dashboard with G mail login | Medium | Sprint-1 |
| | Login | USN-5 | As a user, I can log into the application by entering email & password | I can access the dashboard | High | Sprint-1 |
| | Search | USN-6 | As a user , I can search for the desired companies | Companies related to the search terms are listed | High | Sprint-2 |
| Customer (Web user) | Apply | USN-7 | As a user , I can apply for a company | Application is submitted to the company | High | Sprint-2 |
| | Review | USN-8 | As a user, I can review the company | Review is listed on the company profile | Medium | Sprint-2 |
| Admin | Forward | USN-9 | As a customer care executive , I must forward the applications to the respective company | The application is received by the company | High | Sprint-1 |
| | Send confirmation | USN-10 | Confirmation mail is sent from the respected company | Confirmation is received by the user | High | Sprint-2 |
| | Manage review | USN-11 | As an admin , I must make the reviews appear on the company's profile | Reviews appear on the company's profile | Low | Sprint 2 |

# Project P anning

## Sprint Pl nning anc Delivery:

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint 1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Varshini J |
| | | USN-2 | As a user, I will receive confirmation email once I have registered for the application | 1 | High | Sneha R L |
| | Login | USN-3 | As a user, I can log into the application by entering email & password | 1 | High | Savitha G |
| | Dashboard | USN-4 | Logging in takes to the dashboard for the logged user. | 2 | High | Pradhiksha R |
| | Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Sprint 2** | Workspace | USN-1 | Workspace for personal expense tracking | 2 | High | Varshini J |
| | Charts | USN-2 | Creating various graphs and statistics of customer's data | 1 | Medium | Sneha R L |
| | Connecting to IBM DB2 | USN-3 | Linking database with dashboard | 2 | High | Savitha G |
| | | USN-4 | Making dashboard interactive with JS | 2 | High | Pradhiksha R |
| **Sprint-3** | | USN-1 | Wrapping up the server side works of frontend | 1 | Medium | Varshini J |
| | Watson Assistant | USN-2 | Creating Chatbot for recommending jobs and forclarifying user's query | 1 | Medium | Sneha R L |
| | SendGrid | USN-3 | Using SendGrid to send mail to the user about their expenses | 1 | Low | Savitha G |
| | | USN-4 | Integrating both frontend and backend | 2 | High | Pradhiksha R |

| | | | | | | |
|---|---|---|---|---|---|---|
| Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only | | | | | | |
| **Sprint-4** | Docker | USN-1 | Creating image of website using docker/ | 2 | High | Savitha G |
| | Cloud Registry | USN-2 | Uploading docker image to IBM Cloud registry | 2 | High | Varshini J |
| | Kubernetes | USN-3 | Create container using the docker image and hosting the site | 2 | High | Sneha R L |
| | Exposing | USN-4 | Exposing IP/Ports for the site | 2 | High | Pradhiksha R |

| | | | | | | |
|---|---|---|---|---|---|---|
| Sprint-1 | | USN-3 | As a user, I will receive confirmation email once I have registered for the application | High | I can receive confirmation email & click confirm | Shashank, Srinivas |
| Sprint-1 | | USN-4 | As a user, I can register for the application through Facebook | Low | I can register & access the dashboard with Facebook Login | Shashank, Srinivas |
| Sprint-1 | | USN-5 | As a user, I can register for the application through Gmail | Medium | I can receive confirmation email & click confirm | Shashank, Srinivas |
| Sprint-1 | Login | USN-6 | As a user, I can log into the application by entering email & password | High | I can access my account / dashboard | Siva Sathwik, Sreenath |

## 7. CODING & SOLUTIONING (Explain the features added in the project along with code)

**7.1** Feature 1

```
<script>
  window.watsonAssistantChatOptions = {
   integrationID: "1df8b62f-ee5c-42bb-9a2f-6edcb89d85f7", // The ID of this integration.
   region: "jp-tok", // The region your integration is hosted in.
   serviceInstanceID: "ee519db2-aba8-4373-a6a2-e4e6fea336d7", // 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>
```

**7.2** Feature 2

Job_post.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
  <title>Document</title>
  <link rel="stylesheet" href="static/css/job_post.css" />
 </head>
  <body style="background-color: rgb(255, 204, 0)" height="40%"
width="80%">
  <script>
   window.watsonAssistantChatOptions = {
    integrationID: "6e54406e-1b6b-4b7c-8b9e-a21f9adbb3de", // The ID of
this integration.
     region: "au-syd", // The region your integration is hosted in.
      serviceInstanceID: "bc1d370b-fe79-4e14-be04-a0cacaef09e2", // 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>
    <header>
     <div class="wrapper">
      <div class="logo">
       <a href="/"><img src="static/img/job_logo.png" alt="" /></a>
      </div>

      <ul class="nav-area">
       <li><a href="posts">post Jobs</a></li>
       <li><a href="list">view job</a></li>
      </ul>
     </div>

     <section>
      <div class="contentBx">
       <div class="card">
        <div class="formBx">
         <h2>Job Posting form</h2>
```

```
<br />
<form action="{{ url_for('addrec') }}" method="get|post">
 <div class="inputBx">
  <span>Company name</span>
  <input
   type="text"
   name="dname"
   id="dname"
   placeholder=""
   required
   class="form-control"
  />
 </div>
 <br />

 <div class="inputBx">
  <span>Job title</span>
  <input
   type="text"
   name="dtitle"
   id="dtitle"
   placeholder=""
   required
   class="form-control"
  />
 </div>
```

```
<br />
<div class="inputBx">
 <span>Job role</span>
 <input
  type="text"
  name="drole"
  id="drole"
  placeholder=""
  required
  class="form-control"
 />
</div>
<br />
<div class="inputBx">
 <span>Job Description</span>
 <label for="Description"></label>
 <textarea
  rows=""
  cols=""
  name="description"
  id="description"
 ></textarea>
</div>
<br />
<div class="remember">
 <input type="checkbox" id="rem1" name="" value="" />
```

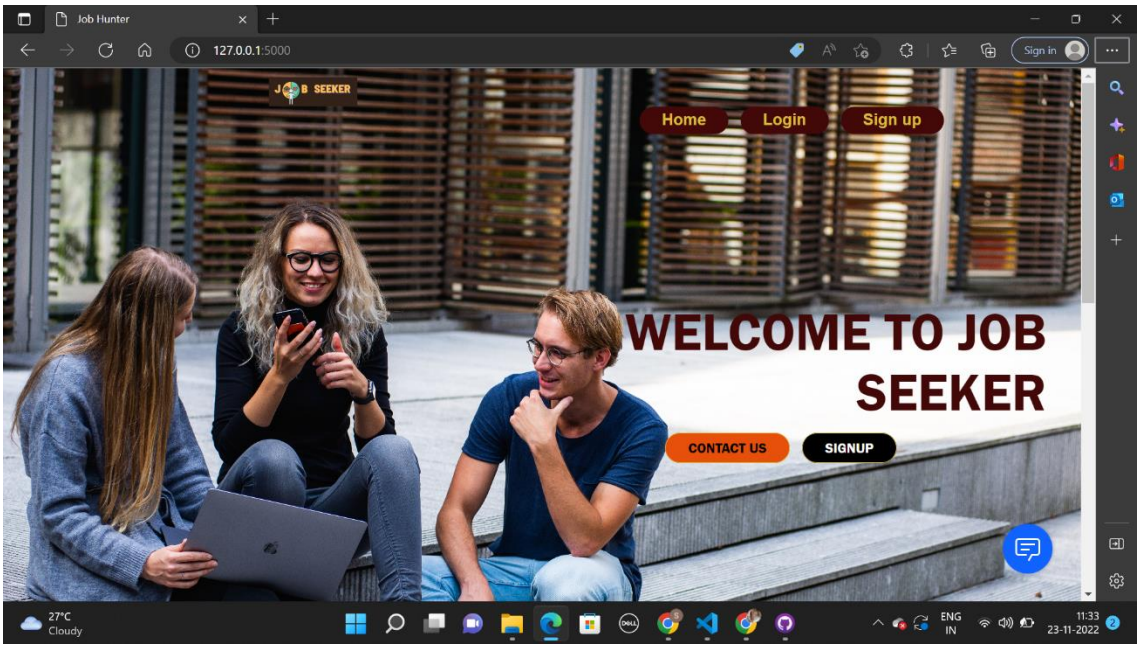
```
            <label for="rem1"
              >I agree to the terms and condition of the form</label
            ><br />
           </div>
           <br />
           <div class="inputBx">
            <input type="submit" value="Submit" name="" />
           </div>
          </form>
         </div>
        </div>
       </div>
      </section>
     </header>

     <div class="grid-container">
      <div class="grid">
       <img
        src="../static/img/33172-01-finishig-studies.gif"
        width="100px"
        height="100px"
       />
          <h5>Get your dream job within a week.<br />Gain on-demand
skills.</h5>
      </div>
      <div class="grid">
```

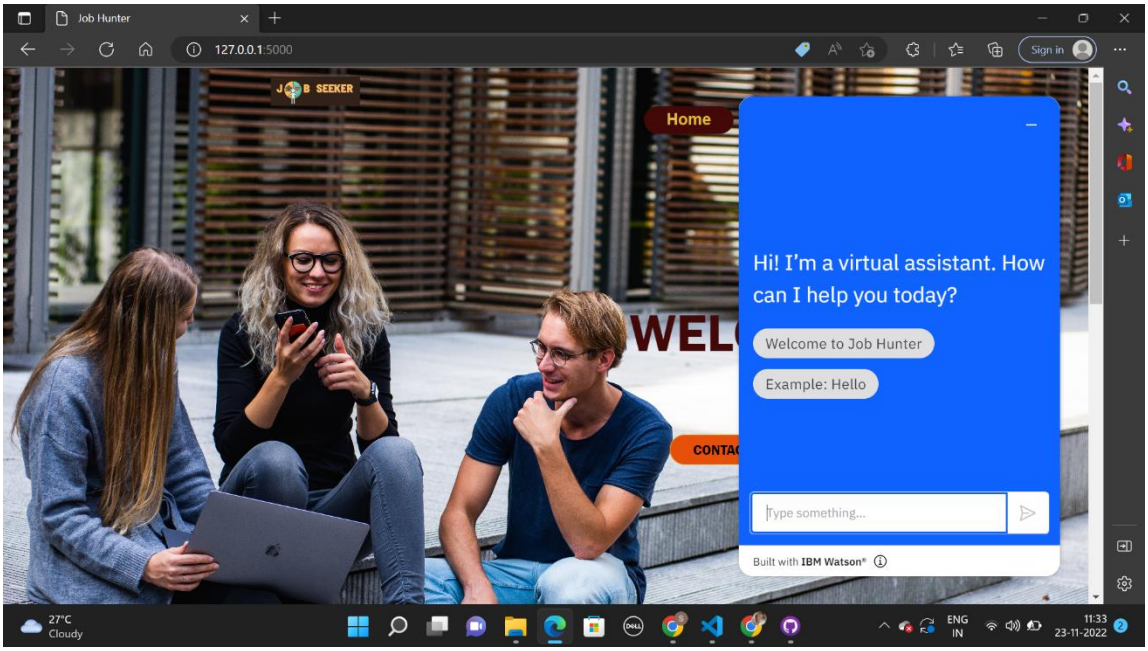
```
                <img src="../static/img/27425-train.gif" width="100px"
height="100px" />
      <h5>Personalized Recommendation.</h5>
     </div>
     <div class="grid">
      <img
       src="../static/img/75164-glurtr-anmation.gif"
       width="100px"
       height="100px"
      />
      <h5>Learn at your own pace, with lifetime access.</h5>
     </div>
   </div>

   <hr id="start" />
   <div class="job-sign">
    <br /><br />
    <h1>Find the best job for your career</h1>
    <button id="sign-center">View jobs</button>
   </div>
  </body>
</html>
```

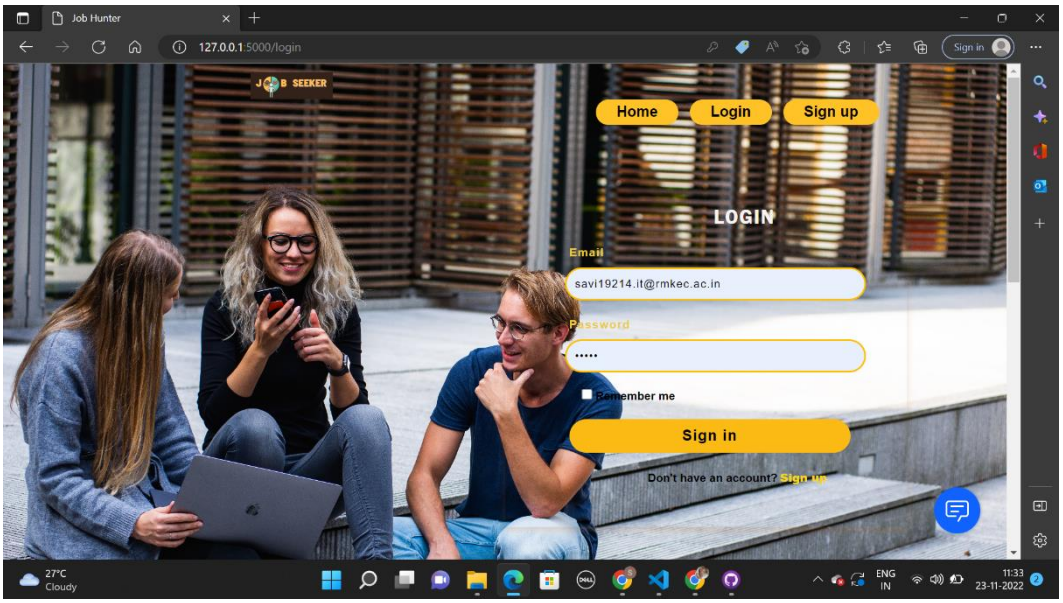
## 8. TESTING

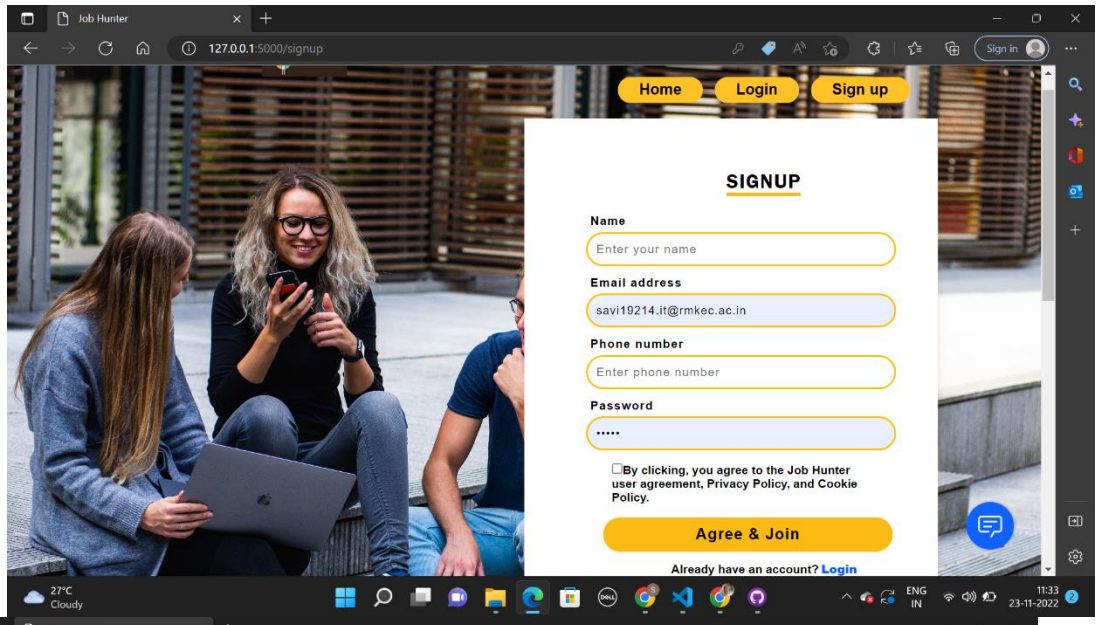
Black Box testing is the method that does not consider the internal structure, design, and product implementation to be tested. In other words, the tester does not know its internal functioning. The Black Box only evaluates the external behaviour of the system. The inputs received by the system and the outputs on responses it produces are tested. The White Box test method is the one that looks at the code and structure of the product to be tested and uses that knowledge to perform the tests. This method is used in the Unit testing phase, although it can also occur in other stages such as Integration tests. For the execution of this method, the tester or the person who will use this method must have extensive knowledge of the technology used to develop the program.

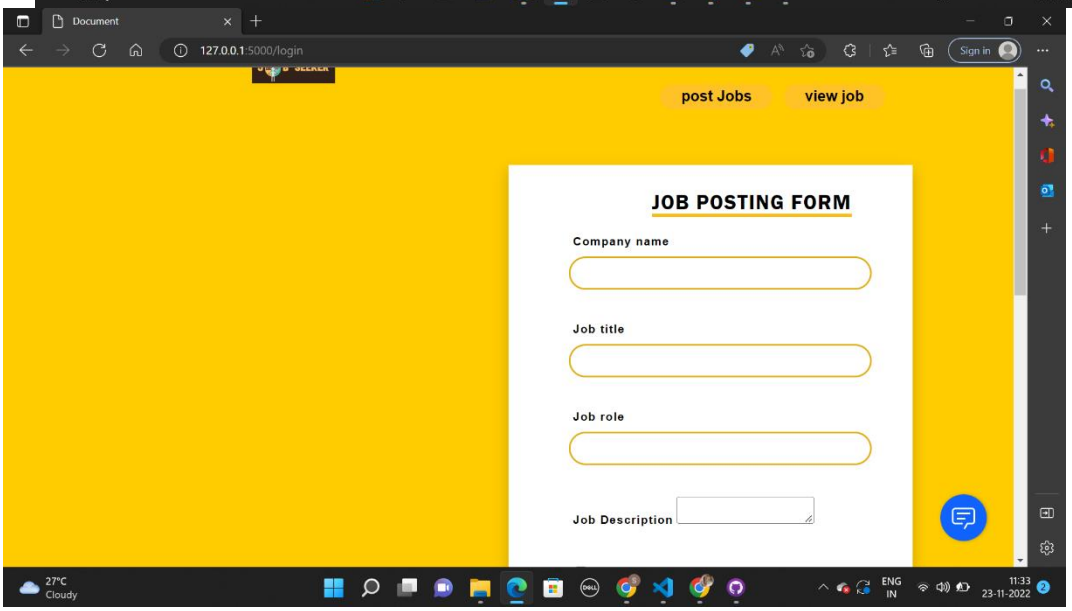
# 9. RESULTS


Job Hunter
127.0.0.1:5000
Sign in
JOB SEEKER
Home
Login
Sign up
WELCOME TO JOB SEEKER
CONTACT US
SIGNUP
27°C
Cloudy
ENG
IN
11:33
23-11-2022


Job Hunter
127.0.0.1:5000
Sign in
JOB SEEKER
Home
Hi! I'm a virtual assistant. How can I help you today?
Welcome to Job Hunter
Example: Hello
Type something...
Built with IBM Watson®
27°C
Cloudy
ENG
IN
11:33
23-11-2022


Job Hunter
127.0.0.1:5000/login
Sign in
JOB SEEKER
Home
Login
Sign up
LOGIN
Email
savi19214.it@rmkec.ac.in
Password
Remember me
Sign in
Don't have an account? Sign up
27°C
Cloudy
ENG
IN
11:33
23-11-2022


Job Hunter
127.0.0.1:5000/signup
Sign in
Home
Login
Sign up
SIGNUP
Name
Enter your name
Email address
savi19214.it@rmkec.ac.in
Phone number
Enter phone number
Password
By clicking, you agree to the Job Hunter user agreement, Privacy Policy, and Cookie Policy.
Agree & Join
Already have an account? Login
27°C
Cloudy
ENG
IN
11:33
23-11-2022


Document
127.0.0.1:5000/login
Sign in
post Jobs
view job
JOB POSTING FORM
Company name
Job title
Job role
Job Description
27°C
Cloudy
ENG
IN
11:33
23-11-2022

### 9.1 Performance Metrics

The performance of a recommendation algorithm is evaluated by using some specific metrics that indicate the accuracy of the system. The type of metric used depends on the type of filtering technique. Root Mean Square Error (RMSE), Receiver Operating Characteristics (ROC), Area Under Cover (AUC), Precision, Recall and F1 score is generally used to evaluate the performance or accuracy of the recommendation algorithms.

***Root-mean square error (RMSE)***. RMSE is widely used in evaluating and comparing the performance of a recommendation system model compared to other models. A lower RMSE value indicates higher performance by the recommendation model. RMSE, as mentioned by [69], can be as represented as follows:

RMSE=√ 1/Np∑u,i(pui−rui)^2 (1)

where, $N_p$ is the total number of predictions, $p_{ui}$ is the predicted rating that a user $u$ will select an item $i$ and $r_{ui}$ is the real rating.

***Precision***. Precision can be defined as the fraction of correct recommendations or predictions (known as True Positive) to the total number of recommendations provided, which can be as represented as follows:

Precision=True Positive (TP)/True Positive(TP)+False Positive (FP)(2)

It is also defined as the ratio of the number of relevant recommended items to the number of recommended items expressed as percentages.

***Recall***. Recall can be defined as the fraction of correct recommendations or predictions (known as True Positive) to the total number of correct relevant recommendations provided, which can be as represented as follows:

Recall=True Positive (TP)/True Positive(TP)+False Negative (FN) (3)

It is also defined as the ratio of the number of relevant recommended items to the total number of relevant items expressed as percentages.

***F1 Score***. F1 score is an indicator of the accuracy of the model and ranges from 0 to 1, where a value close to 1 represents higher recommendation or prediction accuracy. It represents precision and recall as a single metric and can be as represented as follows:

F1 score=2×Precision∗Recall/Precision+Recall (4)

***Coverage***. Coverage is used to measure the percentage of items which are recommended by the algorithm among all of the items.

***Accuracy***. Accuracy can be defined as the ratio of the number of total correct recommendations to the total recommendations provided, which can be as represented as follows:

Accuracy=TP+FN/TP+FN+TN+FP (5)

***Intersection over union (IoU)***. It represents the accuracy of an object detector used on a specific dataset [70].

IoU=TP/TP+FN+FP (6)

***ROC***. ROC curve is used to conduct a comprehensive assessment of the algorithm's performance [57].

***AUC***. AUC measures the performance of recommendation and its baselines as well as the quality of the ranking based on pairwise comparisons [5].

*Rank aware top-N metrics*. The rank aware top-N recommendation metric finds some of the interesting and unknown items that are presumed to be most attractive to a user [71]. Mean reciprocal rank (MRR), mean average precision (MAP) and normalized discounted cumulative gain (NDCG) are three most popular rank aware metrics.

*MRR.* MRR is calculated as a mean of the reciprocal of the position or rank of first relevant recommendation [72,73]. MRR as mentioned by [72,73] can be expressed as follows:

MRR=1/Nu ∑u∈Nu 1/Lnu[k]∈Ru (7)

where $u$, $N_u$ and $R_u$ indicate specific user, total number of users and the set of items rated by the user, respectively. $L$ indicates list of ranking length ($n$) for user ($u$) and $k$ represents the position of the item found in the he lists $L$.

*MAP:* MAP is calculated by determining the mean of average precision at the points where relevant products or items are found. MAP as mentioned by [73] can be expressed as follows.

MAP=1Nu|Ru|∑nk=1k(Lnu[k]∈Ru)Pu@k (8)

where Pu represents precision in selecting relevant item for the user.

NDCG: NDCG is calculated by determining the graded relevance and positional information of the recommended items, which can be expressed as follows [73].

NDCGu=∑nk=1G(u,n,k)D(k)/∑nk=1G∗(u,n,k)D(k) (9)

where $D$ ($k$) is a discounting function, $G$ ($u$, $n$, $k$) is the gain obtained recommending an item found at $k$-th position from the list L and $G^*$ ($u$, $n$, $k$) is the gain related to $k$th item in the ideal ranking of $n$ size for u user.

## 10. ADVANTAGES & DISADVANTAGES

### Advantages

- 1. Job recommendations can help you find a job that fits your skills and interests.
- Job recommendations can save you time by showing you jobs that are more likely to be a good match for you.
- Job recommendations can help you discover new opportunities that you may not have considered otherwise.

- The biggest advantage of job recommendation systems for recruiters is that they can help save time by automating the process of finding and recommending candidates for open positions.

- Job recommendation systems can also provide a more objective perspective on candidates by using data and algorithms to identify which candidates are the best match for a specific job.

- In addition, job recommendation systems can help recruiters keep track of a large pool of candidates and automatically notify them when new job openings that match their skills and experience become available.

- Job recommendations can help you find jobs that are not advertised to the general public.

- Job recommendations can help you connect with recruiters and hiring managers.

- Job recommendations can help you find jobs that match your salary requirements.

- Job recommendations can help you find jobs that are located in your desired geographical area..

**Disadvantages**

- The app may not have all the skills you need in its database.
- The app may not be updated regularly, so you could end up with outdated information.
- The app may not be able to recommend skills based on your specific needs.
- The system may not be able to accurately recommend jobs that are a good fit for the user, leading to frustration and wasted time.
- Job recommendations may not be accurate or tailored to the user. The system may not be able to take into account the user's preferences or specific situation.
- The system may be biased. For example, if the system relies on input from previous employers, it may favor candidates who have already worked for the company.
- The system may be expensive to implement and maintain.
- The system may be abused. For example, users may try to game the system by entering inaccurate or misleading information.

## 11. CONCLUSION

In this project, we proposed a framework for skill/job recommendation application. This framework facilitates the understanding of the 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.

## FUTURE SCOPE

The app could be expanded to include a social media component, where users could connect with each other and share tips and tricks. The app could also be expanded to include a gamification element, where users could earn points and badges for using the app frequently or for completing tasks.

There are many ways in which the skill recommender app could be improved. For example, the app could be made more personalized by taking into account the user's specific skills and interests. Additionally, the app could be made more interactive, perhaps by incorporating game-like elements or by allowing users to ask questions of the app's artificial intelligence (AI) system. Finally, the app could be made more comprehensive, perhaps by including a database of all known skills and by allowing users to search for skills by keyword.

## 12. APPENDIX

Source Code

```
from flask import Flask,render_template,request,session, redirect,url_for
import ibm_db
from dotenv import load_dotenv
import os

load_dotenv()

app = Flask(__name__)
app.secret_key= os.getenv("SECRET_KEY")

#Cloud Connection values
database_name = os.getenv("DATABASE")
host_name = os.getenv("HOSTNAME")
port = os.getenv("PORT")
uid = os.getenv("UID")
password = os.getenv("PASSWORD")

try:
    conn = ibm_db.connect(
        f"DATABASE={database_name};HOSTNAME={host_name};PORT={port
};SECURITY=SSL;SSLServiceCertificate=DigiCertGlobalRootCA.crt;UID={uid
};PWD={password}","",""
    )
    print(conn)
    print("connection successful...")
```

```
except:
   print("Connection Failed")
   print(ibm_db.conn_error())



@app.route('/')
def home():
   return render_template('index.html')

@app.route('/contacts')
def contacts():
   return render_template('contacts.html')

@app.route('/forgot')
def forgot():
   return render_template('forgotten-password.html')

@app.route('/signup', methods=['POST','GET'])
def signup():
   if request.method == 'POST':
      # conn = connection()
       try:
                                    sql = "INSERT INTO user_data
VALUES('{}','{}','{}','{}')".format(request.form["name"],request.form["email
"],request.form["phone"],request.form["password"])
          ibm_db.exec_immediate(conn,sql)
          #flash("successfully Registered !")
          return render_template('login.html')
```

```
        except:
            #flash("Account already exists! ")
            return render_template('signup.html')
    else:
            return render_template('signup.html')
        # name = request.form['name']
        #email = request.form['email']
        #phone = request.form['phone']
        #password = request.form['password']

        #sql ="INSERT INTO users VALUES (?,?,?,?)"
        #stmt = ibm_db.prepare(conn,sql)
        #ibm_db.bind_param(stmt, 1, name)
        #ibm_db.bind_param(stmt, 2, email)
        #ibm_db.bind_param(stmt, 3, phone)
        #ibm_db.bind_param(stmt, 4, password)
        #ibm_db.execute(stmt)
    # return render_template('signup.html')

@app.route('/login', methods=['POST','GET'])
def login():
    if request.method == 'POST':
        # conn =connection()
        email = request.form["email"]
        password = request.form["password"]
            sql = "SELECT COUNT(*) FROM user_data WHERE EMAIL=? AND
PASSWORD=?"
        stmt = ibm_db.prepare(conn,sql)
```

```
        ibm_db.bind_param(stmt, 1, email)
        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['email'] = email
            return render_template('job_post.html')
        else:
            #flash("email/ Password isincorrect! ")
            return render_template('login.html')
    else:
            return render_template('login.html')

        #email = request.form['email']
        #password = request.form['password']

        #sql = "SELECT * FROM users WHERE email=%s AND password=%s"
        #stmt = ibm_db.prepare(conn, sql)
        #ibm_db.bind_param(stmt,1,email)
        #ibm_db.bind_param(stmt,2,password)
       # user = ibm_db.execute(stmt).fetchone()

  #    return render_template('login.html' ,msg="success")

@app.route('/posts')
def posts():
```

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

@app.route('/addrec',methods=['POST','GET'])
def addrec():
        arr = []
        sql = "SELECT * FROM job_list"
        stmt = ibm_db.exec_immediate(conn,sql)
        dictionary = ibm_db.fetch_both(stmt)
        while dictionary != False:
         inst={}
         inst['DNAME']=dictionary['JOBNAME']
         inst['DTITLE']=dictionary['JOBTITLE']
         inst['DROLE']=dictionary['JOBROLE']
         inst['DESCRIPTION']=dictionary['JOBDESCRIPTION']
         arr.append(inst)
         dictionary = ibm_db.fetch_both(stmt)

        return render_template('list.html',arr=arr)

@app.route('/list')
def list():
        arr = []
        sql = "SELECT * FROM job_list"
        stmt = ibm_db.exec_immediate(conn,sql)
        dictionary = ibm_db.fetch_both(stmt)
        while dictionary != False:
         inst={}
         inst['DNAME']=dictionary['JOBNAME']
```

```
        inst['DTITLE']=dictionary['JOBTITLE']
        inst['DROLE']=dictionary['JOBROLE']
        inst['DESCRIPTION']=dictionary['JOBDESCRIPTION']
        arr.append(inst)
        dictionary = ibm_db.fetch_both(stmt)
       print(arr)
       return render_template('list.html',arr=arr)


if_name_=='_main_':
    app.run(debug=True)
```

GitHub & Project Demo Link

https://github.com/IBM-EPBL/IBM-Project-23741-1659924454

https://drive.google.com/file/d/1EDOvYZ--IxYsj0ObsEPYIZQYM8gR8pW-/view?usp=drivesdk