

**SKILL/JOB RECOMMENDER APPLICATION**  
**NALAIYA THIRAN PROJECT BASED LEARNING**  
**on**  
**PROFESSIONAL READINESS FOR INNOVATION,**  
**EMPLOYABILITY AND ENTREPRENEURSHIP**  
**A PROJECT REPORT**

<b>DINESH S</b>	<b>(711519BCS013)</b>
<b>MITHUNAVARSHINI A P</b>	<b>(711519BCS033)</b>
<b>HARIHARAN C</b>	<b>(711519BCS014)</b>
<b>YOGESH KUMAR S</b>	<b>(711519BCS063)</b>

**BACHELOR OF ENGINEERING**  
**IN**  
**COMPUTER SCIENCE AND ENGINEERING**  
**KIT - KALAI GNANAKARUNANIDHI INSTITUTE OF TECHNOLOGY**  
**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

## TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGENO
	ABSTRACT	4
1	INTRODUCTION	6
1.1	Project Overview	7
1.2	Purpose	7
2	Literature Survey	8
2.1	Existing Problem	9
2.2	Problem Statement	10
3	IDEATION& PROPOSED SOLUTION	11
3.1	Empathy Map Canvas	11
3.2	Ideation & Brainstorming	12
3.3	Proposed Solution	13
3.4	Problem Solution fit	15
4	REQUIREMENT ANALAYSIS	
4.1	Functional Requirements	16
4.2	Non-Functional Requirements	16
5	PROJECT DESIGN	
5.1	Data Flow Diagrams	17
5.2	Solution & Technical Architecture	18
5.3	User Stories	21
6	PROJECT PLANNING & SCHEDULING	

<b>6.1</b>	<b>Sprint Planning And Estimation</b>	<b>24</b>
<b>6.2</b>	<b>Sprint Delivery Schedule</b>	<b>27</b>
<b>7</b>	<b>CODING &amp; SOLUTIONG</b>	
<b>7.1</b>	<b>Feature 1</b>	<b>28</b>
<b>7.2</b>	<b>Feature 2</b>	<b>35</b>
<b>8</b>	<b>TESTING</b>	
<b>8.1</b>	<b>Test Cases</b>	<b>50</b>
<b>8.2</b>	<b>User Acceptance Testing</b>	<b>54</b>
<b>9</b>	<b>RESULTS</b>	
<b>9.1</b>	<b>Performance Metrics</b>	<b>56</b>
<b>10</b>	<b>CONCLUSION</b>	<b>58</b>
<b>11</b>	<b>REFERENCES</b>	<b>58</b>

## **ABSTRACT:**

Machine learning is a sub-field of data science that concentrates on designing algorithms that can learn from and make predictions on the data. Presently recommendation frameworks are utilized to take care of the issue of the overwhelming amount of information in every domain and enable the clients to concentrate on information that is significant to their area of interest. One domain where such recommender systems can play a significant role to help college graduates to fulfill their dreams by recommending a job based on their skill set. Currently, there are plenty of websites that provide heaps of information regarding employment opportunities, but this task is extremely tedious for students as they need to go through large amounts of information to find the ideal job. And many students are not aware of which job is suitable for them. Nowadays, the IT fields are in a boom. Many engineering students are learning some technical skills by doing some courses but they don't know which skill is for which job. Simultaneously, existing job recommendation systems only take into consideration the domain in which the user is interested while ignoring their profile and skillset, which can help recommend jobs that are tailor-made for the user. This paper examines the user's resume then compares the knowledge of degree, soft skills, hard skills, and the projects he has done and then only the system recommends the jobs for that user. The system not only recommends the jobs but also shows the score of his/her resume for the respective job. Then, the system also recommends skills to improve the scores of their Machine learning is a sub-field of data science that concentrates on designing algorithms that can learn from and make predictions on the data. Presently recommendation frameworks are utilized to take care of the issue of the overwhelming amount of information in every domain and enable the clients to concentrate on information that is significant to their area of interest. One domain where such recommender systems can play a significant role to help college graduates to fulfill their dreams by recommending a job based on their skill set. Currently, there are plenty of websites that provide heaps of information regarding employment opportunities, but this task is extremely tedious for students as they need to go through large amounts of information to find the ideal job. And many students are not aware of which job is suitable for them. Nowadays, the IT fields are in a boom. Many engineering students are learning some technical skills by doing some courses but they don't know which skill is for

which job. Simultaneously, existing job recommendation systems only take into consideration the domain in which the user is interested while ignoring their profile and skillset, which can help recommend jobs that are tailor-made for the user. This paper examines the user's resume then compares the knowledge of degree, soft skills, hard skills, and the projects he has done and then only the system recommends the jobs for that user. The system not only recommends the jobs but also shows the score of his/her resume for the respective job. Then, the system also recommends skills to improve the scores of their

Machine learning is a sub-field of data science that concentrates on designing algorithms that can learn from and make predictions on the data. Presently recommendation frameworks are utilized to take care of the issue of the overwhelming amount of information in every domain and enable the clients to concentrate on information that is significant to their area of interest. One domain where such recommender systems can play a significant role to help college graduates to fulfill their dreams by recommending a job based on their skill set. Currently, there are plenty of websites that provide heaps of information regarding employment opportunities, but this task is extremely tedious for students as they need to go through large amounts of information to find the ideal job. And many students are not aware of which job is suitable for them. Nowadays, the IT fields are in a boom. Many engineering students are learning some technical skills by doing some courses but they don't know which skill is for which job. Simultaneously, existing job recommendation systems only take into consideration the domain in which the user is interested while ignoring their profile and skillset, which can help recommend jobs that are tailor-made for the user. This paper examines the user's resume then compares the knowledge of degree, soft skills, hard skills, and the projects he has done and then only the system recommends the jobs for that user. The system not only recommends the jobs but also shows the score of his/her resume for the respective job. Then, the system also recommends skills to improve the scores of their

# **CHAPTER-1**

## **INTRODUCTION**

A recent report claims that most college graduates have difficulty in choosing their domain in their job. Many engineers are trying to shift the domain from their field to IT. So, they are doing some courses in online and randomly searching for a job. Nowadays, IT fields are the targets of many students but they don't know which domain is fit for them. To avoid this situation candidates, need a Job recommendation that analyses the skills to recommend a suitable job for the candidate. The solution is to design a system that reads a resume and their skills. The resumes are going through pre-processing to make the design more efficient. For pre-processing top words and porter Stemmer, Porter Stemmer will make every word their root word, and stop words will remove every meaningless word. This makes the system more efficient. Using of-if reflectorized for both resume and job description. Then compare the skills in the resume and description. For comparing, it uses the Cosine Similarity function and finds the scores of the resume for the respective jobs. Now it sorts the list in descending order with respect to their scores. Now, he got a hierarchical order of jobs from top to bottom. So, he can go with the first job or second which the skill he had already. He can be successful in that domain. The System not only shows the job but also recommends the skills to be improved for the job. Because of this, the candidate can train himself/herself for the future purpose and be a more achievable or talented person in his/her domain.

For comparing, it uses the Cosine Similarity function and finds the scores of the resume for the respective jobs. Now it sorts the list in descending order with respect to their scores. Now, he got a hierarchical order of jobs from top to bottom. So, he can go with the first job or second which the skill he had already. He can be successful in that domain. The System not only shows the job but also recommends the skills to be improved for the job. Because of this, the candidate can train himself/herself for the future purpose and be a more achievable or talented person in his/her domain.

## **1.1 PROJECT OVERVIEW:**

To find suitable jobs and their scores, this application receives the resume and has a dataset for a job with their description. It will pre-process the resume and job description with the stop words and porter's steamer. Then it reduces into a meaningful bag of words.

Now the application uses a of-id f reflectorized to convert a raw text into a matrix which makes it easy while compare. The main step is comparing the two bag words. For that, it uses the Cosine Similarity function, which is an angle dependent calculation. By using cosine, it has a list of jobs in descending order with respect to scores. The system will move on to the next progress which is finding the skills to be improved by the candidates. The system will take the resume and the skills dataset then compares both and display the skills which are all not in the resume. The major contribution of this work is as follows: The large MNC businesses use the mechanism currently in place for employment recommendations. The method is employed by businesses, not by regular people. If not, they will charge a small subscription fee to check the user's career options. The system functions for the average guy from city to village to modify this predicament. Because the students would look for employment based on their own skills, this approach will reduce unemployment. This company will also grow more quickly, which will result in more job openings.

## **1.2.PURPOSE:**

The dataset used for this research are sourced from Stack overflow survey data which is modeled as the user data for this research. Another dataset was created by web scrapping the Job board Using R programming language to fulfill the road map.

## **CHAPTER-2**

### **2 LITERATURE SURVEY**

#### **LITERATURE SURVEY 1:**

**NAME OF THE PAPER :** Job Recommendation based on Job Seeker Skills.

**NAME OF THE AUTHOR :** Jorge Valverde-Rebaza ,Ricardo Puma ,Paul Bustios,Nathalia C. Silva. **JOURNAL PUBLISHED :** First Workshop on Narrative Extraction From Text co-located with 40th European Conference on Information Retrieval.

**PUBLISHED MONTH :** March

**PUBLISHED YEAR** 2018

**OBJECTIVE OF THE PROJECT:**

- > In this ,when a candidate submits his/ her profile at a job seeker engine.
- > Their job recommendations are mostly suggested taking their academic qualification and work experience into considerations.

#### **LITERATURE SURVEY 2:**

**NAME OF THE PAPER :** A survey of job recommender systems.

**NAME OF THE AUTHOR :** Shaha Alotaibi.

**JOURNAL PUBLISHED :** International Journal of Physical Sciences

**PUBLISHED MONTH :** July

**PUBLISHED YEAR** 2012

**OBJECTIVE OF THE PROJECT:**

- > The recommender system technology aims to help users in finding items that match their personnel interests, it has a successful usage in e-commerce applications to deal with problems related to information overload efficiently.
- > This article will present a survey of e-recruiting process and existing recommendation approaches for building personalized recommender systems for candidates/job

#### **LITERATURE SURVEY 3:**

**NAME OF THE PAPER :** A Research of Job Recommendation System Based on Collaborative Filtering.

**NAME OF THE AUTHOR :** Cheng Yang, Yingya Zhang, Zhixiang Niu. **JOURNAL PUBLISHED :** 2014 Seventh International Symposium on Computation Intelligence and Design.

**PUBLISHED MONTH :** December

**PUBLISHED YEAR** 2014



## **2.1.EXISTING PROBLEM:**

The major contribution of this work is as follows: The large MNC businesses use the mechanism currently in place for employment recommendations. The method is employed by businesses, not by regular people. If not, they will charge a small subscription fee to check the user's career options. The system functions for the average guy from city to village to modify this predicament. Because the students would look for employment based on their own skills, this approach will reduce unemployment. This company will also grow more quickly, which will result in more job openings. The goal of the proposed work is to suggest a job that is ideal for the user. It displays the hierarchical jobs that are best for the user, not just one job.

Additionally, it suggests skills for the jobs that were suggested for the user. This project is intended for someone who simply has no idea what they are going to do. Additionally, there are no logins available because doing so increases the likelihood that users would reject you. The subsequent chapter goes over the specifics of the implementation. The rest of the paper organizes as follows: Chapter 2 provides the literature review conducted for this project. Chapter 3 presents the System Design and Architecture of the project along with the methodology. Chapter 4 discusses the algorithms proposed in this project. Chapter 5 presents the project conclusion and future works on this project.

## **2.2.PROBLEM STATEMENT:**

The dataset used for this research are sourced from Stack overflow survey data which is modeled as the user data for this research.

Another dataset was created by web scrapping the Job board Using python programming language to fulfill the road map of this dissertation.

The research question proposed by this research is "Can an efficient recommender system be modeled for the Job seekers which recommend Jobs with the user's skill set and job domain and also addresses the issue of cold start?".

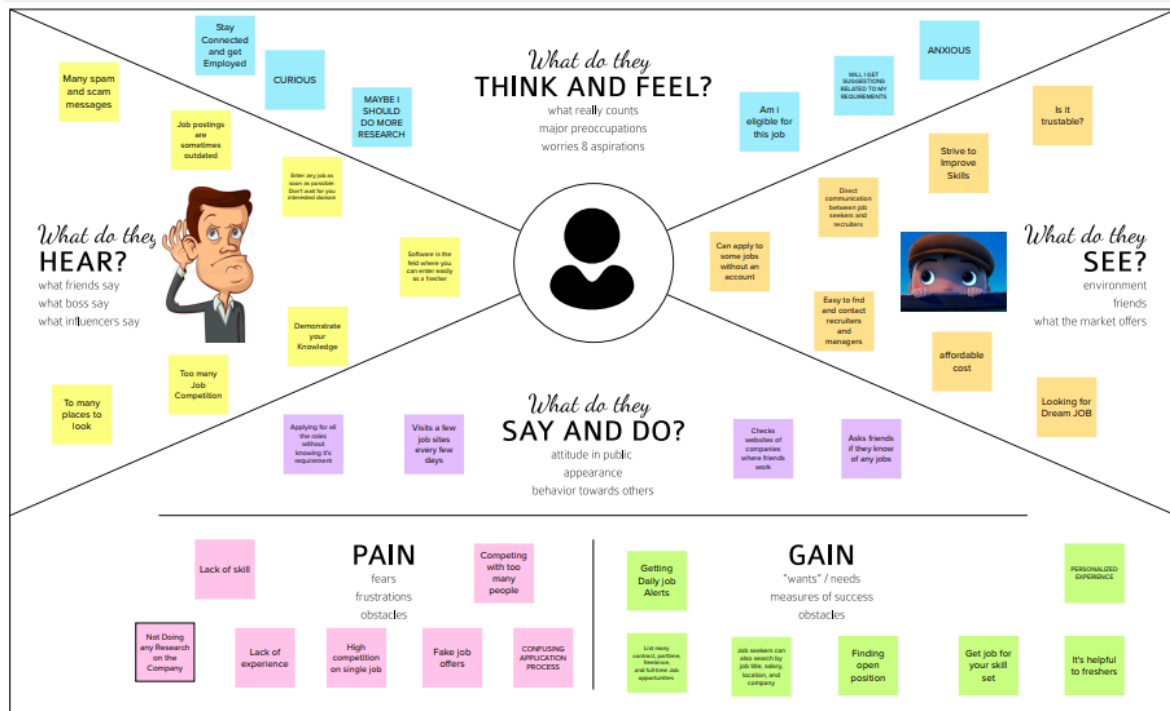
To answer the research question, below are the objectives that need to be satisfied with going forward

## CHAPTER-3

### IDEATION & PROPOSED SOLUTION

#### 3.1. EMPATHY MAP:

Build empathy and keep your focus on the user by putting yourself in their shoes.



## 3.2.Ideation & Brain Storming:

### Brainstorm & idea prioritization

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

- 10 minutes to prepare
- 1 hour to collaborate
- 2-3 people recommended

[Show template feedback](#)

#### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

- 1. Team gathering: Gather the ideal participants in the session and send an invite. Share relevant information or pre-work ahead.
- 2. Set the goal: Think about the problem you'll be focusing on solving in this brainstorming session.
- 3. Warm-up: Use the following facilitation tools like the Random Prompter to set a happy and productive session.

[Open article](#)

#### Define your problem statement

What problem are you trying to solve? Frame your problem as a clear, right-like statement. This will be the focus of your brainstorm.

[Exercise](#)

**Problem**

The search provided over the **random prompter** is **inspiration** for a **number of fields for research**.

**Key rules of brainstorming**

Focus on **ideas** and **productive** ideas.

- Stay in topic
- Encourage wild ideas
- Defer judgment
- Quantity over quality
- Go for volume
- If possible, be visual

#### Brainstorm

Write down any ideas that come to mind that address your problem statement.

[10 minutes](#)

**1. Researcher**

1. Researcher

2. Researcher

3. Researcher

4. Researcher

**2. Researcher**

1. Researcher

2. Researcher

3. Researcher

4. Researcher

**3. Researcher**

1. Researcher

2. Researcher

3. Researcher

4. Researcher

**4. Researcher**

1. Researcher

2. Researcher

3. Researcher

4. Researcher

[Show template feedback](#)

Whether brainstorming, ideation or collaborative sessions are your go-to, these will help you generate ideas in a structured and efficient way. If you're looking for a way to generate ideas, these are the best tools to use.

#### Guarantee

100% of the ideas generated in this session will be used in the final product.

#### Expectations

100% of the ideas generated in this session will be used in the final product.

#### Brainstorming

Brainstorming is a process of generating ideas. It's a collaborative session where team members share their ideas and build on each other's ideas. It's a great way to generate ideas and build on each other's ideas.

[Show template feedback](#)

#### Brainstorming

Brainstorming is a process of generating ideas. It's a collaborative session where team members share their ideas and build on each other's ideas. It's a great way to generate ideas and build on each other's ideas.

[Show template feedback](#)

#### Brainstorming

Brainstorming is a process of generating ideas. It's a collaborative session where team members share their ideas and build on each other's ideas. It's a great way to generate ideas and build on each other's ideas.

[Show template feedback](#)

#### Brainstorming

Brainstorming is a process of generating ideas. It's a collaborative session where team members share their ideas and build on each other's ideas. It's a great way to generate ideas and build on each other's ideas.

[Show template feedback](#)

### **3.3.PROPOSED SOLUTION:**

S. No	Parameter	Description
1	Problem Statement (Problem to be solved)	Having lots of skills but wondering which job will best suit you? Don't need to worry! We have come up with a skill recommender solution through which the fresher or the skilled person can log in and find the jobs by using the search option or they can directly interact with the chatbot and get their dream job. To develop an end-to-end web application capable of displaying the current job openings based on the user skillset. The user and their information are stored in the Database. An alert is sent when there is an opening based on the user skillset. Users will interact with the chatbot and can get the recommendations based on their skills. We can use a job search API to get the current job openings in the market which will fetch the data directly from the webpage
2	Idea / Solution description	The contributions of this work are threefold, we: i) made publicly available a new dataset formed by a set of job seekers profiles and a set of job vacancies collected from different job search engine sites ii) put forward the proposal of a framework for job recommendation based on professional skills of job seekers iii) carried out an evaluation to quantify recommendation abilities of two state-of-the art methods, considering different configurations, within the proposed framework. We thus present a general

		panorama of job recommendation task aiming to facilitate research and real-world application design regarding this important issue
3	Novelty / Uniqueness	The best position are suggested to any person according to her skills. While the position of known profiles are assumed should be noted that there are usually multiple advisable positions corresponding to a set of skills. A recommendation system should return a set of most likely positions and all of them can be equally valid. The recommendation method we use is simply based on representing both positions and profiles as comparable vectors and seeking for each profile the positions with the most similar vectors.
4	Social Impact / Customer Satisfaction	Students will be benefited as they will get to know which job suits them based on their skill set and therefore Lack of Unemployment can be reduced.
5	Business Model (Revenue Model)	We can provide the application for job seekers in a subscription based and we can share the profiles with companies and generate the revenue by providing them best profiles.
6	Scalability of the Solution	Data can be scaled up and scaled down according to number of current job openings available

### 3.4. PROBLEM SOLUTION FIT:

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why

Purpose:

- ☐ Solve complex problems in a way that fits the state of your customers.
- ☐ Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.
- ☐ Sharpen your communication and marketing strategy with the right triggers and messaging.
- ☐ Increase touch-points with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.
- ☐ Understand the existing situation in order to improve it for your target group.

Template:

Define CS, fit into CC	<b>1. CUSTOMER SEGMENTS</b>  1) Jobless people 2) New college grads	<b>6. CUSTOMER CONSTRAINTS</b>  For the website to operate as intended, basic needs such as an internet connection and laptop are required.	<b>5. AVAILABLE SOLUTIONS</b>  Earlier, job seekers used TV adverts and paper columns, as a result of the expanding digital world, the use of suggestion websites.	Explore AS, differentiate
focus on J&P, tap into BE	<b>2. JOBS-TO-BE-DONE/PROBLEM</b>  Make some work recommender site with an inbuilt chatbot help	<b>9. PROBLEM ROOT CAUSE</b>  The vast majority don't know about their positions accessible in the market/sites	<b>7. BEHAVIOURS</b>  The users attempt to first analyse job searches on websites, papers, and adverts depending on their requirements.	focus on J&P, tap into BE
Identify strong TR&EM	<b>3. TRIGGERS</b> Seeing other find a new line of work  <b>4. EMOTIONS: BEFORE/AFTER</b> User will be satisfied with the services and higher possibility of job offer	<b>10. YOUR SOLUTION</b>  To build a platform that helps freshers and under graduates to get a job	<b>8. CHANNELS OF BEHAVIOUR</b>  ONLINE : Ready to explore a suitable job based on their skill sets and necessities  OFFLINE : Attend interviews on-site and try and get a job	Identify strong TR&EM

## **CHAPTER-4**

### **REQUIREMENT ANALYSIS:**

#### **4.1 FUNCTIONAL REQUIREMENTS:**

S. No	FUNCTIONAL REQUIREMENT (Epic)	SUB REQUIREMENT (Story)
1.	Sign In / Login	Register with username, password
2.	Profile Registration	Register with username, password, email, qualification, skills. This data will be stored in a database.
3.	Job profile display	Display job profiles based on availability, location ,skills
4.	Chatbot	A chat on the webpage to solve user queries and issues
5.	Job registration	A copy of the company the user applied for with its registration/description details will be sent to the registered email id
6.	Logout	

#### **4.2.NON-FUNCTIONAL REQUIREMENTS:**

S. No	NON-FUNCTIONAL REQUIREMENT	DESCRIPTION
1.	Usability	The webpage will be designed in such a way that any non-technical user can easily navigate through it and complete the job registration work. (Easy and Simple design.)
2.	Security	Using of SSL certificate will provide security to the project. Database will be safely stored in DB2.
3.	Reliability	To make sure the webpage doesn't go down due to network traffic.



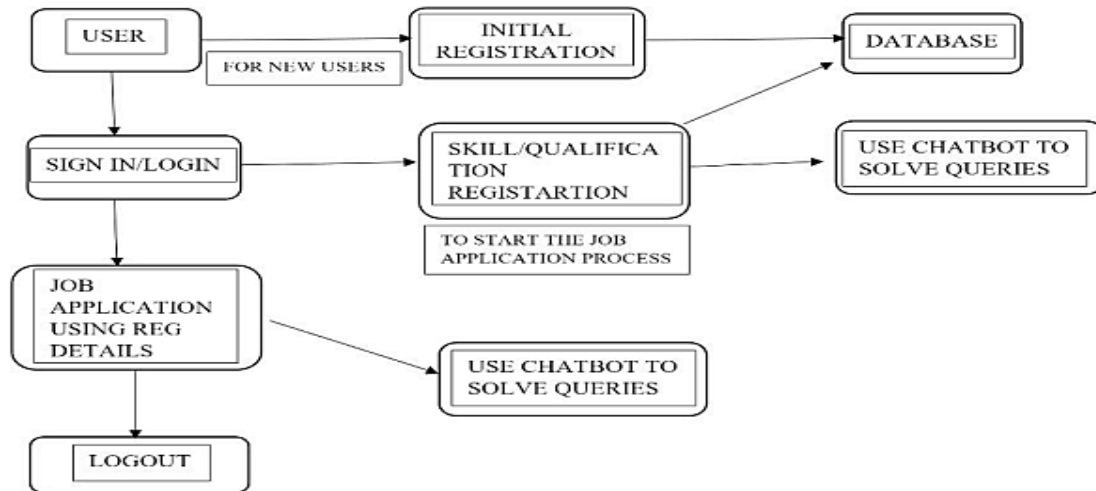
4.	Availability	This webpage will be available to all users (network connectivity is necessary) at any given point of time
5.	Scalability	Increasing the storage space of database can increase the number of users. Add some features in future to make the webpage unique and attractive
6.	Performance	Focus on loading the webpage as quickly as possible irrespective of the number of user/integrator traffic

## **CHAPTER-5**

### **PROJECT DESIGN**

#### **5.1.DATA FLOW DIAGRAMS:**

Data Flow Diagrams: A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

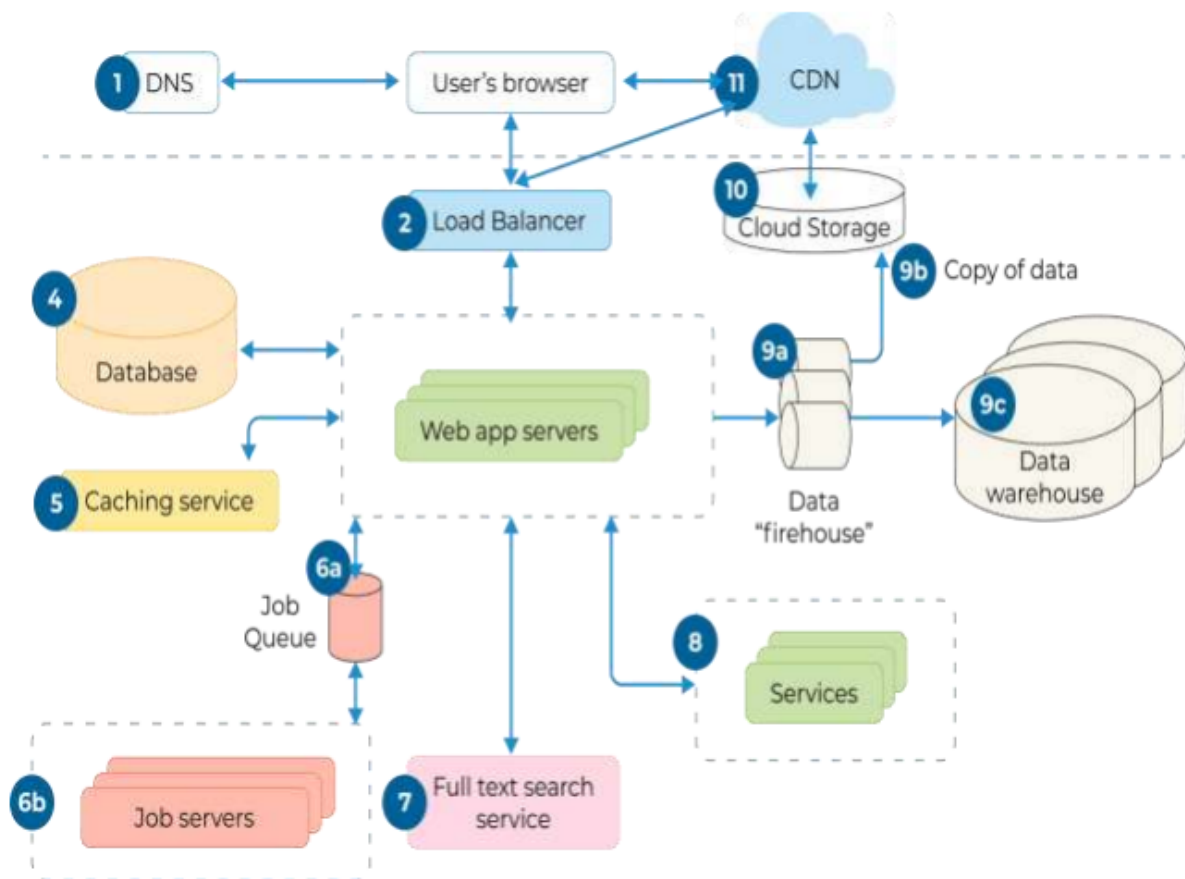


It shows how data enters and leaves the system, what changes the information, and where data is stored.

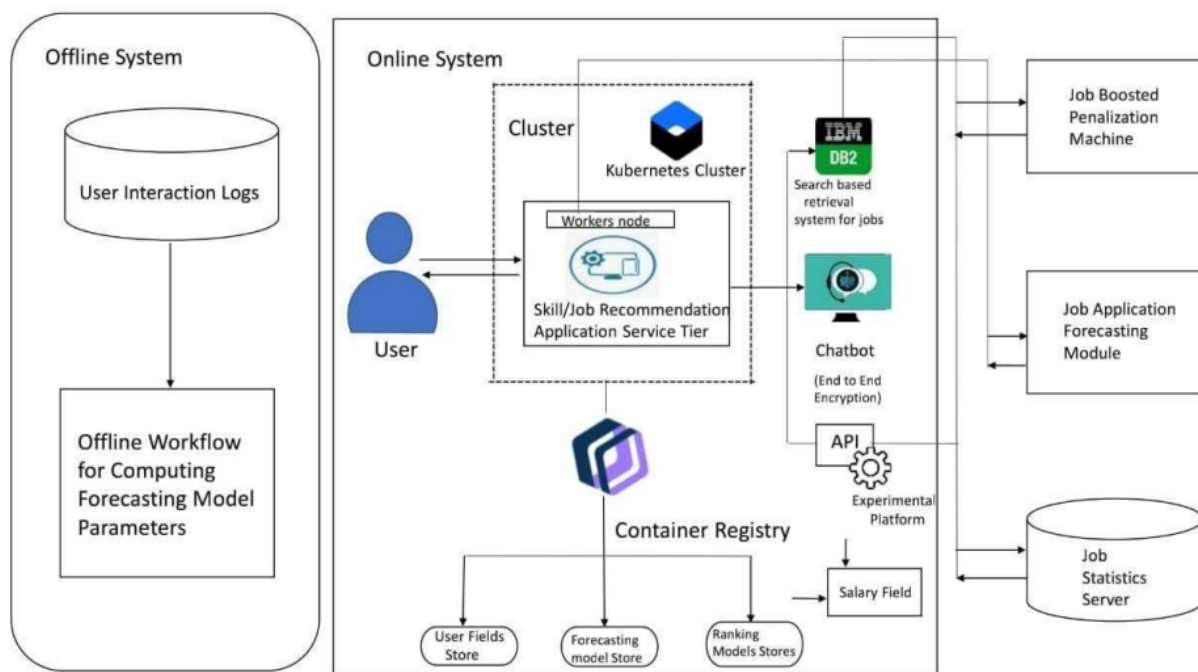
The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

## **5.2.SOLUTION & TECHNICAL ARCHITECTURE:**

### **Solution architecture:**



## Technical architecture:



S. No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc	HTML, CSS, JavaScript / Angular Js / React Js etc
2.	Developing Interface	Developing application for the task	Java / Python
3.	Voice Assistance	Voice commands instead of typing	IBM Watson STT service
4.	Chatbot Assistance	Conversational Interface	IBM Watson Assistant
5.	Database	Data Type, Configurations etc	MySQL, NoSQL, etc
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local File system

8.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

### **5.3.USER STORIES:**

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	I can receive confirmat	High	Sprint -1

			receive confirmat ion email once I have register ed for the applicati on	ion email & click confirm		
		USN-3	As a user, I can register for the applicati on through Facebo ok	I can register & access the dashboa rd with Facebo ok Login	Low	Sprint-2
		USN-4	As a user, I can register for the applicati on through Gmail	I can receive confirmat ion email & click confirm	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the applicati on by entering email & password	I can access my account / dashboa rd	High	Sprint-1
	Dashboa rd	USN-6	Create a model	Assign that	High	Sprint-1

			set that contains those models, then assign it to a role	group to the appropriate roles on the Roles page		
Customer (Web user)	Identity-aware	USN-7	Open, public access, User-authenticated access, Employee e-restricted access.	Company public website. App running on the company intranet. App with access to customer private information.	High	Sprint-1
Customer Care Executive	Communication	USN-8	A customer care executive is a professional responsible for communicating the how's and why's regarding service expectations	For how to tackle customer queries	Medium	Sprint-1

			within a company			
Administrator	Device management	USN-9	You can Delete/Disable/Enable devices in Azure Active Directory but you cannot Add/Remove Users in the directory.	Ease of use	Medium	Sprint-1

## **CHAPTER-6**

### **PROJECT PLANNING & SCHEDULING**

#### **6.1 SPRINT PLANNING AND ESTIMATION:**

Sprint	Functional Requirement (Epic	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Members
Sprint-1	Registration	USN - 1	As a user, I can register for the application by	I can access my account / dashboard	High	Chaduvu. Viswanth Kumar Goli.siva kesava



			entering my email, password, and confirming my password			Reddy
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	Bathula.r ama njanyeu lu Kottam.U pendr a
Sprint-2		USN - 3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Chaduvu. Vis wanth Kumar USN - 9 Goli.siva kesava Reddy
Sprint-3		USN - 4	As a user, I can register for the application through Gmail	I can receive confirmation email & click confirm	Medium	Chaduvu. Vis wanth Kumar Goli.siva kesava Reddy
Sprint-2	Login	USN - 5	As a	I can	High	

			user, I can log into the application by entering email & password	access my account / dashboard		
Sprint-2	Dashboard	USN - 6	Create a model set that contains those models, then assign it to a role	Assign that group to the appropriate roles on the Roles page	High	Bathula.r ama njanyeu lu Kottam.U pendr a
Sprint-4	Identity-Aware	USN - 7	Open, public access, User□authenticated access, Employee□restricted access	Company public website. App running on the company intranet. App with access to customer private information	High	Bathula.r ama njanyeu lu Kottam.U pendr a
Sprint-1	Communication	USN - 8	A customer care executive is a professional responsi	For how to tackle customer queries	Medium	Bathula.r aUSN - 9 ma njanyeu lu Kottam.U pendr a

			ble for communicating the how's and why's regarding service expectations within a company			
Sprint-3	Device management	USN - 9	You can Delete/Disable/Enable devices in Azure Active Directory but you cannot Add/Remove Users in the directory	Ease of use.	Medium	Chaduvu. Viswanth Kumar Goli.siva Kesava Reddy

## **6.2 Sprint Delivery Schedule:**

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	30 Oct 2022	04 Nov 2022	20	04 Nov 2022
Sprint-2	20	6 Days	04 Nov 2022	09 Nov 2022	18	09 Nov 2022
Sprint-3	20	6 Days	09 Nov 2022	14 Nov 2022	20	14 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	19	19 Nov 2022

## 7.CODING & SOLUTIONING

### 7.1. FEATURE-1(SPRINT-1)

#### IBM.HTML:

```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>Home</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>

<body>

  <nav class="navbar navbar-expand-lg bg-light" style="background-
color: #E9FDF5;">
    <!-- Navbar content -->

    <div class="container-fluid">
      <div class="collapse navbar-collapse"
id="navbarNavAltMarkup">
        <div class="navbar-nav">

```

```

                                <a class="nav-link active" aria-current="page"
href="#">Home</a>
                                <a class="nav-link"
href="{{url_for('about')}}">About</a>
                                <a class="nav-link"
href="{{url_for('signin')}}">SignIn</a>
                                <a class="nav-link"
href="{{url_for('signup')}}">SignUp</a>
                                </div>
                            </div>
                        </div>
                    </nav>
                    <br><br>
                    <div>
                        <h4>
                            <b>Welcome to Project</B>
                        </h4>
                    </div>

</body>

</html>

```

## ABOUT.HTML:

```

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width,initial-scale=1"
/>
        <title>Home Page</title>
        <link
            href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta2/dist/css/bootstrap.min.css"
            rel="stylesheet"
        />

```

```

</head>

<body>
  <nav
    class="navbar navbar-expand-lg bg-light"
    style="background-color: #e3f2fd"
  >
    <!-- Navbar content -->

    <div class="container-fluid">
      <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
          <a class="nav-link active" aria-current="page"
href="#">Home</a>
          <a class="nav-link" href="about.html">About</a>
          <a class="nav-link" href="signin.html">SignIn</a>
          <a class="nav-link" href="signup.html">SignUp</a>
        </div>
      </div>
    </div>
  </nav>
  <br /><br />
  <div>
    <h4>
      <b
        >Welcome to Job Seeker !! here you can find the jobs that you
need and
        fit for your resume and your skills !!THE MORE SKILLS THE
MORE
        RECOMMENDATIONS!!
      </b>
    </h4>
  </div>
</body>
</html>

```

## SIGNUP.HTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1"
  />
    <title>SignIn</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta2/dist/css/bootstrap.min.css"
      rel="stylesheet"
    />
  </head>

  <body>
    <nav
      class="navbar navbar-expand-lg bg-light"
      style="background-color: #e3f2fd"
    >
      <!-- Navbar content -->

      <div class="container-fluid">
        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
          <div class="navbar-nav">
            <a class="nav-link" href="home.html">Home</a>
            <a class="nav-link" href="about.html">About</a>
            <a class="nav-link active" aria-current="page"
href="#">SignIn</a>
            <a class="nav-link" href="signup.html">SignUp</a>
          </div>
        </div>
      </div>
    </nav>
    <div
```

```

class="text-center my-5"
style="
background-image:
url('https://png.pngtree.com/thumb_back/fh260/background/20200714/pngtree-modern-double-color-futuristic-neon-background-image_351866.jpg');
background-repeat: no-repeat;
background-size: cover;
"
>
<section class="h-100">
<div class="container h-100">
<div class="row justify-content-sm-center h-100">
<div class="col-xxl-4 col-xl-5 col-lg-5 col-md-7 col-sm-9">
<div class="text-center my-5"></div>
<div class="card shadow-lg">
<div class="card-body p-5">
<h1 class="fs-4 card-title fw-bold mb-4">Submit</h1>
<form
method="POST"
class="needs-validation"
novalidate=""
autocomplete="off"
>
<div class="mb-3">
<label class="mb-2 text-muted" for="email">
>E-Mail Address</label>
>
<input
id="email"
type="email"
class="form-control"
name="email"
value=""
required
autofocus
/>

```



```
<div class="invalid-feedback">Email is invalid</div>
</div>
```

```
<div class="mb-3">
  <div class="mb-2 w-100">
    <label class="text-muted" for="password">
      >Password</label>
    >
    <!--a href="forgot.html" class="float-end">
      Forgot Password?
    </a-->
  </div>
  <input
    id="password"
    type="password"
    class="form-control"
    name="password"
    required
  />
  <div class="invalid-feedback">Password is required</div>
</div>
```

```
<div class="d-flex align-items-center">
  <div class="form-check">
    <input
      type="checkbox"
      name="remember"
      id="remember"
      class="form-check-input"
    />
    <label for="remember" class="form-check-label">
      >Remember Me</label>
    >
  </div>
  <button type="submit" class="btn btn-primary ms-auto">
    Submit
  </button>
</div>
```



```

        <!-- Navbar content -->

        <div class="container-fluid">
            <div class="collapse navbar-collapse"
id="navbarNavAltMarkup">
                <div class="navbar-nav">
                    <a class="nav-link active" aria-current="page"
href="#">Home</a>
                    <a class="nav-link"
href="{{url_for('about')}}">About</a>
                    <a class="nav-link"
href="{{url_for('signin')}}">SignIn</a>
                    <a class="nav-link"
href="{{url_for('signup')}}">SignUp</a>
                </div>
            </div>
        </div>
    </nav>
    <br><br>
    <div>
        <h4>
            <b>Welcome to IBM!!!</B>
        </h4>
    </div>

</body>

</html>

```

## 7.2.FEATURE-2(SPRINT-2).

### CONIGURE.PY:

```

config.py
# Saved file for each job info
JOBS_INFO_JSON_FILE = r'./data/indeed_jobs_info.json'
# Path to sample resume
SAMPLE_RESUME_PDF_DIR = r'./data/'

```

## FUNCTIONFORJOBRECOMMENDE.PY:

```
from functools import reduce
import re
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import PyPDF2
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from collections import Counter
import numpy as np

pd.options.mode.chained_assignment = None

# Skill dictionary used for the project
SKillDictionary = ['bash', 'r', 'python', 'java', 'c++', 'ruby', 'perl', 'matlab',
'javascript', 'scala', 'php',
                    'jquery', 'angularjs', 'excel', 'tableau', 'sas', 'spss', 'd3',
'saas', 'pandas', 'numpy', 'scipy',
                    'sps', 'spotfire', 'scikit', 'splunk', 'power', 'h2o', 'pytorch',
'tensorflow', 'caffe', 'caffe2',
                    'cntk', 'mxnet', 'paddle', 'keras', 'bigdl', 'hadoop',
'mapreduce', 'spark', 'pig', 'hive', 'shark',
                    'oozie', 'zookeeper', 'flume', 'mahout', 'etl', 'aws', 'azure',
'google', 'ibm', 'agile', 'devops',
                    'scrum', 'agile', 'devops', 'scrum', 'sql', 'nosql', 'hbase',
'cassandra', 'mongodb', 'mysql',
                    'mssql', 'postgresql', 'oracle', 'rdbms', 'bigquery']

# creating a dataframe to add job description list
JobDescriptionDataframe = pd.DataFrame()

# class for job recommendation using dynamic weightage on Implicit and
```

Explicit skills of Job description.

class FunctionsForJobRecommendation:

```
# Init to convert job description list to a dataframe
def __init__(self, jobs_list):
    pd.set_option('display.max_columns', None)
    pd.set_option('display.max_rows', None)
    self.JobDescriptionDataframe = pd.DataFrame(jobs_list)

# Function to extract keywords extracted and filtered by using Skill
dictionary
def ExtractKeywords(self, text):
    text = text.lower()
    text = re.sub(r"(<>/)", ', ', text) # substitute (<>&/ to comma and
space
    text = re.sub(r"&", 'and', text) # substitute (<>&/ to comma and
space
    text = re.sub(r"[?!]", ' . ', text) # substitute ?! to dot and space
    text = re.sub(" [a-z0-9]+[.\'-a-z0-9_]*[a-z0-9]+@\w+\.com", "", text)
# substitute email address to dot
    text = re.sub(' +', ' ', text) # replace multiple whitespace by one
whitespace
    text = text.lower().split()
    stops = set(stopwords.words("english")) # Filter out stop words in
english language
    text = [w for w in text if not w in stops]
    text = list(set(text))

# Skills are extracted from the preprocessed text
# keywords extracted and filtered by using Skill dictionary
Keywords = [str(word) for word in text if word in SKillDictionary]
return Keywords

# Function to use counter to count the frequency of the keywords
def CountKeywords(self, keywords, counter):
    KeywordCount = pd.DataFrame(columns=['Freq'])
    for EachWord in keywords:
```

```

        KeywordCount.loc[EachWord] = {'Freq': counter[EachWord]}
    return KeywordCount

# Function to extract skill keywords from job description
def ExtractJobDescKeywords(self):
    # removing duplicate Jobs
    self.JobDescriptionDataframe.drop_duplicates(subset=['desc'],
inplace=True, keep='last', ignore_index=False)
    # Extract skill keywords from job descriptions and store them in a
new column 'keywords'
    self.JobDescriptionDataframe['keywords'] =
[self.ExtractKeywords(job_desc) for job_desc in
                                self.JobDescriptionDataframe['desc']]

# Function to extract resume keywords from resume
def ExtractResumeKeywords(self, resume_pdf):
    # Open resume PDF
    Resume = open(resume_pdf, 'rb')
    # creating a pdf reader object
    ReadResume = PyPDF2.PdfFileReader(Resume)
    # Read in each page in PDF
    ResumeContext = [ReadResume.getPage(x).extractText() for x in
range(ReadResume.numPages)]
    # Extract key skills from each page
    ResumeKeywords = [self.ExtractKeywords(page) for page in
ResumeContext]
    # Count keywords
    ResumeFrequency = Counter()
    for item in ResumeKeywords:
        ResumeFrequency.update(item)
    # Get resume skill keywords counts
    ResumeSkilllist = self.CountKeywords(SkillDictionary,
ResumeFrequency)
    return ResumeSkilllist[ResumeSkilllist['Freq'] > 0]

# Cosine similarity function to calculate cosine score between two
documents

```

```

def CalculateCosineSimilarity(self, documents):
    Countvectorizer = CountVectorizer()
    Matrix = Countvectorizer.fit_transform(documents)
    DocumentMatrix = Matrix.todense()
    df = pd.DataFrame(DocumentMatrix,
                       columns=Countvectorizer.get_feature_names(),
                       index=['ind1', 'ind2'])
    return cosine_similarity(df)[0][1]

# Function to calculate similarity and pick top10 jobs that match the
resume
def CalculateSimilarity(self, ResumeSkillList):
    # copy of job description dataframe as JobDescriptionSet
    JobDescriptionSet = self.JobDescriptionDataframe.copy()
    # To calculate similarity between resume skills and skills extracted
from job description
    for ind, x in JobDescriptionSet.iterrows():
        JobDescriptionString = ' '.join(map(str, x.keywords))
        ResumeKeywordString = ' '.join(map(str, ResumeSkillList))
        documents = [JobDescriptionString, ResumeKeywordString]
        # Created a column 'cosinescore' to store cosine score for top10
jobs
        JobDescriptionSet.loc[ind, 'cosinescore'] =
self.CalculateCosineSimilarity(documents)
        # to sort the top10 description based on cosine score
        MainTop10JDs = JobDescriptionSet.sort_values(by='cosinescore',
ascending=False).head(10)
        return MainTop10JDs

# Function to extract top20 Job description for each of the top10 jobs
to get implicit skills
def Extract20SimilarJDs(self, dynStat, MainTop10JDs,
ResumeSkillList):

    JobDescriptionSet = self.JobDescriptionDataframe.copy()
    SimilarJobIdsDataframe = pd.DataFrame()
    SimilarJobIdsDataframe.loc[0, 'similarJDs'] = 'NaN'

```

```

count2 = 0
finalSkillWeightList = []

# Iterate through each of the top 10 Jobs to extract similar 20 JDs
for ind, x in MainTop10JDs.iterrows():
    # variables for GraphPlot function ##
    impSkillCountResumeMatch = 0
    ImpSkillWeightCount = 0
    implicitSkillList = []
    implicitSkillWeightList = []

    # To extract each JD keyword set
    PickedJobDescriptionString = ' '.join(map(str, x.keywords))
    JDKeywordsSet = set(x.keywords)

    # To pick the common skills between resume and TopJD and
    added them to exSkillCountResumeMatch list##
    intersection = JDKeywordsSet.intersection(ResumeSkillList)
    exSkillCountResumeMatch = len(intersection)

    # Variable declared to calculate 20 similar Job description for
    each of Top10 Jobs
    rows = []
    count2 = count2 + 1
    # Iterate through the whole job description dataset to pick 20
    similar Job description for each Top10 Jobs
    for ind2, x2 in JobDescriptionSet.iterrows():
        # To skip the topJD within the job description
        if ind == ind2:
            continue
        JobDescriptionString = ' '.join(map(str, x2.keywords))
        # to calculate cosine score between topJD skills and
        pickedJD
        documents = [JobDescriptionString,
PickedJobDescriptionString]
        rows.append([ind2,
self.CalculateCosineSimilarity(documents)])

```



```

        # create a dataframe column for each of 20 similar Jds to
store their cosine score
        SimilarJobIdsDataframe['JD'] = ind2
        SimilarJobIdsDataframe['cosScore'] =
self.CalculateCosineSimilarity(documents)
        rows.sort(key=lambda i: i[1], reverse=True)
        count = 0
        JobDescriptionString = ' '
        for row in rows:
            indexval = 'JDind' + str(count)
            count = count + 1
            MainTop10JDs.loc[ind, indexval] = row[0]
            JobDescriptionString = JobDescriptionString + ' ' + ' '.join(
                map(str, JobDescriptionSet.keywords[MainTop10JDs.at[ind,
indexval]]))
        # set a threshold to collect top20 JobIds for each of
Top10Jobs
        if count > 20:
            break
        # Create a dataframe 'skill_list' to store the implicit skills of
top20 JDs for each top Job
        MainTop10JDs.loc[ind, 'skill_list'] = JobDescriptionString

        # Assign skill_list to WordList to assign static and dynamic
weightage.
        WordList = MainTop10JDs.loc[ind, 'skill_list']
        WordList = WordList.split()
        ImplicitWeight = 10

        # For Graph plot function ####
        skillList = []
        for implicitSkill in np.unique(np.array(WordList)):
            if implicitSkill in ResumeSkillList:
                if implicitSkill not in x.keywords:
                    impSkillCountResumeMatch =
impSkillCountResumeMatch + 1
                # implicitSkillList is the list of implicit skills which are

```

also present in resume

```
        implicitSkillList.append(implicitSkill)
    MainTop10JDs.loc[ind, 'exSkillCountResumeMatch'] =
exSkillCountResumeMatch
    MainTop10JDs.loc[ind, 'impSkillCountResumeMatch'] =
impSkillCountResumeMatch

    # for each implicit skill and its term frequency in the implicit skill
list
    for word, freq in Counter(WordList).items():
        if word in MainTop10JDs.keywords[ind]:
            continue
        # For dynamic approach, assign weightage based on term
frequency. Higher the count of the term present in the skilllist, higher
the weightage.
        if (dynStat == 1):
            tmpList = (word, freq / sum(Counter(WordList).values())) *
ImplicitWeight)
            if word in implicitSkillList:
                ImpSkillWeightCount = ImpSkillWeightCount + tmpList[1]

        # For static approach, setting weight to 1 and disabling
dynamic weight
        else:
            tmpList = (word, 1)
            if word in implicitSkillList:
                ImpSkillWeightCount = ImpSkillWeightCount + tmpList[1]
            skillList.append(tmpList)

    # For Graph plot function
    if dynStat == 1:
        for skill, weight in skillList:
            if skill in implicitSkillList:
                implicitSkillWeightList.append((skill, weight))
            finalSkillWeightList.append((ind, implicitSkillWeightList))

    # Assign weightage of 1 to explicit skills for both static and
```

dynamic approach

```
    top10keywords = MainTop10JDs.keywords[ind]
    exSkillList = []
    for skill in top10keywords:
        tmpList = (skill, 1)
        exSkillList.append(tmpList)
    MainTop10JDs.keywords[ind] = exSkillList
    MainTop10JDs.keywords[ind] = MainTop10JDs.keywords[ind] +
skillList
    sorted(MainTop10JDs.keywords[ind], key=lambda x: x[1],
reverse=True)

    # top_10_jd_matches - to return top10 Jobs with 20 similar JD for
each top Job and their skill weightage.
    # finalSkillWeightList - for Graph plot function, pick the implicit
skills which match the resume along with its dynamic weightage.
    return MainTop10JDs, finalSkillWeightList

# Function to calculate final cosine score for each top Job using
weighted cosine similarity and rank them according to the cosine score.
def WeightedCosineSimilarity(self, ResumeSkillList, Implicit):
    rsmSkillList = []
    # adding wightage of 1 to resume skill list as they should be given
high priority
    for skill in ResumeSkillList:
        rsmSkillList.append((skill, 1))
    # For each of the Top 10 Jobs
    for ind, x in Implicit.iterrows():
        # Create one dictionary for resume skill list and another for job
description skills(Implicit +explicit)
        d1 = dict(rsmSkillList)
        d2 = dict(Implicit.keywords[ind])
        # Using weightage cosine similarity because the weightage
differ based on term frequency for implicit skills in dynamic approach
        allkey = reduce(set.union, map(set, map(dict.keys, [d1, d2])))
        v1 = np.zeros((len(allkey),))
        k = 0
```

```

    for i in allkey:
        if i in d1.keys():
            v1[k] = d1[i]
            k = k + 1
    v2 = np.zeros((len(allkey),))
    k = 0
    for i in allkey:
        if i in d2.keys():
            v2[k] = d2[i]
            k = k + 1
    # v1 and v2 are 1-d np arrays representing resume skill list and
    job description skills
    v1 = (v1 / np.sqrt(np.dot(v1, v1))) ## normalized
    v2 = (v2 / np.sqrt(np.dot(v2, v2))) ## normalized

    Implicit.loc[ind, 'final_cosine'] = np.dot(v1, v2)
    # sort values based on cosine score
    Implicit = Implicit.sort_values(by='final_cosine',
ascending=False)
    Implicit.reset_index(inplace=True)
    Implicit = Implicit.rename(columns={'index': 'Jobid'})
    # return dataframe which consists of final cosine score calculated
    using dynamic weightage and ranked top10 JDs that best match the
    resume.
    return Implicit

# Function to plot graphs for evaluation of the proposed approach

def AllGraphPlotsForEvaluation(self, StaticGraph, DynamicGraph,
finalSkillWeightList, dynStat):

    for dynStat in range(0, 2):
        if (dynStat == 0):
            ImplicitGraph = StaticGraph
        else:
            ImplicitGraph = DynamicGraph

```

```

# create a scaler object for normalizing data points
scaler = MinMaxScaler()
df_norm = pd.DataFrame(scaler.fit_transform(ImplicitGraph),
columns=ImplicitGraph.columns)
ImplicitGraph['final_cosine'] = df_norm['final_cosine']

# Scatter plot for graph showing difference in cosine score
size = np.array([])
for x in ImplicitGraph['final_cosine']:
    size = np.append(size, x * 1000)
plt.scatter(x=ImplicitGraph['final_cosine'],
y=ImplicitGraph['Jobid'], s=size,
            c=ImplicitGraph['final_cosine'], cmap='viridis',
alpha=0.5)
plt.colorbar(label='Normalized cosine score')
# Creating comparative bar plot for implicit and explicit skill
count for referenced and proposed solution
# creating a list of all inputs:
# Jobid
# expcount- count of the explicit skills of the job description
which match the resume
# impcount - count of implicit skills of the job description which
match the resume
index = ImplicitGraph['Jobid'].tolist()
expCount = ImplicitGraph['exSkillCountResumeMatch'].tolist()
impCount = ImplicitGraph['impSkillCountResumeMatch'].tolist()
df = pd.DataFrame({'exSkillCountResumeMatch': expCount,
'impSkillCountResumeMatch': impCount}, index=index)

ax = df.plot.bar(rot=0)
ax.set_xlabel('Job ID')

ax.set_ylabel('Implicit_and_Explicit_Resume_match_with_Implicit')

# Barplot for dynamic approach to show how the implicit skills
weightage influence ranking of the job list.
df2 = df

```

```

if (dynStat == 1):
    index = []
    df = pd.DataFrame()
    indexNo = 0
    for ind, skillList in finalSkillWeightList:
        if not skillList:
            continue
        index.append(ind)
        for skill, weight in skillList:
            df.loc[indexNo, [skill]] = weight
        indexNo = indexNo + 1
    # print
    df.index = index
    df = df.reindex(index=df2.index)

    ax = df.plot.bar(rot=0)
    ax.set_xlabel('Job ID')

ax.set_ylabel('Implicit_and_Explicit_Resume_match_with_Implicit')

plt.show()
plt.clf()

```

## **JOB RECOMMENDED.PY:**

```

import config
import glob
import numpy as np
import matplotlib.pyplot as plt
from FunctionsForJobRecommendation import
FunctionsForJobRecommendation
import os
import json

def main():
    # The data scraped from web is obtained from reference dataset

```

which is stored in JSON file

```
exists = os.path.isfile(config.JOBS_INFO_JSON_FILE)
if exists:
    with open(config.JOBS_INFO_JSON_FILE, 'r') as fp:
        JobsInfo = json.load(fp)

# Initialize skill_keyword_match with JobsInfo
skill_match = FunctionsForJobRecommendation(JobsInfo)

# Extract skill keywords from job descriptions
skill_match.ExtractJobDescKeywords()

# Extract resume skills from given resume and store them in a list
for resumePDF in
glob.glob(config.SAMPLE_RESUME_PDF_DIR+"SampleResume*.pdf"):

print("=====
==")
    print("Processing the resume : ",resumePDF)

print("=====
==")
    ResumeSkills = skill_match.ExtractResumeKeywords(resumePDF)
    ResumeSkills.reset_index(inplace=True)
    ResumeSkills.rename(columns={'index': 'skillsinresume'},
inplace=True)
    ResumeSkillList = ResumeSkills['skillsinresume'].tolist()
    resume_skill_list_dummy =
['azure','sql','mysql','c++','excel','power','keras','agile','r','tableau','googl
e']

    print("Skills extracted from resume are : \n",ResumeSkillList)

# Calculate similarity of skills from a resume and job post and get
top10 job descriptions
MainTop10JDs = skill_match.CalculateSimilarity(ResumeSkillList)
# copy of the dataframe as "MainTop10JDs2" to keep them different
```

```

for static and dynamic approach
    MainTop10JDs2 = MainTop10JDs.copy()

    # Extract 20 similar Job description for each of the top10 job
descriptions
    # Explicit and Implicit skills extracted for static weight approach
    ImplicitStatic,finalSkillWeightList =
skill_match.Extract20SimilarJDs(0,MainTop10JDs, ResumeSkillList)

    # Calculating Final cosine score based on term frequency and
weighted cosine similarity
    FinalJDPrev =
skill_match.WeightedCosineSimilarity(ResumeSkillList, ImplicitStatic)
    print("Below is the reference approach job listing
ranking\n",FinalJDPrev[['Jobid','final_cosine']])

    # Extract 20 similar Job description for each of the top10 job
descriptions
    # Explicit and Implicit skills extracted for dynamic weight approach
    ImplicitDynamic,finalSkillWeightList =
skill_match.Extract20SimilarJDs(1,MainTop10JDs2, ResumeSkillList)
    # Calculating Final cosine score based on term frequency and
weighted cosine similarity

    FinalJD = skill_match.WeightedCosineSimilarity(ResumeSkillList,
ImplicitDynamic)
    print("Below is the proposed approach job listing
ranking\n",FinalJD[['Jobid','final_cosine']])
    topIndex = FinalJD['Jobid'][0]
    allTopSkills = ImplicitDynamic.loc[topIndex]['keywords']

    topExSkills = []
    topImpSkills = []
    for skill, weight in allTopSkills:
        if weight ==1:
            topExSkills.append(skill)
        else:

```



```

        topImpSkills.append(skill)
    print("Explicit skills to upskill :
",np.setdiff1d(topExSkills,ResumeSkillList))
    diffImpSkills = np.setdiff1d(topImpSkills,ResumeSkillList)
    if len(diffImpSkills)>5:
        print("Implicit skills to upskill :
",np.setdiff1d(topImpSkills,ResumeSkillList)[0:5])
    else:
        print("Implicit skills to upskill :
",np.setdiff1d(topImpSkills,ResumeSkillList))

    # Graph plot with explicit and implicit skills that match the resume
    for static approach
        ImplicitStaticGraph =
        FinalJDPrev[["Jobid","final_cosine","exSkillCountResumeMatch","impSkillCountResumeMatch"]]
        #
        skill_match.GraphPlotsForEvaluation(ImplicitStaticGraph,finalSkillWeightList,0)

    # Graph plot with explicit and implicit skills that match the resume
    for dynamic approach
        # Graph plot to show how the ranking of the top10 job postings
        differ due to the Implicit weightage of skills
        ImplicitDynamicGraph =
        FinalJD[["Jobid","final_cosine","exSkillCountResumeMatch","impSkillCountResumeMatch"]]
        #
        skill_match.GraphPlotsForEvaluation(ImplicitDynamicGraph,finalSkillWeightList,1)
        if(resumePDF.count(r'SampleResume1') == 1):
            plt.figure()

    skill_match.AllGraphPlotsForEvaluation(ImplicitStaticGraph,ImplicitDynamicGraph,finalSkillWeightList,1)

if __name__ == "__main__":

```

main()

## 8.TESTING

### 8.1 TEST CASES:

TestcaseID	FeatureType	Component	TestScenario
LoginPage_TC_O O1	Functional	HomePage	Verifyuser is able to see theLogin/Signup popup when userclickedonMy accountbutton
LoginPage_TC_O O2	UI	HomePage	Verify the UI elements inLogin/Signupp up
LoginPage_TC_O O3	Functional	Home page	Verify user is able to log intoapplicationwithV alidcredenti als
LoginPage_TC_O O4	Functional	Loginpage	Verify user is able to log intoapplicationwithI nValidcredenti also
LoginPage_TC_O O5	Functional	Loginpage	Verify user is able to log intoapplicationwithI nValidcredenti als

Pre-Requisite	StepsToExecute	TestData
---------------	----------------	----------

	1.EnterURLandclickgo 2.Click on My Account dropdownbutton 3.Verifylogin/Singuppopupd isplayed ornot	index.html
	1.EnterURLandclickgo 2.Click on My Account dropdownbutton 3.Verify login/Singup popup withbelow UI elements: a.email textbox b.password text boxc.Loginbutton d.New customer? Create account linke.Last password? Recovery p	index.html
	1.Enter URL(index.html) and click go 2.Click on My Account dropdownbutton 3.Enter Valid username/email in Emailtextbox 4.Entervalidpasswordinpas swordte xtbox 5.Clickonloginbutton	Username: viswanthkumar9999@gmai l. com password:Viswanth@2328
	1.Enter URL(index.html) and click go 2.Click on My Account dropdownbutton 3.Enter InValid username/email inEmailtext box 4.Entervalidpasswordinpas swordte xtbox 5.C	Username: viswanthkumar9999@gmai l. com password:Viswanth@2328
	1.Enter URL(index.html) and click go2.Click on My Account dropdownbutton 3.Enter Valid username/email in	Username: viswanthkumar9999@gmai l. com password:Viswanth@2328

	Emailtextbox 4.Enter Invalid password in passwordtextbox 5.Clickonloginbutton	
	1.Enter URL(index.html) and click go2.Click on My Account dropdownbutton 3.Enter InValid username/email inEmailtext box 4.Enter Invalid password in passwordtextbox 5.Clickonloginbutton	Username: viswanthkumar9999@gmail. com password:Viswanth@2328

ExpectedResult	ActualResult	Status	Commnets
Login/Signuppopup should display	Working asexpected	pass	
Application should show below Uelements: a.email text boxb.passwordtext box c.Login button with orange colourd.New customer? Create account linke.Last passw	Working asexpected	pass	
User should navigate to user accounthomepage	Working asexpected	pass	
Application should show 'Incorrectemail or password validationmessage.	Working asexpected	pass	
Application should show 'Incorrectemail or password validationmessage.	Working asexpected	pass	
Application should show 'Incorrectemail or password validationmessage.	Working asexpected	pass	

## **8.2.USER ACCEPTANCE TESTING:**

### **PurposeofDocument:**

Thepurposeofthisdocumentistobrieflyexplainthetestcoverageandopenissue  
softhe[ProductName]projectatthetimeofthereleasetoUserAcceptanceTesti  
ng (UAT).

### **DefectAnalysis:**

Thisreportshowsthenumberofresolvedor closedbugsat eachseverity  
level,andhowthey wereresolved.

Resolution	Severity1	Severity2	Severity3	Severity4	Subtotal
ByDesign	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
NotReprodu ced	0	0	1	0	1
Skipped	0	0	1	1	2
Won'tFix	0	5	2	1	8
Totals	24	14	13	26	77

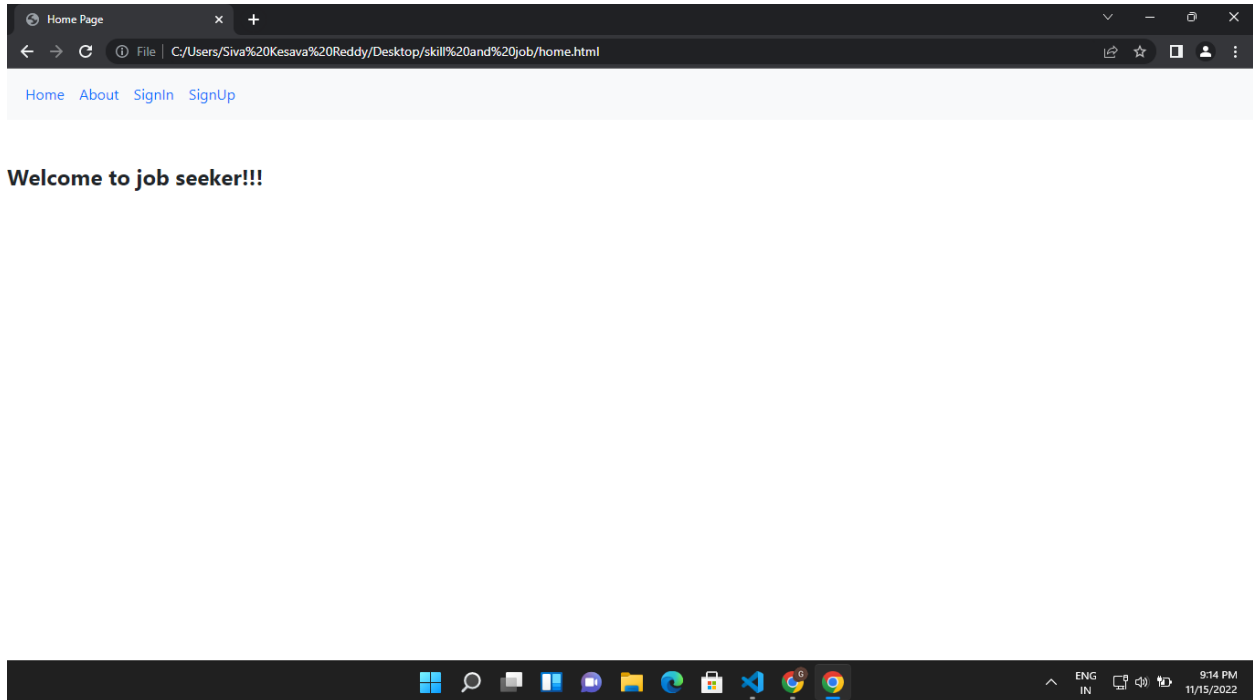
## TestCaseAnalysis:

This report shows the number of test cases that have passed, failed, and untested

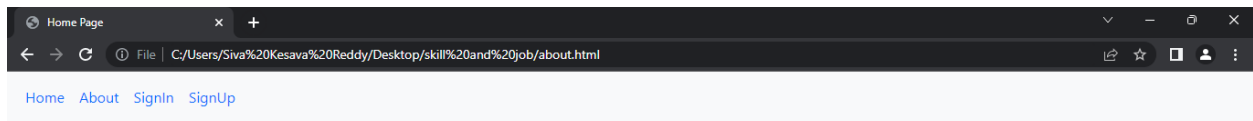
Section	TotalCases	NotTested	Fail	Pass
PrintEngine	7	0	0	7
ClientApplication	51	0	0	51
Security	2	0	0	2
OutsourceShipping	3	0	0	3
ExceptionReporting	9	0	0	9
FinalReportOutput	4	0	0	4
VersionControl	2	0	0	2

## 9.RESULTS

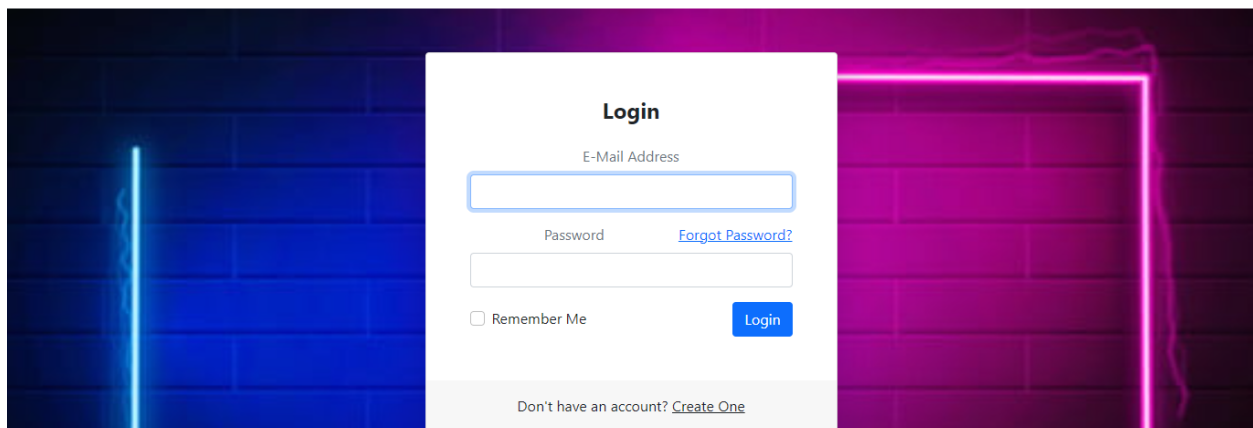
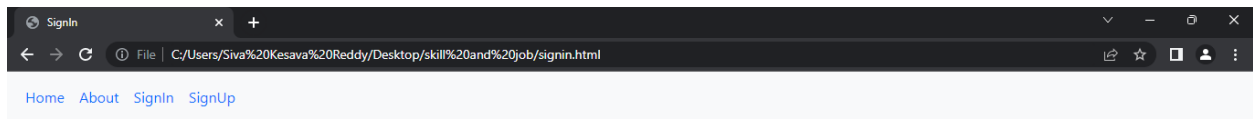
### 9.1.PERFORMANCE METRICS:

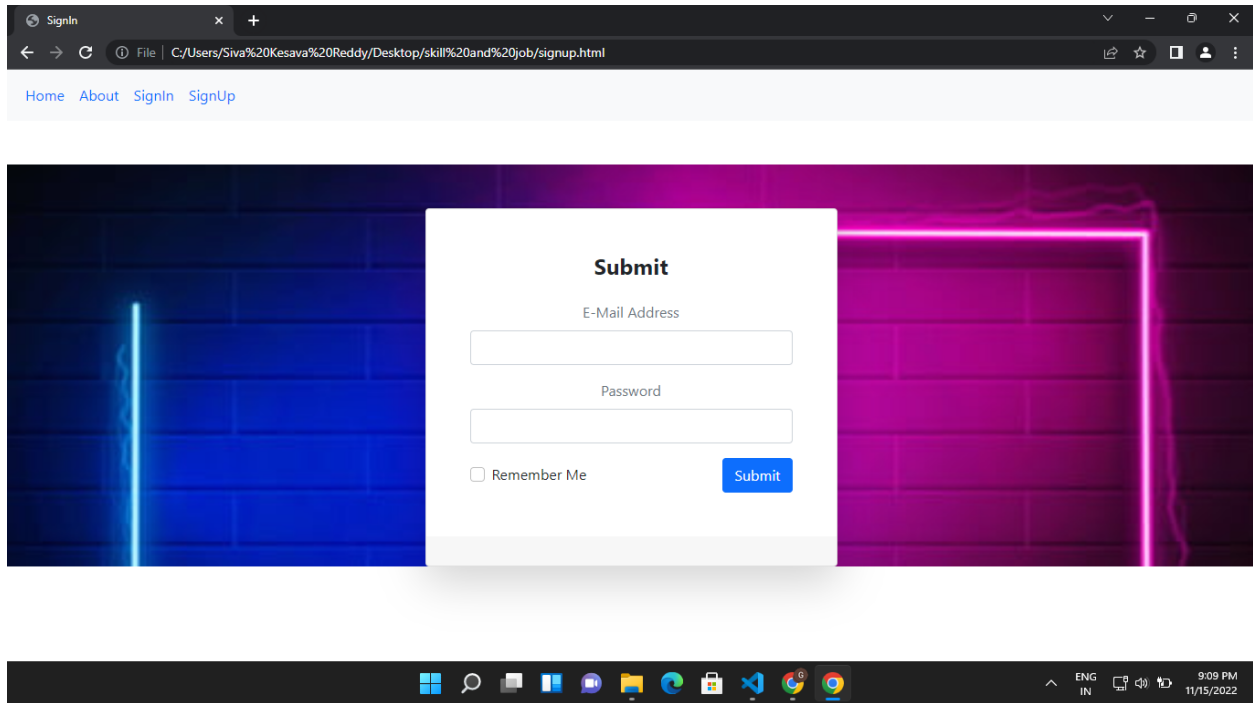






**Welcome to Job Seeker !! here you can find the jobs that you need and fit for your resume and your skills !!THE MORE SKILLS THE MORE RECOMMENDATIONS!!**





## 10.CONCLUSION

Job Recommendation System has a major role to play among recommending systems. With the presence of new algorithms and techniques, the system needs to evolve along with it. The main objective of this project is to recommend a suitable job for the candidates. This project has two pre-processing methods, one text mining method and one similarity function. The pre-processing methods are stop words and porter stemmer. The text mining method is tf-idf. The similarity function is a cosine similarity function. Pre-processing methods are used with resumes and with jobs description, to make the system more efficient by avoiding some garbage words. Tf-idf is used in processed resumes and processed jobs descriptions to convert it from text to matrix to compare. Cosine Similarity will measure the similarity between the resume and each job description.

Finally, it will display the scores for the jobs in a sorted way. There is also a pie chart which is used to visualize the percentage of the scores which is got by the candidate for the jobs. Then use a list compare

method to compare the resume and job skills to recommend the skills to be improved by the candidate.

## **11.REFERENCES**

- [1] R. J. Mooney and L. Roy, "Content-Based Book Recommending Using Learning for Text Categorization," in In Proceedings of DL '00: Proceedings of the Fifth ACM Conference on Digital Libraries, New York, NY, pp. 13-20, 2000.
- [2] Li-Ping Jing, Hou-Kuan Huang, Hong-Bo Shi, "Improved feature selection approach TFIDF in text mining", International Conference on Machine Learning and Cybernetics, pp. 944-946, 2002, doi:10.1109/icmlc.2002.1174522.
- [3] Shouning Qu ,Sujuan Wang,Yan Zou, " Improvement of Text Feature Selection Method Based on TFIDF", International Seminar on Future Information Technology and Management Engineering, pp. 79-81, 2008, doi:10.1109/fitme.2008.25.
- [4] I. A. Braga, "Evaluation of stopwords removal on the statistical approach for automatic term extraction," Seventh Brazilian Symposium in Information and Human Language Technology, pp. 142-149, 2009.
- [5] Nikolaos D. Almalis, Prof. George A. Tsihrintzis, Nikolaos Karagiannis, Aggeliki D. based job recommendation algorithm for job seeking and recruiting", 6th International Conference on Information, Intelligence, Systems and Applications (IISA), pp. 1-7, 2015, doi:10.1109/iisa.2015.7388018.
- [6] Mohammad Alodadi and Vandana P. Janeja, " Similarity in Patient Support Forums Using TF-IDF and Cosine Similarity Metrics", International Conference on Healthcare Informatics, pp. 521-522, 2015, doi:10.1109/ichi.2015.99.
- [7] L. Zahrotun, "Comparison jaccard similarity, cosine similarity and combined both of the data clustering with shared nearest neighbor method," Computer Engineering and Applications Journal. vol. 5. Pp. 11-18, 2016, doi:10.18495/comengapp.v5i1.160, 2016. [8] Peng Yi, Cheng Yang ,Chen Li, Yingya Zhang, "A Job Recommendation Method Optimized by Position Descriptions and Resume Information", IEEE Advanced Information Management, Communicates, Electronic and Automation

Control Conference (IMCEC), pp. 762 -764, March 2017, doi:10.1109/rteict.2017.8256590.

[9] Minh-Luan Tran, Anh-Tuyen Nguyen, Quoc-Dung Nguyen, Tin Huynh, "A comparison study for job recommendation", International Conference on Information and Communications (ICIC), pp. 199-204, 2017, doi:10.1109/infoc.2017.8001667.

[10] Gokul P.P, Akhil BK, Shiva Kumar K.M, "Sentence similarity detection in Malayalam language using cosine similarity", 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), pp. 221-225, 2017, doi:10.1109/rteict.2017.8256590.

[11] Leah G. Rodriguez, Enrico P. Chavez, "Feature Selection for Job Matching Application using Profile Matching Model", IEEE 4th International Conference on Computer and Communication Systems (ICCCS), pp. 2-4 , FebStrati.

[12] Nunik Destria Arianti, Mohamad Irfan, Undang Syaripudin, Dina Mariana, Neny Rosmawarni, Dian Sa'adillah Maylawati, "Porter Stemmer and Cosine Similarity for Automated Essay Assessment", 5th International Conference on Computing Engineering and Design (ICCED)", pp. 1-5, 2019, doi:10.1109/icced46541.2019.91610.

[13] Garima Koushik, Dr. Prof. K. Rajeswari, Mr. Suresh Kannan Muthusamy, "Automated Hate Speech Detection on Twitter. 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)", pp. 421- 425, 2019, doi:10.1109/iccubea47591.2019.912.

[14] Ravali Boorugu, Dr. G. Ramesh, "A Survey on NLP based Text Summarization for Summarizing Product Reviews", Second International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 352- 356, 2020, doi: 10.1109/ICIRCA48905.2020.9183355.

[15] Tanya V. Yadalam,Vaishnavi, M.Gowda, Vanditha Shiva Kumar, Disha Girish, Namratha M, "Career Recommendation System Using Content Based Filtering", International Conference on Communication and Electronics Systems (ICCES), pp. 2-5, June 2020, doi: 10.1109/ICCES48766.2020.9137992

[16] Jeevamol Joy and Renumol V G, "Comparison of Generic Similarity Measures in E-learning Content Recommender System in Cold-Start Condition", IEEE Bombay Section Signature Conference (IBSSC)", pp.

175- 179, 2020, doi:10.1109/ibssc51096.2020.9332162.

[17] M. Alamelu, D.Sathish Kumar, R. Sanjana, J.Subha Sree, A.Sangeerani Devi, D. Kavitha, "Resume Validation and Filtration using Natural Language Processing", 10th International Conference on Internet of Everything, Microwave Engineering, Communication and Networks (IEMECON), pp. 412-430, 2021, doi:10.1109/IEMECON53809.2021.9689075.

[18] Swaranjali Jugran, Ashish Kumar, Bhupendra Singh Tyagi, Mr. Vivek Anand,"Extractive Automatic Text Summarization using SpaCy in Python & NLP", International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), pp. 582-585, 2021, doi:10.1109/icacite51222.2021.9404712.

[19] S. Prathyusha, S. Jadhav, K. Kommu, M.S. Velpuru, "Text summarization using NLTK with GUI interface", 4th Smart Cities Symposium (SCS 2021), pp. 435-442, 2021, doi: 10.1049/icp.2022.0369.

[20] Meenakshi A. Thalor, "A Descriptive Answer Evaluation System Using Cosine Similarity Technique", International Conference.

GITHUB LINK : <https://github.com/IBM-EPBL/IBM-Project-29091-1660120734>

VIDEO LINK : [https://drive.google.com/drive/folders/19kO\\_uJdQta551zDQMAg8p7rA\\_Y50Y8UU?usp=sharing](https://drive.google.com/drive/folders/19kO_uJdQta551zDQMAg8p7rA_Y50Y8UU?usp=sharing)