

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

- 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

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

Source Code

GitHub & Project Demo Link

1.INTRODUCTION

The Internet-based recruiting platforms become a primary recruitment channel in most companies. While such platforms decrease the recruitment time and advertisement cost, they suffer from an inappropriateness of traditional information retrieval techniques like the Boolean search methods. Consequently, a vast number of candidates missed the opportunity of recruiting. 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. In order to improve the e-recruiting functionality, many recommenders' system approaches have been proposed. This article will present a survey of e-recruiting process and existing recommendation approaches for building personalized recommender systems for candidates/job matching. Recruiting process is a core function of human resource management treating the labor as one of the important factors of production. Objective of the recruiting process is to hire candidates who are valuable for the company. Two viewpoints are distinguished: from recruiters' and job seekers. The recruiters generate the job description by determining the set of requirements and constraints on skills, expertise levels, and degrees. The job-seeker, on the otherhand, generates his/her CV by specifying the academic background, previous work experience and skills. The IT support for the recruiting activities is ranging from finding talent to choose and retain candidates. The degree of process integration represents the complexity of using e-recruitment solutions. The e-recruitment is a system for quickly reaching a large set of potential job-seekers. Recruiting has attractive growth since the late 1990s when the rapid economy changes produced a high demand for qualified candidates that the labor market could not fully satisfy. The e-recruiting platforms such as corporate homepages and job portals have driven this development.

1.1 PROJECT OVERVIEW:

The Project is one which helps out the HR Personal in the recruitment of new candidates to the company and helps the recruitment process as a whole. A college system consists of a student login, company login and an admin login. This project is beneficial for college students, various companies visiting the campus for recruitment and even the college placement officer. The said software system allows the students to create their profiles and upload all their details including their marks onto the system. The admin can check each student details and can remove faulty accounts. This system also consists of a company login where various companies visiting the college can view a list of students in that college and also their respective resumes. This software system allows students to view a list of companies who have posted for vacancy. The admin has overall rights over the system and can moderate and delete any details not pertaining to college placement rules. This system handles student as well as company data and efficiently displays all this data to respective sides. This system also contains discussion page where many people could discuss or chat on multiple topics.

1.2 PURPOSE:

Improves accuracy in result. It has user-friendly interface having quick authenticated access to documents. It provides the facility of maintaining the details of the students. It will reduce the paper work and utilize the maximum capabilities of the setup and organization as well as it will save time and money, which are spending in making reports and collecting data. It can be access throughout the organization and outside as well with proper login provided. This system can be used as an application for college to manage the student information concerning placement. Helps company coming for campus recruitment to see student details. Before coming for campus, company can get information about eligible students along with interested students. This project can be used very easily in the process of decision making in new recruitments. Elective way of providing communication between job providers and job seekers. Reliable and consistent way of searching jobs. Conducting secured and restricted online exam for screened employees.

2.LITERATURE SURVEY

2.1 Existing Problem:

Students are not getting any proper information regarding interview process and fake job postings. The current problem is recruitment is done manually, most available jobs can only be applied at the agency and can be done for which job seekers have to go to the agency to check the available jobs at the agency. 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. CRS processes were used to be manual. The administrator has to refer all the records kept for years ago to simply know the details. This is very tedious. There are many limitations for the existing systems. In manual campus placement, all the work done manually so there were many chances of getting errors. The interface of student and admin is maximum which makes the system time consuming. Students created and submitted their CVs early in the year, leaving them frozen in time. Lists were produced for each company, and students had to regularly travel in to review the notice board. The process is too slow and consumes valuable academic time that was diverted from activity which is more useful. The College records are stored and modified in excel sheets hence sorting is a problem. Searching was used to be done manually based on the company criteria. Training and Placement Officers (TPO) will identify the eligible student by altering the excel sheet. TPO has to see each and every student marks and their eligibility. No searching method is provided. The student will get notified through traditional notice board only. There may be chances of loss of opportunity. The students were not being made aware of the TPO activity. For communication, one has to go to

TPO personally to obtain relevant information such as company question paper ,job details ,which is not available with the student.

2.2 References:

1. A survey of Job Recommender Systems : ISSN 1992-1950
2. Career Recommendation System Design by adopting Machine Learning Techniques: ISSN 2393-9028
3. Job Recommendation System based on skillsets : ISSN 2320-2882
4. Job Seekers Acceptance of Job Recommender System : Results of an Empirical Study ISSN 978-09981331-1-9

2.3 PROBLEM STATEMENT DEFINITION:

Problem:

“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?”

In current situation recruitment s done manually for lakhs of students in which many talented students may lose their opportunities due to different reasons since it is done manually, and company also need the highly talented people from the mass group for their growth. So we have build a cloud application to do this process in a efficient manner.

Solution:

“The purpose of job oriented application is to help both the job seekers and recruiters find the right organization or the employers.”

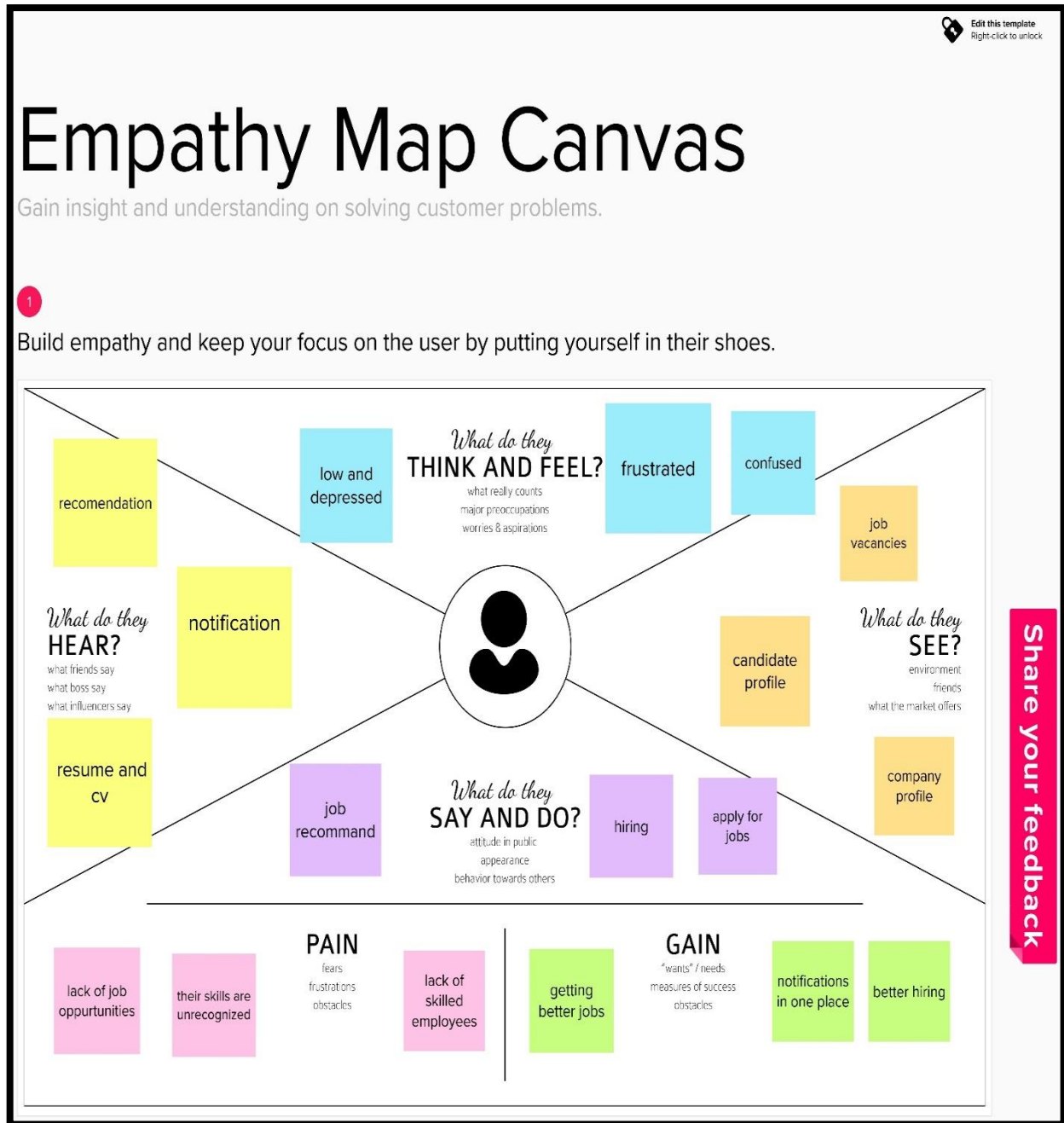
Goal:

Searching for a job is very intuitive for students. This application helps them to find a job based on their skills , position , industry , role and location of the company.

- The job Skills recommended application is an example of a search where documents are bulky because of the content in candidate resumes.
- The search provided over the candidate database is required to have a huge set of fields to search.

3. IDEATION & PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS:



3.2 IDEATION AND BRAINSTORMING :

Brainstorm & idea prioritization

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

10 minutes to prepare
 1 to 10 participants
 10 people recommended

Before you ideate

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

10 minutes

- Get priority: Write the issue statement in the space and add it to the board when the session starts.
- Get people: This session is best when you have a mix of people from across the organization.
- Get the room: This session is best when you have a mix of people from across the organization.

Define your problem statement

To develop an end-to-end solution capable of solving the problem, you need to define the problem.

10 minutes

Issue: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

Brainstorm

To create an application regarding the clarity of ideas to be presented by the team, you need to define the problem.

10 minutes

Key result: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

Create ideas

Take time during your idea session to generate ideas in related areas as you go. In the end, 10 minutes, you will have a list of ideas to work on. To create a list of ideas, you need to define the problem.

10 minutes

Key result: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

Prioritize

Your team should all be on the same page about what's important to solving the problem. Place your ideas in the grid to determine which ideas are important and which are feasible.

10 minutes

Key result: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

After you ideate

You can expect the team to have a list of ideas to work on. To create a list of ideas, you need to define the problem.

10 minutes

Key result: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

Key result: To create a solution to solve the problem of...

3.3 PROPOSED SOLUTION:

The proposed solution templates for skill and job recommender system is as follows:

S.No	Parameter	Description
1	Problem Statement	Students are not getting any proper information regarding interview process and fake job postings.
2	Idea/Solution description	<ol style="list-style-type: none">1) To devise recommender model which recommends Job to the job seeker based on skills.2) 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.
3	Novelty/Uniqueness	<ol style="list-style-type: none">1) We are using the CHATBOT. Users will interact with the chatbot and can get the recommendations based on their skills.2) Maintaining database which contains details of the job availability.
4	Social Impact	<ol style="list-style-type: none">1) It helps us to get job easily.2) Very helpful in making decision faster.3) Reduce Unemployment people.
5	Business Model	<ol style="list-style-type: none">1) This can be implemented in all region and also provide accurate job recommendations for the students.2) Accurate recommendation can encourage the users.
6	Scalability of Solution	<ol style="list-style-type: none">1) High performance.2) Accurate recommendation.3) Availability.

3.4 PROBLEM SOLUTION FIT:

Project Title: Skill/Job Recommender Application

Project Design Phase-I - Solution Fit Template

Team ID:PNT2022TMID39089

1. CUSTOMER SEGMENT(S) CS		6. CUSTOMER CONSTRAINTS CC		5. AVAILABLE SOLUTIONS AS	
<ul style="list-style-type: none"> Freshers who are looking for entry level positions Experienced but unemployed people looking for a job Those looking to change career tracks 		<ul style="list-style-type: none"> Lack of awareness about a job opening Network connectivity issues that might hinder their access to the website Difficulty in distinguishing between real and fake job openings Security of personal data that they are providing 		<ul style="list-style-type: none"> Prior to digitalization, TV advertisements and newspaper columns were used Currently, websites like LinkedIn, GlassDoor, Indeed, etc enable job seekers to find openings 	
Focus on J&P, tap into BE, understand RC	2. JOBS-TO-BE-DONE / PROBLEMS J&P	9. PROBLEM ROOT CAUSE RC		7. BEHAVIOUR BE	Focus on J&P, tap into BE, understand RC
	<ul style="list-style-type: none"> User provided with real time updates regarding openings at relevant companies Chatbot assistance for easier navigation Awareness about fraudulent job postings 	<ul style="list-style-type: none"> Increased population and hence increased competition for limited job openings Improper curriculum in universities in order to properly assist freshers in the job market Lack of awareness of job openings by deserving candidates 		<ul style="list-style-type: none"> Develop and improve industry required skills Search for jobs based on requirements Network with recruiters for possible openings 	
Identify strong TR & EM	3. TRIGGERS TR	10. YOUR SOLUTION SL		8. CHANNELS of BEHAVIOUR CH	Identify strong TR & EM
	<ul style="list-style-type: none"> Financial Problems Societal Pressure Job dissatisfaction 	<ul style="list-style-type: none"> Filter jobs according to requirements Providing resources for hot-skills in the industry Alerts sent to users regarding job openings Chatbot assistance for easier navigation of the application Prevent fake job postings in the application 		8.1 ONLINE <ul style="list-style-type: none"> Search for job openings Apply and keep track of applications 8.2 OFFLINE <ul style="list-style-type: none"> Improve relevant skills by means of learning platform resources provided in the application Prepare for and attend in-person interviews 	

4. EMOTIONS: BEFORE / AFTER EM		
BEFORE <ul style="list-style-type: none"> Low self esteem Pressurized from society Fear of rejection AFTER <ul style="list-style-type: none"> Connected to the society Positivity in life Reverence of smartness 		

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Sign in / Login	Registering with username and password
FR-2	Profile Registration	Registering with username, password, email, qualification, skills. These data will be stored in a database.
FR-3	Job profile display	This will display job profiles based on availability, location, skills.
FR-4	Chatbot	A chat on the webpage to solve user queries and issues.
FR-5	Job Registration	The company's registration/Description details will be sent to the registered email id of the user.
FR-6	Logout	This option is for leaving the site after completing the job registration process.

4.2 NON –FUNCTIONAL REQUIREMENT:

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

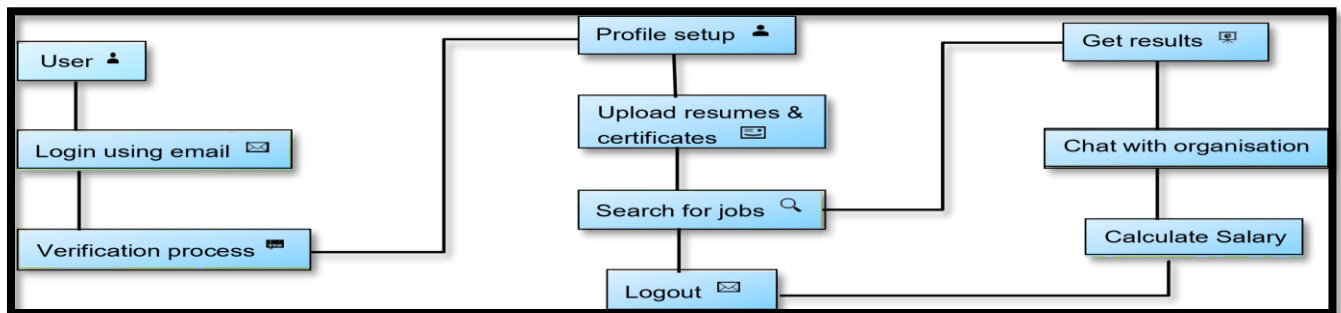
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	It is easy for any non-technical user, where they can easily navigate through it and complete the job registration work. (easy and simple design)
NFR-2	Security	Python flask is used for cloud connection and that will provide security to the project. Database will be safely stored in DB2.
NFR-3	Reliability	To make sure the webpage doesn't go down due to network traffic.

NFR-4	Performance	Focus on loading the webpage as quickly as possible irrespective of the number of user/integrator traffic.
NFR-5	Availability	The webpage will be available to all users (network connectivity is necessary) at any given point of time.
NFR-6	Scalability	Increasing the number of users by increasing the storage space in the database. By enhancing some features in the future which will make a webpage more reliable and attractive.

5. PROJECT DESIGN

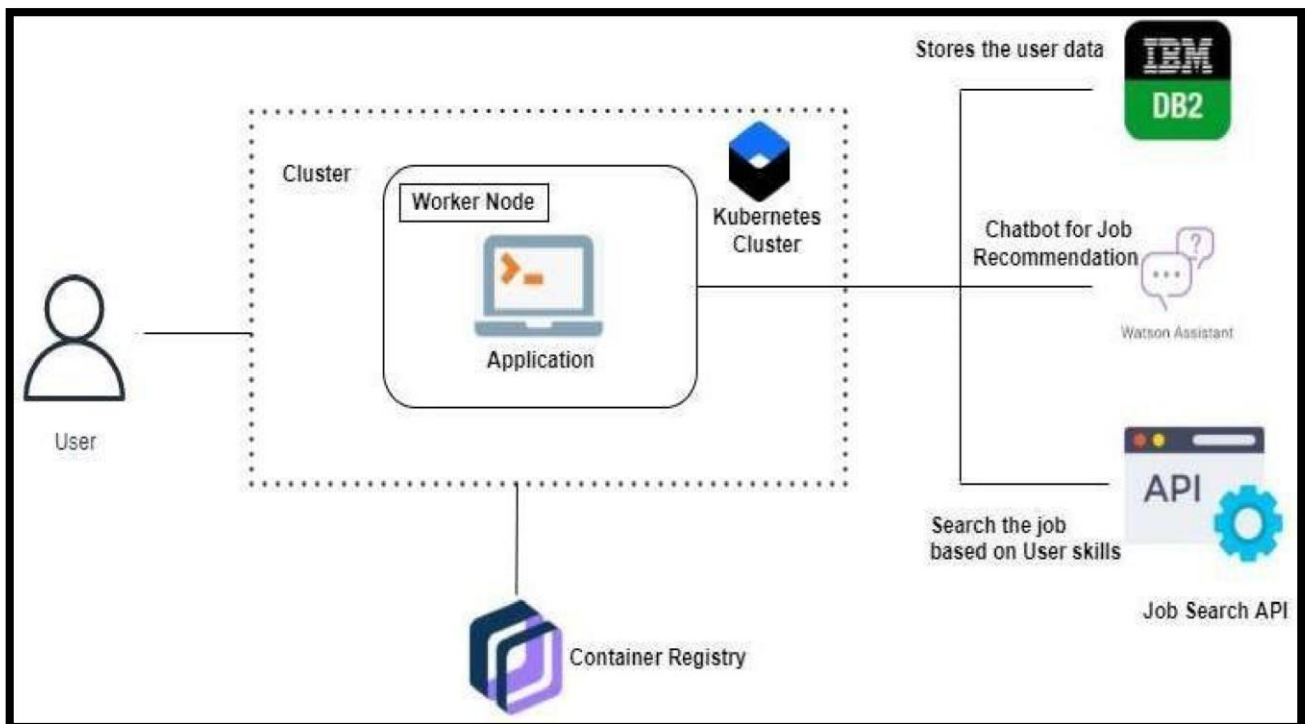
5.1 DATA FLOW DIAGRAM:

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



5.2 SOLUTION & TECHNICAL ARCHITECTURE:

SOLUTION ARCHITECTURE:



TECHNICAL ARCHITECTURE:

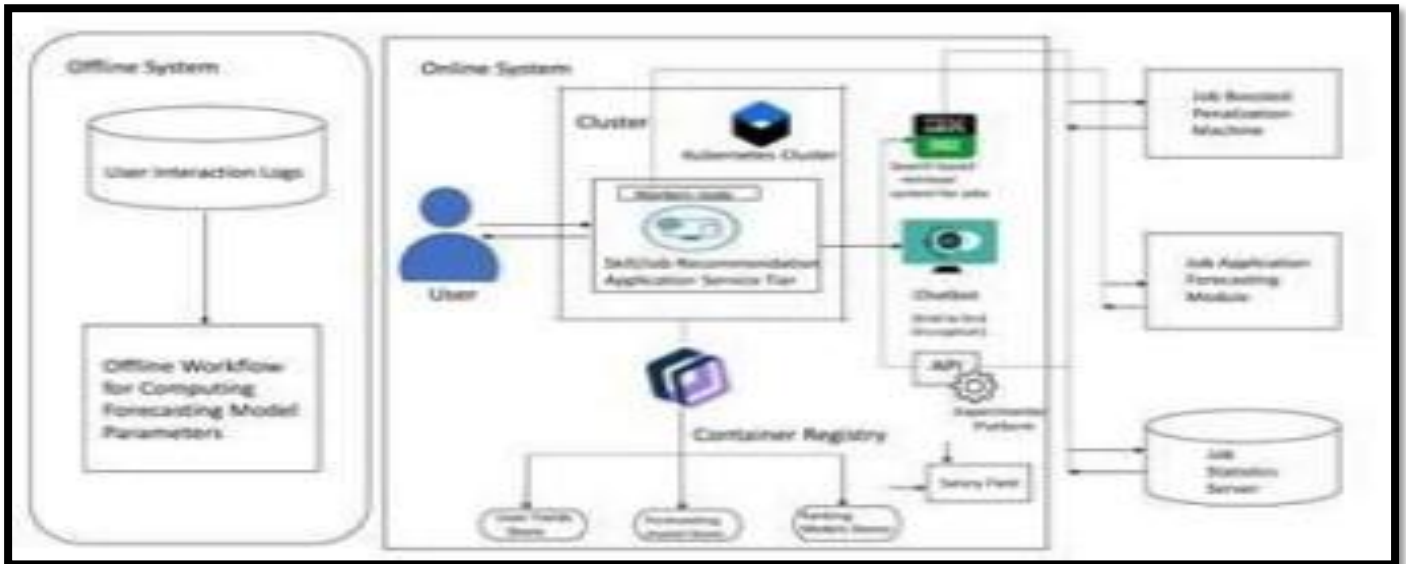


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The user interacts with software 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.	Chatbot Assistance	Conversational Interface	IBM Watson Assistant
4.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
5.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
6.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
7.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
8.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Open Source framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Microservices)	Artificial Intelligence (AI)
4.	Availability	Justify the availability of applications (e.g. use of load balancers, distributed servers etc.)	RAID(redundant array of independent disks)
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	DRAM or flash memory

5.3 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	The user can register for the application by entering email and password.	User can access account / dashboard	High	Sprint-1
		USN-2	The user will receive confirmation email on after registered for the application	The user can receive confirmation mail & click confirm	High	Sprint-1
		USN-3	The user can register for the application through Facebook/LinkedIn	User can register & access dashboard with Facebook or LinkedIn login	Low	Sprint-2
	Login	USN-4	The user can login to the application by entering email & password		High	Sprint-1
	Dashboard	USN-5	The user can access the dashboard after signing in.	I can access my account / dashboard	High	Sprint-1
Customer (Web user)	Access	USN-6	The user can setup a profile, and basic details by signing in.			
		USN-7	The user will upload my resume, certificates, and other requirements.	User can perform several task in the application	Medium	Sprint-1
Customer Care Executive	Chatbot	USN-8	The user can seek guidance from the customer care executive.		High	Sprint-1
Administrator	DBMS	USN-9	The administrator can keep the applications of your organization relies on running.	Admins can perform various modifications in the applications.	High	Sprint-1

6.PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	The user can register for the applicationby entering email and password	2	High	Gayathri k
Sprint-1		USN-2	The user will receive confirmation emailonce user have registered for the application	1	High	Logeshwari J
Sprint-1		USN-3	The user can register for the applicationthrough Facebook	2	Low	Mehaboob Nasreen S
Sprint-1	Login	USN-4	The user can login to application byentering email & password	1	High	Catherine Rozario
	Dashboard	USN-5	The user can access the website in a second.	2	High	Gayathri K
Sprint-1	Dashboard	USN-6	If user logged in correctly, user can view dashboard and can navigate to ay pages which are already listed there.	2	High	Logeshwari J
Sprint-2	User Profile	USN-7	The user can view and update the details	2	Medium	Mehaboob Nasreen S

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Database	USN-8	The user can store my details and data in the website.	2	Medium	Catherine Rozario
Sprint-2	Cloud Storage	USN-9	The user can upload photo, resume and much more in the website.	1	Medium	Gayathri K
Sprint-2	Chatbot	USN-10	The user can ask the Chatbot about latest job openings, which will help and show the recent job openings based on profile		High	Logeshwari J
Sprint-2	Identity-Aware	USN-11	The User can access the account by entering correct login credentials. User credentials are only displayed.	2	High	Mehaboob Nasreen S
Sprint-3	SendGrid service	USN-12	The user can get a notification or mail about a job opening with the help of send grid service.	1	Medium	Catherine Rozario
Sprint-3	Docker	USN-13	The user can access the website in any device	2	High	Gayathri K
Sprint-3	Kubernetes	USN-14	The user can access the website in any device	2	High	Logeshwari J
Sprint-3	Deployment in cloud	USN-15	The user can access the website in any device	2	High	Mehaboob Nasreen S
Sprint-3	Technical support	USN-16	The user can get a customer care support from the website which will solve all queries.	1	Medium	Catherine Rozario
Sprint-4	User Acceptance testing	USN-17	The user can access the website without any interruption	2	High	Gayathri K

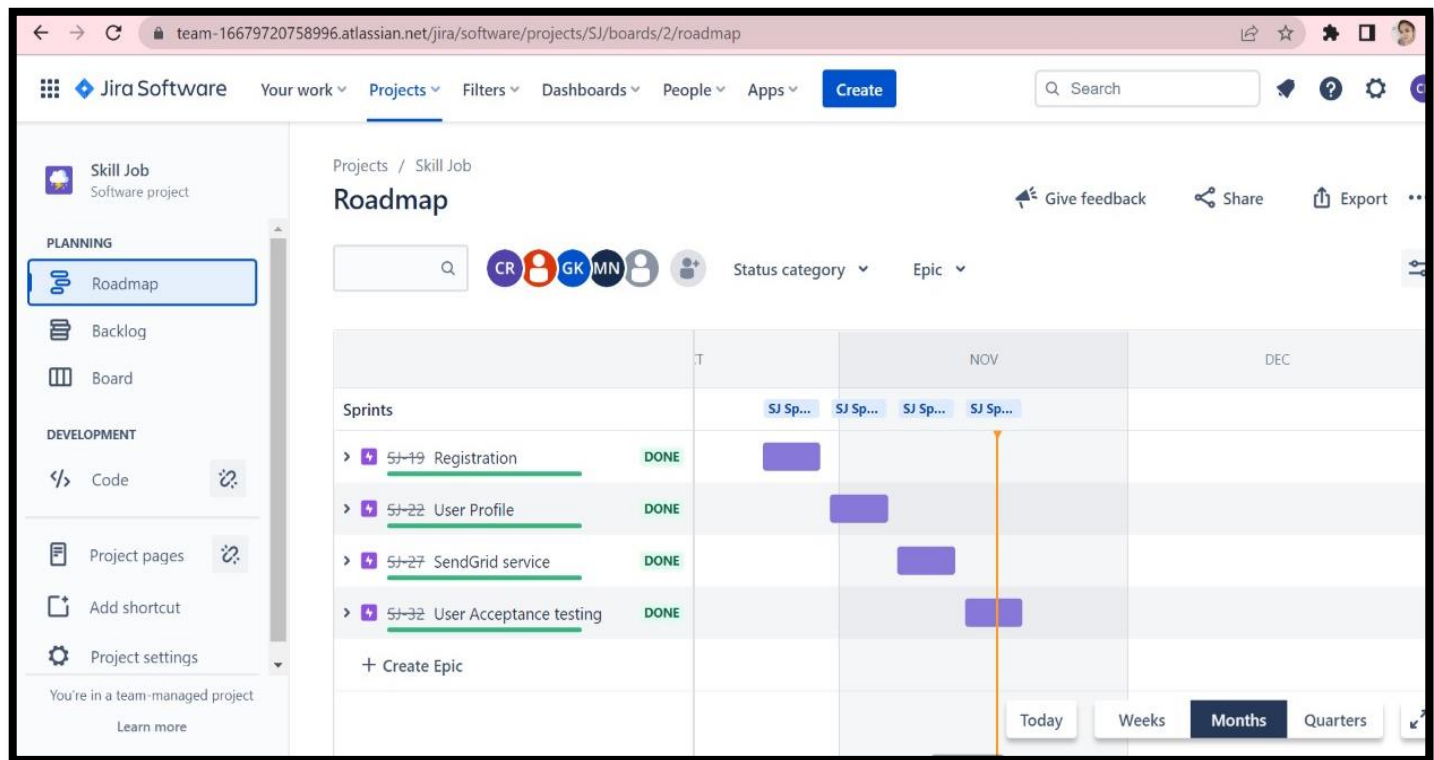
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Performance testing	USN-18	The user can access the website without any interruption	2	High	Logeshwari J

6.2 SPRINT DELIVERY SCHEDULE:

Total Story Points	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
20	6 Days	31 Oct 2022	05 Nov 2022	19	05 Nov 2022
20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA:

Roadmap:



Backlog:

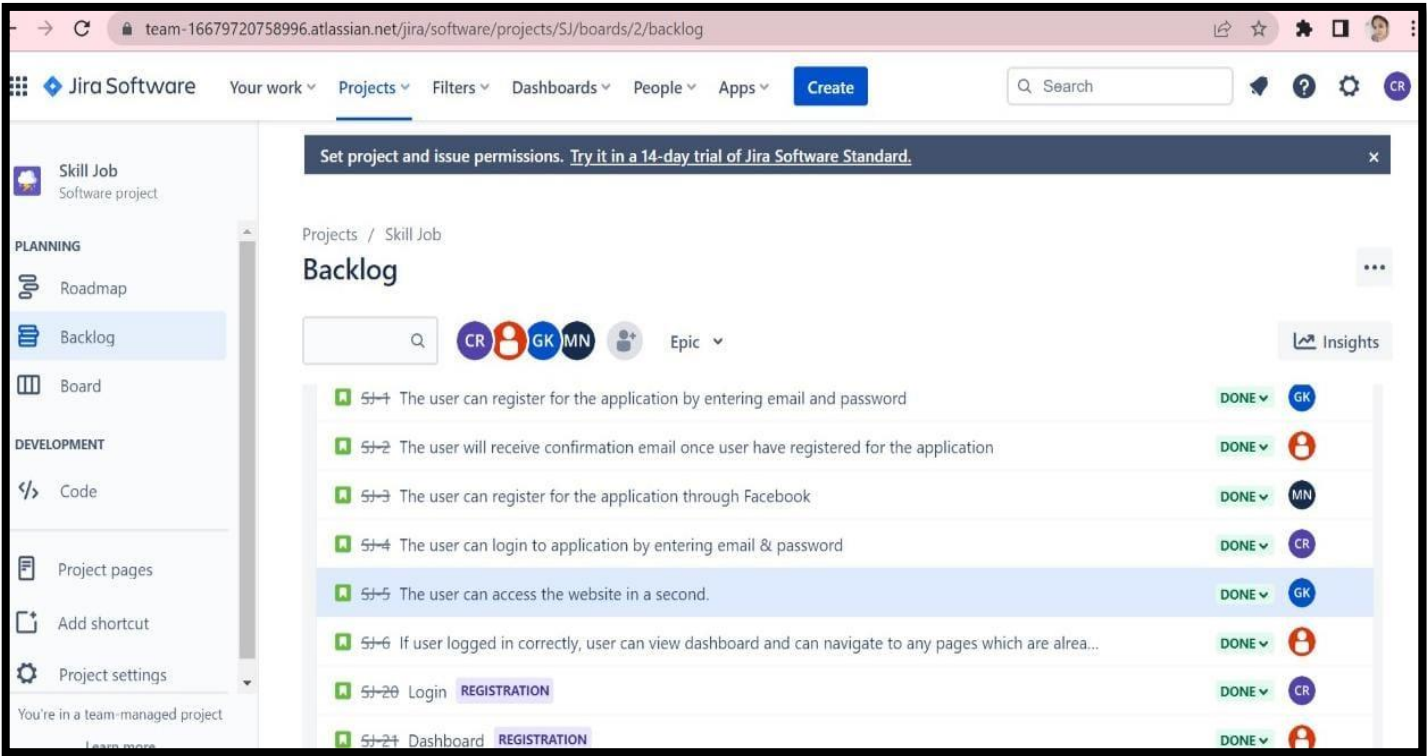


Fig: Sprint-1

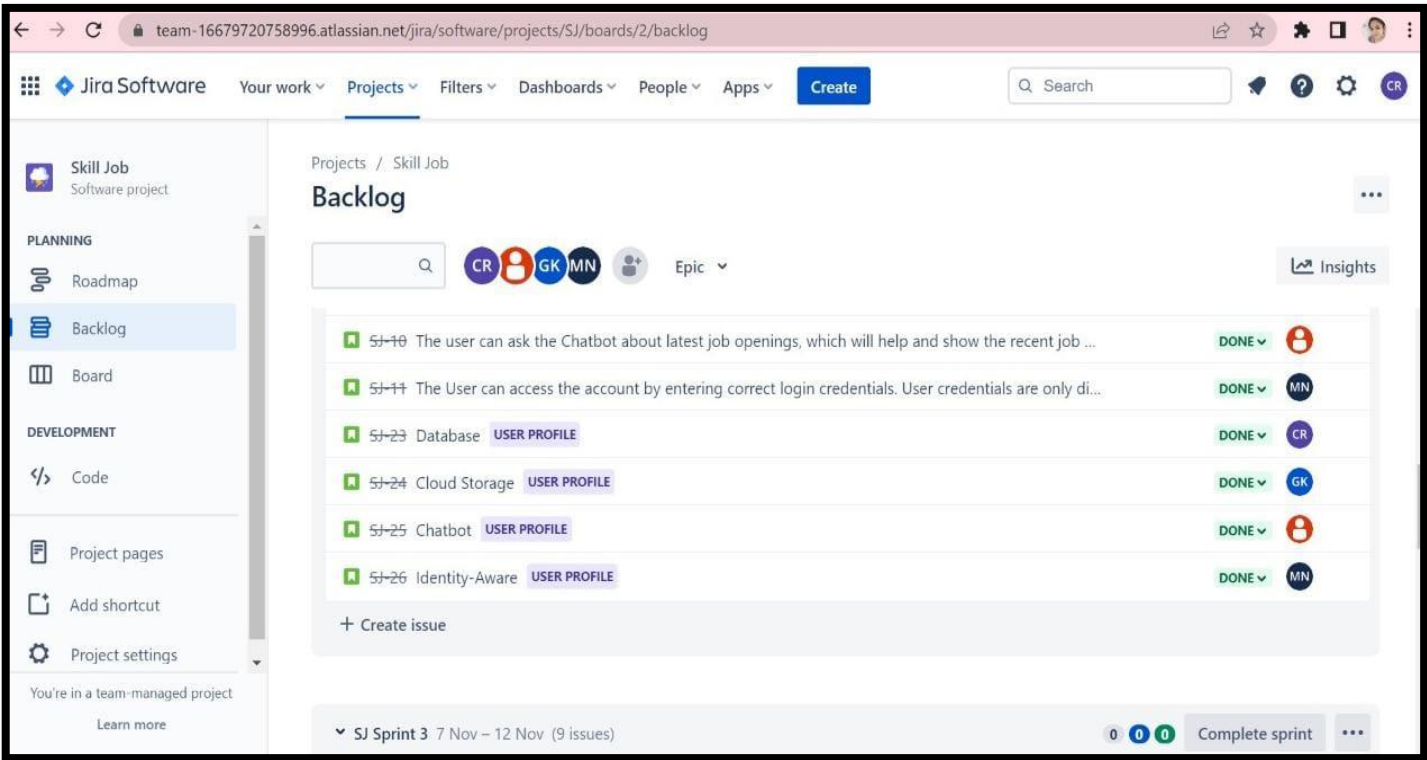


Fig:Sprint-2

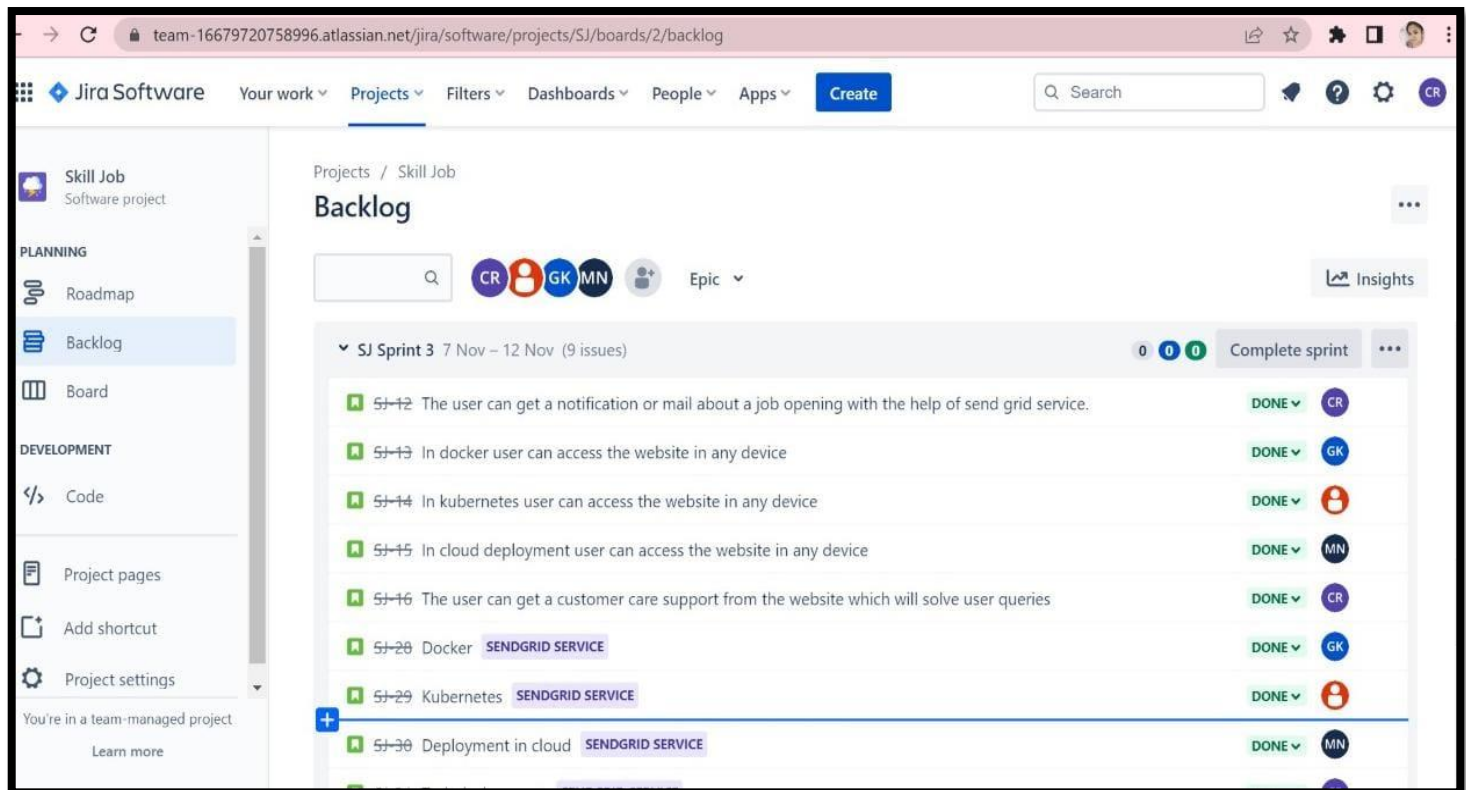


Fig: Sprint-3

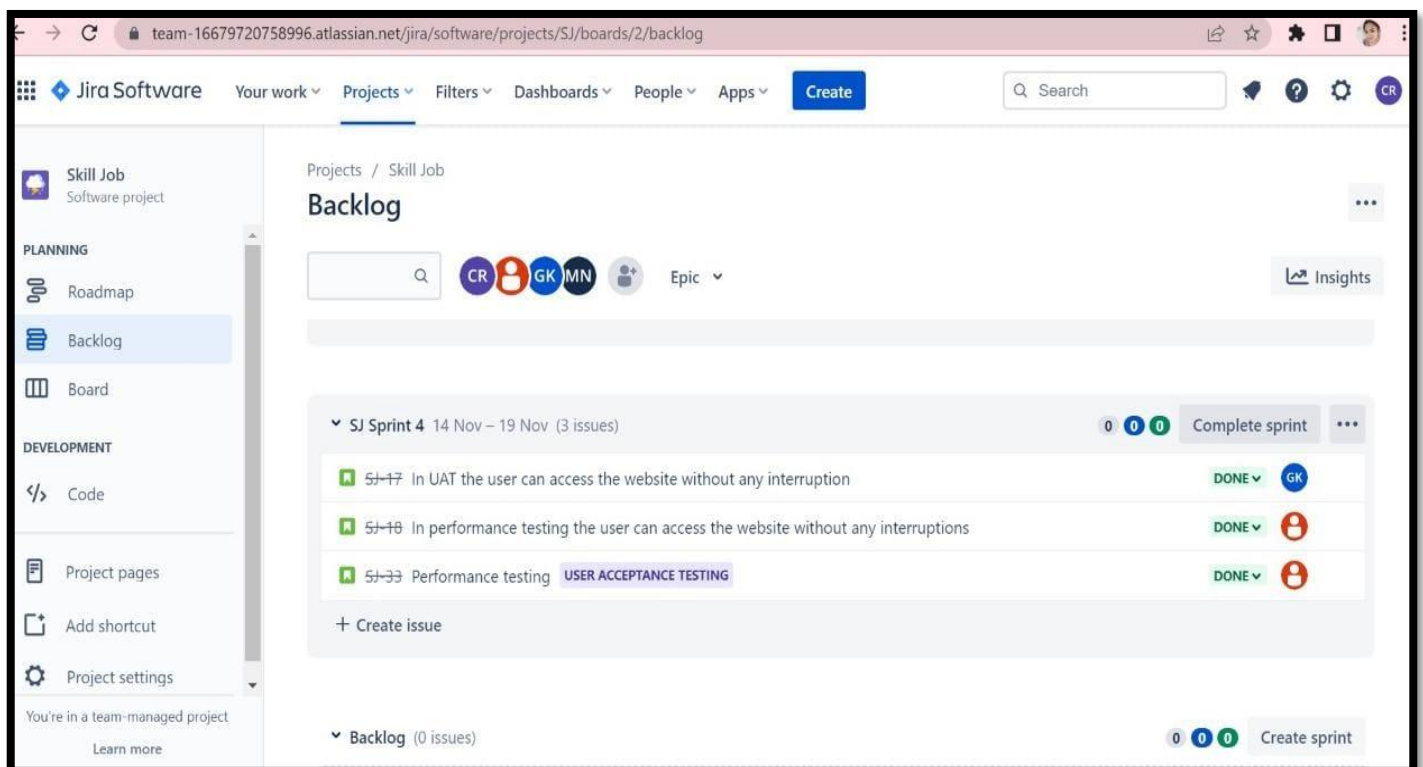


Fig: Sprint-4

7. CODING AND SOLUTIONING:

7.1 FEATURE 1:

Job rec:

This feature helps the endusers to kow about the company details:

Defmain():

```
# 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','google']

    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']

```

7.2 FEATURE 2:

Chatbot using IBM Watson:

This chatbot feature is a computer program that simulates and processes human conversation (either written or spoken), allowing humans to interact with digital devices as if they were communicating with a real person.

```

<script>
window.watsonAssistantChatOptions = {
  integrationID: "81109907-b52e-4569-a5e6-c4e1c7ecb072", // The ID of this integration.
  region: "jp-tok", // The region your integration is hosted in.
  serviceInstanceID: "62023895-efe0-4ba1-97bf-76869f237fc4", // 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>

```


8. TESTING:

8.1 TEST CASES :

				Date	9-Nov-22				
				Team ID	PNT2022TMD28999				
				Project Name	Project - Skill and job recommend				
				Maximum Marks	4 marks				
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Signup popup displayed or not	https://login.html/	Login/Signup popup should display	Working as expected	Pass
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Signup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link	https://homepage.html/	Application should show below UI elements: a. email text box b. password text box c. Login button with orange colour d. New customer? Create account link e. Last password? Recovery password link	Working as expected	pass
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL(https://homepage.html/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: gayathri23@gmail.com password: boot#23	User should navigate to user account homepage	Working as expected	
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL(https://login.html/) and click go 2.Click on My Account dropdown button 3.Enter invalid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: gayathri23@gmail.com password: boot#23	Application should show 'Incorrect email or password' validation message.	working as expected	pass
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL(https://login.html/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter invalid password in password text box 5.Click on login button	Username: gayathri23@gmail.com password: booting	Application should show 'Incorrect email or password' validation message.	working as expected	pass
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL(https://login.html/) and click go 2.Click on My Account dropdown button 3.Enter invalid username/email in Email text box 4.Enter invalid password in password text box 5.Click on login button	Username: Gayathri@gmail.com password: boot23	Application should show 'Incorrect email or password' validation message.	working as expected	pass
Recommender system_TC_OO6	Functional	job vacancy	verify user is able to view the vacancies posted		1.Enter URL and click go 2.Click on dropdown button 3.Enter qualification and experiences in search box 4.click on search button	Qualification : B.Tech Experience : Fresher	Application should show 'list of vacancies based on their qualification and experiences	Working as expected	pass

8.2 USER ACCEPTANCE TESTING:

1.Defect Analysis :

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

2. Test Case Analysis:

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	2	0	0	2
Client Application	2	0	0	2
Security	1	0	0	1
Outsource Shipping	1	0	0	1
Exception Reporting	2	0	0	2
Final Report Output	1	0	0	1
Version Control	1	0	0	1

9. RESULTS

9.1 PERFORMANCE METRICS:



Fig : Home Page

The screenshot shows the 'Registration Form' page. The header is teal with the title 'JOB SEARCH' on the left and navigation links 'Home', 'Login', 'View jobs posted', and 'Apply' on the right. The main content area is light blue and contains a registration form with the following fields: 'First Name:' with a placeholder 'Enter your first name', 'Last Name:' with a placeholder 'Enter your last name', 'Email:' with a placeholder 'it should contain @..', 'Mobile:' with a placeholder 'only 10 digits are allowed', 'Gender:' with radio buttons for 'Male' and 'Female', 'Date Of Birth:' with a placeholder 'dd - mm - yyyy' and a calendar icon, and 'Address:' with a large text area.

Fig: Registration Page

JOB SEARCH

[Home](#) • [Register](#) • [View jobs posted](#) • [Apply](#)

Username

Enter Username

Password

Enter Password

Login

☒ Remember me

Cancel

Forgot [password?](#)

Fig : Login Page

JOBSEARCH

Home

Register

Login

Apply

All Job Posts

Job Name	Job Description	Minimum Salary	Maximum Salary	Experience	Qualification	Action
Software developer	Developing a software	Rs.15,000	Rs.25,000	Fresher	B.TECH	Apply
Software Designer	Designing a software	Rs.18,000	Rs.30,000	Fresher	B.TECH	Apply
Tester	Testing the application	Rs.20,000	Rs.35,000	Fresher	B.E	Apply
Project Engineer	Responsible for the project development	Rs.40,000	Rs.55,000	Fresher	B.E	Apply

Copyright © 2018-2019 JobDeck.com

Fig : Job Vacancies Page

JOB SEARCH

[Home](#)
[Login](#)
[Register](#)
[View jobs posted](#)

Job

Application for *

Software Developer

Personal data

First name *

Family name *

Citizenship

Date of birth

dd-mm-yyyy

Address

ZIP Code

City

Fig : Apply for a job page

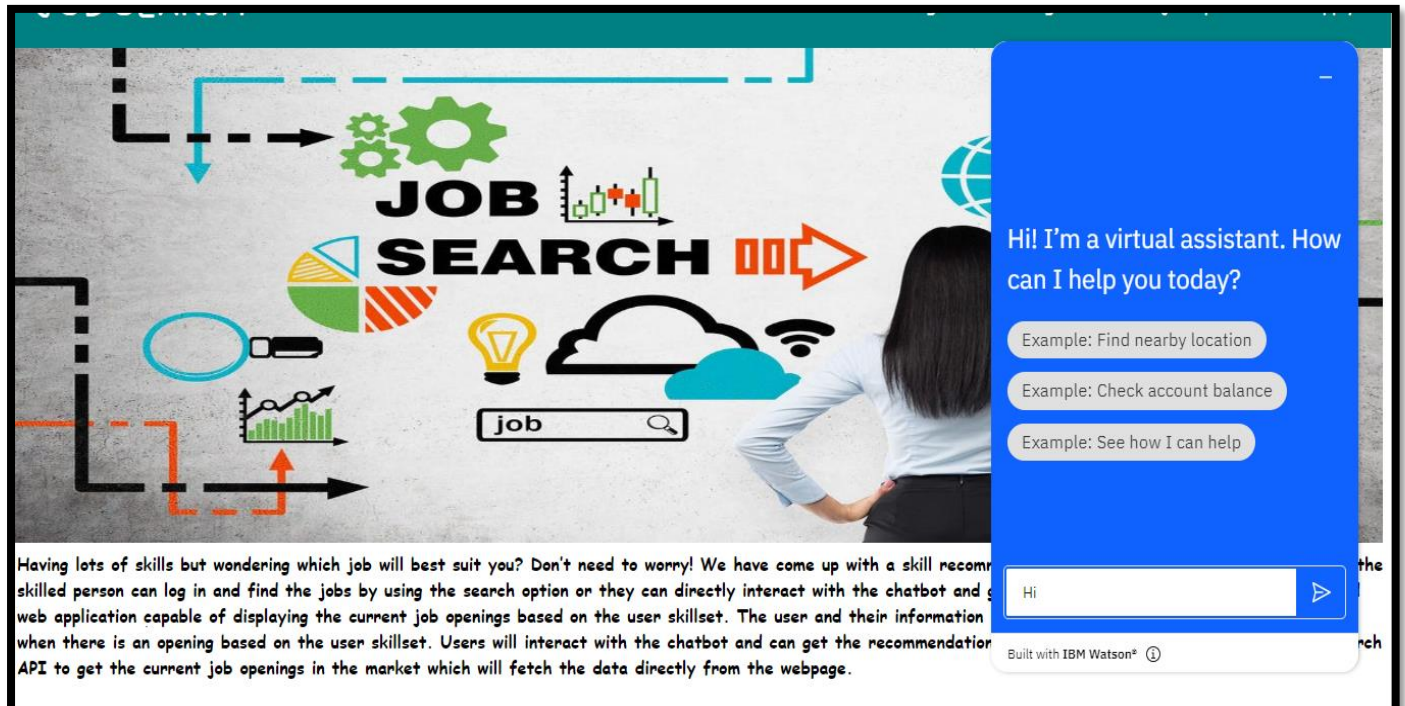




Fig : Virtual Assistant (CHATBOT)






Address
Medavakkam
Chennai-100



Phone
98 9893 5647
96 3434 5678



Email
jobsearch@gmail.com
info.jobsearch@gmail.com

Send us a message

If you have any queries regarding about job vaccancies. It's my pleasure to help you.

Enter your name

Enter your email

Send Now

Fig: Contact Us

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Bidirectional recommendation.
- Students can register online instead of going to the placement department for registration.
- Various information retrieval techniques are used.
- There will be no need to notice or emailing every student about the company coming to the college.
- The students can keep updated themselves through this software.
- Use ontology to categorize jobs and as a knowledge base to define features.
- Effective matching methods.
- Use integration-based similarity in skills matching.
- The company can view all student's details and the system can shortlist students according to their criteria instead of doing manually.
- There is an admin login that can view and manage both student's and company's accounts and can also put up notifications.

DISADVANTAGES:

- Knowledge acquisition and knowledge engineering problems.
- Uninterrupted internet connection is required for smooth functioning of application.

11. CONCLUSION

From a proper analysis of positive points and constraints , it can be safely concluded that the project is a highly efficient, cloud based application and GUI based component. This application is working properly and meeting user requirements. This component can be easily plugged in many other systems. Nowadays manual process of searching a job of one's choice as well as searching the appropriate candidate for a specific job has become a huge task and so realizing the need for easy management of this process, the site has been developed. It is very easy to update and maintain information through this site. . The main features of this site include exibility, ease of manipulation of information, easy access searching, storage, reduction of manual work in an efficient manner, a quick, convenient,

reliable, timely and effective way to reach recruiting , search and employment professionals worldwide and it is also very economical. The project could very well be enhanced further as per the requirements.

12. FUTURE SCOPE

This system is helpful for the college campus recruitment process for training and placement department to shortlist students in advance and prepares students for placements. The system will be helpful to the students to get a rough idea about their CV.

Goals:

- Reduced entry work.
- Easy retrieval of information
- Reduced errors due to human intervention
- User friendly screens to enter the data
- Portable and exile for further enhancement

13. APPENDIX

SOURCE CODE:

```
fromfunctools
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(SKilDictionary, 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

```

```

Jobs
# Iterate through the whole job description dataset to pick 20 similar Job description for each Top10
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)
```

GITHUB AND PROJECT DEMO LINK:

<https://github.com/IBM-EPBL/IBM-Project-11622-1659336389>

https://drive.google.com/folderview?id=1Ox43rCuiOc2y_Y4Nw6CX4SuzIHDSp7-y

