IBM NALAIYATHIRAN PROJECT

SKILL/JOB RECOMMENDER APPLICATION

Team ID	**PNT2022TMID04959**
Project Name	**Skill/Job Recommender Application**
Team Members	**JANANI R(921319104069)** **HEMAROSHINI M (921319104060)** **KAVIYA M(921319104096)** **MALAVIKA A C(921319104108)**

CHAPTER 1

INTRODUCTION

OVERVIEW:

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.

PURPOSE:

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.

CHAPTER 2

LITERATURE SURVEY

SURVEY PERFORMED:

Reference1	Title	Job Recommendation based on Job Seeker Skills: An Empirical Study
	Authors	JorgeValverde-Rebazaet.al.
	Journal Detail	2018-DepartmentofScientificResearch,Visibilia,SP,Brazil

Inference from reference1:

Algorithm used:

Term Frequency-Inverse Document Frequency (TF-IDF) & word2vec, Continuous Bag -of- Words (CBOW) and Skip-gram

Merits:

Word2Vec-SkipGramscore-0.590Precision-0.814ME-0.96

Demerits:

Less accuracy in the correctness of user data.

Reference2	Title	Toward the next generation of recruitment tools: An online social network-based job Recommender system
	Authors	MDiaby, EViennet, and TLaunay.

	Journal Detail	2013 Advances in Social Networks Analysis and Mining, ASONAM

Inference from reference 2:

Algorithm used:

Work4, Support vector Machine

Merits:

For data processing two types of data are used: input- interaction data (user's own data) and social connections data(user's friends data)

Demerits:

Sensitive contents of user are prone to vulnerability.

Reference 3	Title	Matching resumes and jobs based relevance models
	Authors	XingYi , JamesAllan , W.BruceCroft
	Journal Detail	2007 Special Interest Group on Information Retrieval(SIGIR)

Inference from reference3:

Algorithm used:

Structured Relevance Models (SRM)

Merits:

Relevance model makes matching process easier

Demerits:

Only for modeling and retrieving semi-structured documents

Reference 4	Title	Collaborative filtering based online recommendation systems
	Authors	Basit Mehmood Khan et.al.
	Journal Detail	2017 International Conference on Information and Communication Technologies (ICICT)

Inference from reference 4:

Algorithm used:

Collaborative filtering are item based and user based approaches

Merits:

CF algorithms are classified as memory- based approaches and model-basedapproaches and compared

Demerits:

Interest of mobile users may lead to the rejection of skilled candidate

Reference 5	Title	Job Recommendation System Using ProfileMatching and Web- Crawling
	Authors	Deepali V Musale et.al.
	Journal Detail	2016 International Journal of Advance Scientific Research And Engineering Trends

Inference from reference 5:

Algorithm used:

Semantic matching, tree-based knowledge matching and query matching.

Merits:

On campus recruitment process made easier using web crawling

Demerits:

Dataset is taken only from reputed institution and guarantee to employ allstudents is less.

PROBLEM STATEMENT DEFINITION:

Job recommender system have become popular since they reduce information overload by generating personalized job suggestions. Although there exist a variety of techniques and strategies for job recommendation, most of them fail to recommend job vacancies that fit properly to the job seeker skillset. Our system proposes an end-to-end application which recommends the best fit job that matches the users profile. The fresher or the skilled person can log in and find the jobs using the search option or they can directly interact with the chatbot and get their dream job. The seeker can interact with the chatbot and get the recommendations based on their skills. The user and their information are stored in the Database. An alert is sent when there is an opening based on their skillset. This System accomplishes a significant success in a broad range of applications and potentially a powerful searching and recommending technique

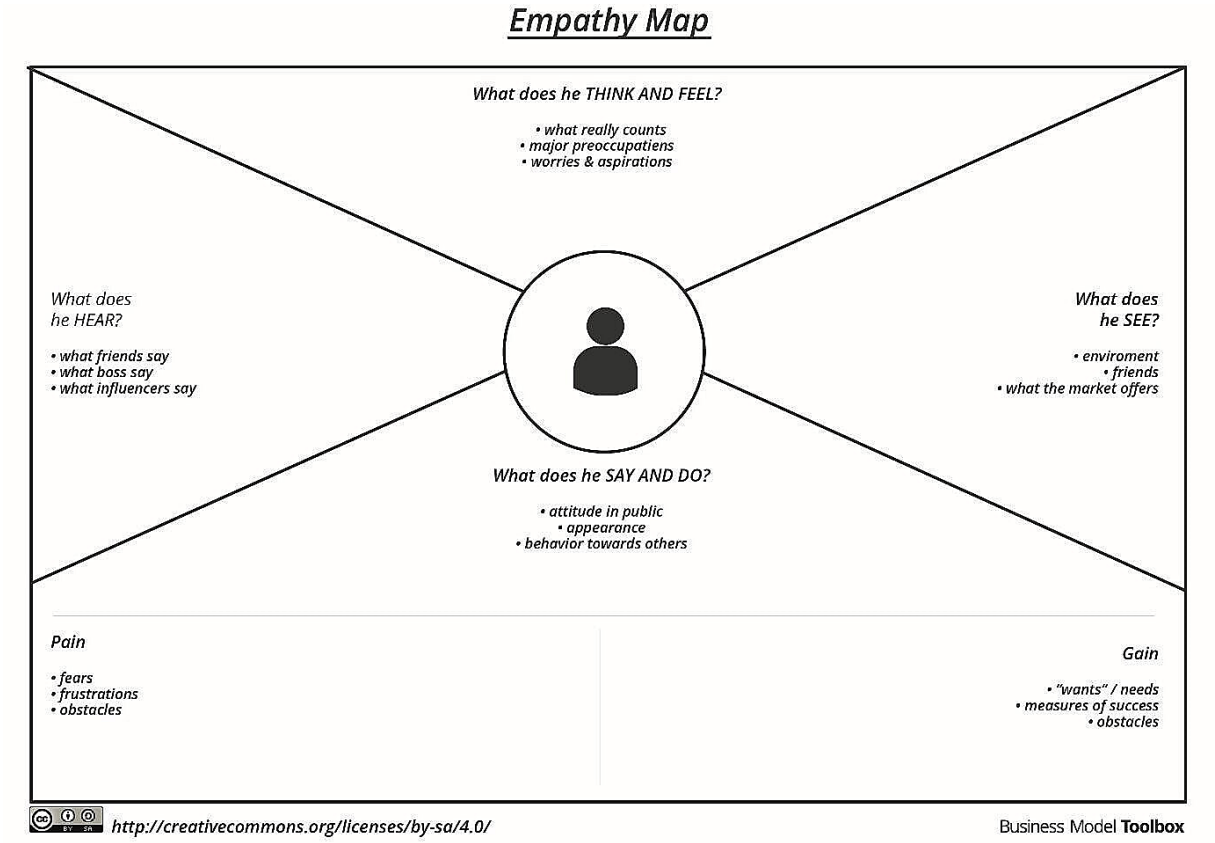
CHAPTER 3

IDEATION & PROPOSED SOLUTION:

EMPATHY MAP CANVAS:

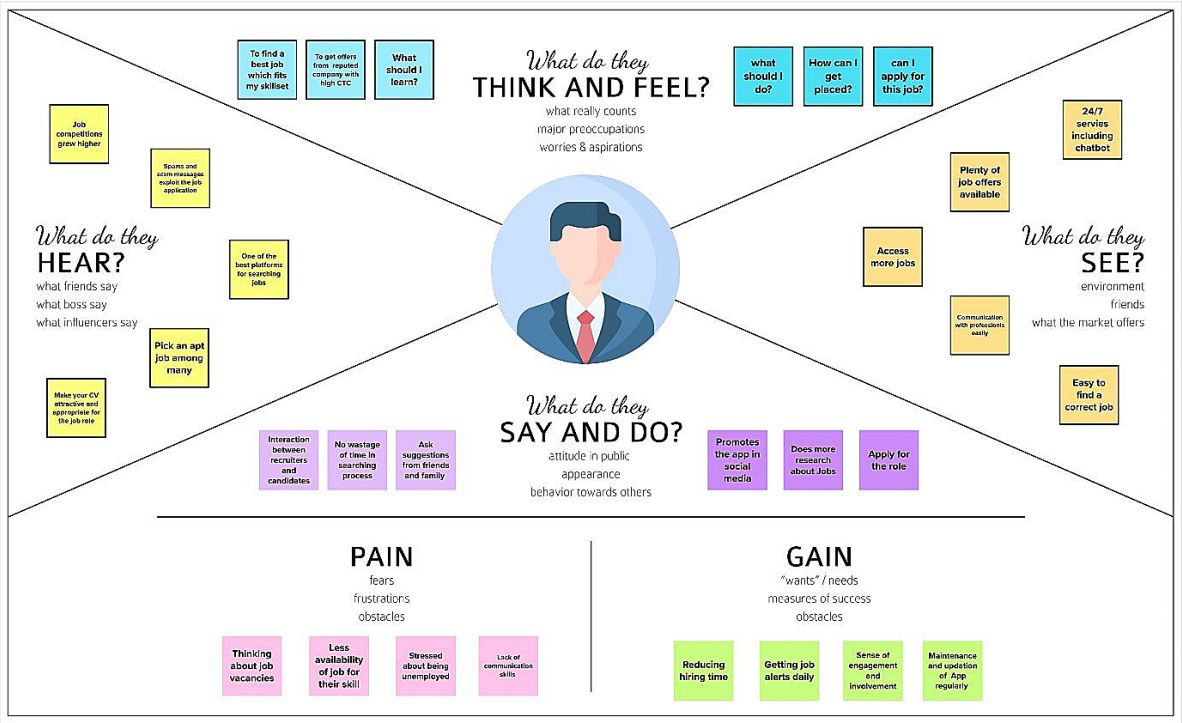
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Example:



Reference: https://www.mural.co/templates/empathy-map-canvas

Skill/Job Recommender Application:



IDEATION AND BRAINSTROMING:

Step-1: Team Gathering, Collaboration and Select the Problem Statement

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

How might we able to come up with a better solution for skill and job recommender applications ?

PROBLEM

How might we able to create a better application for Job seekers?

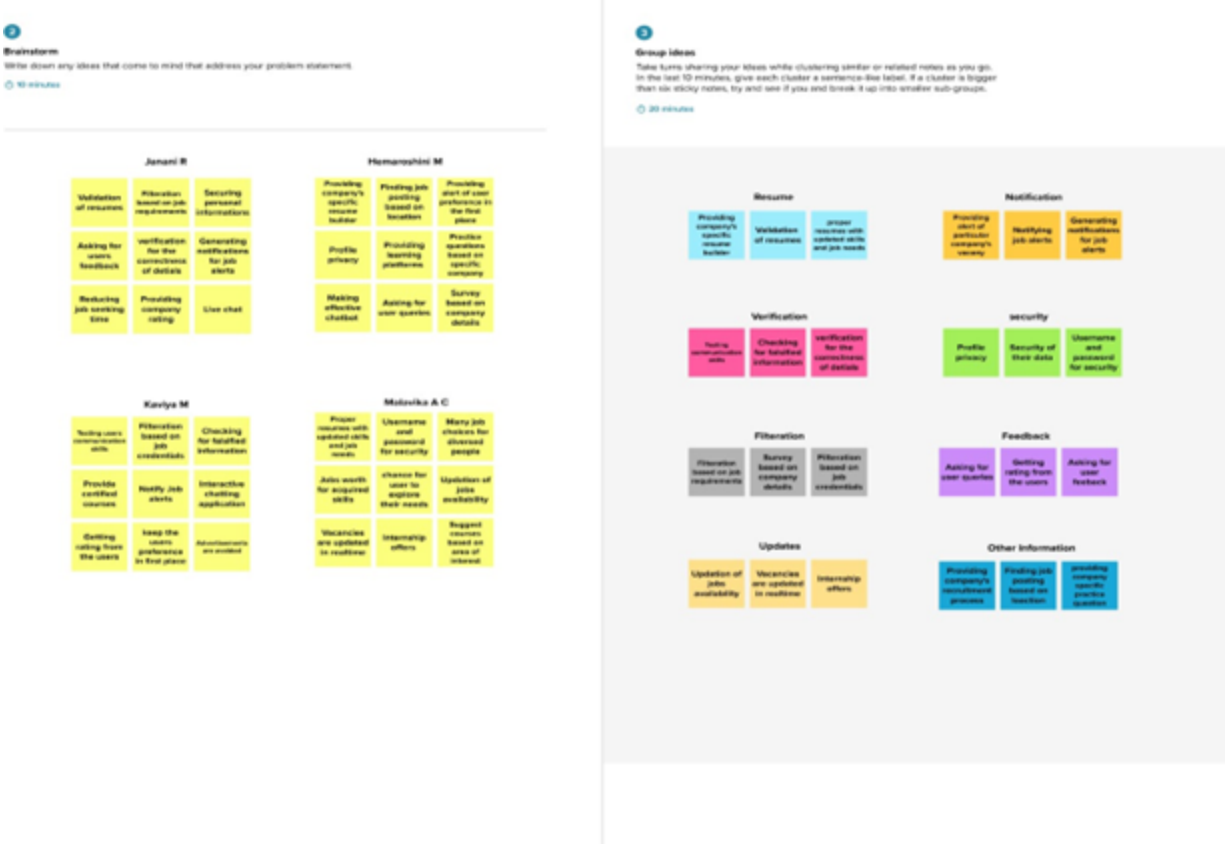
PROBLEM

How might we able to incorporate chatbot for better recruitment process?

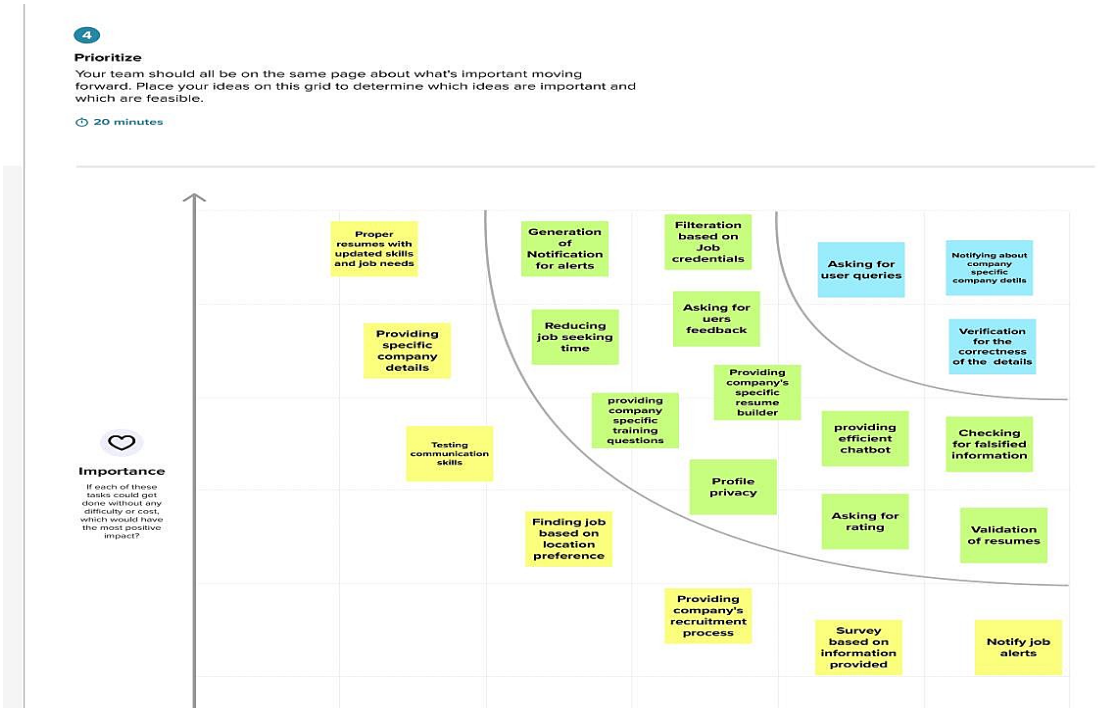
PROBLEM

How might we able to make the selection for the career of Job Seekers easy by efficient filtering process?

Step-2: Brainstrom, Idea Listing and Grouping



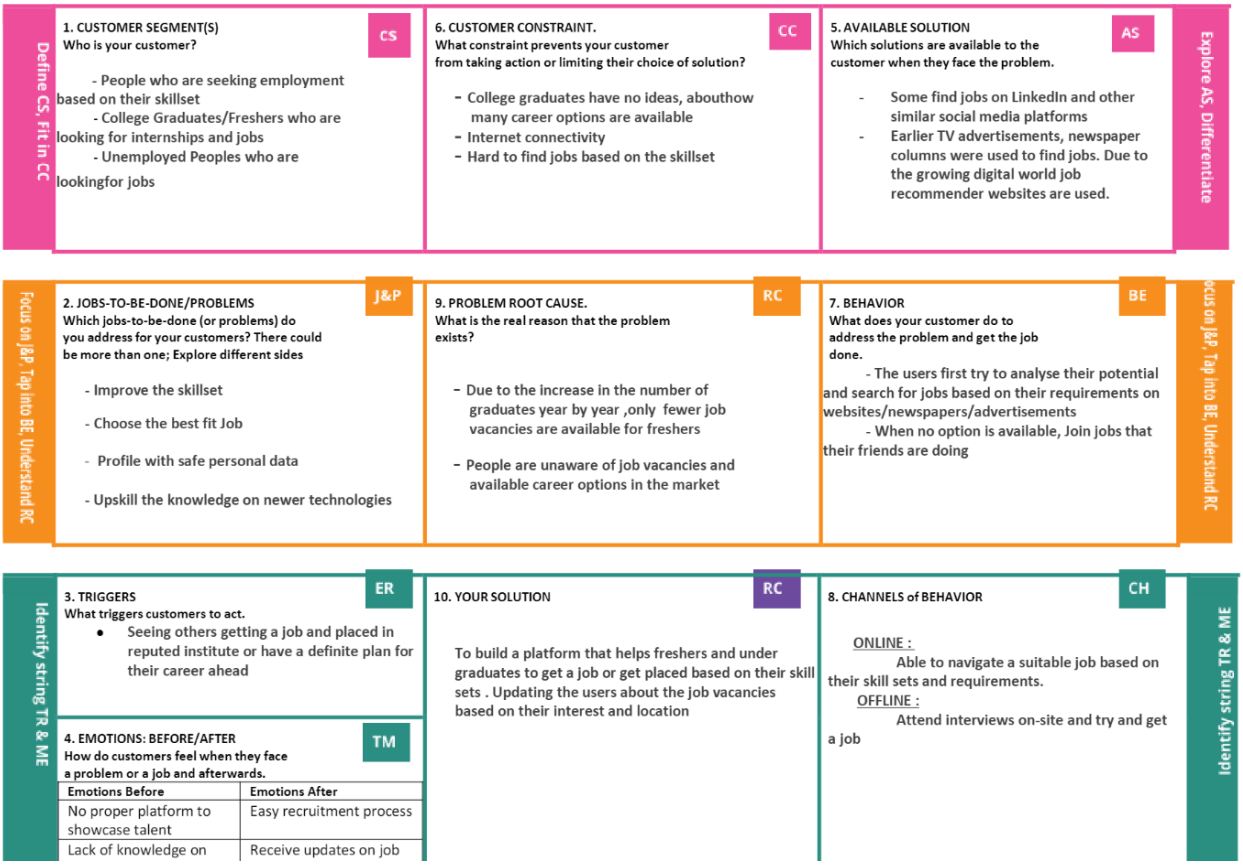
Step-3: Idea prioritization



PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Although there exist a variety of techniques and strategies for job recommendation, mostof them fail to recommend job vacancies that fit properly to the job seeker skillset.
2.	Idea / Solution description	Our system proposes an end-to-end application which recommends the best fit job that matches the user's profile. The fresher or the skilled person can log in and find the jobs using the search option or theycan directly interact with the chatbot and get their dream job
3.	Novelty / Uniqueness	- Our system notifies the user whether their application is accepted or not via the chatbotwhich saves time and effort. - Refinement of the job fields
4.	Social Impact / Customer Satisfaction	-Users stay up to date of the offers. -The user was allowed to choose the required job as per his/her skill level. It helps user to make right decision to choose their required field job
5.	Business Model (Revenue Model)	-We can provide the application for job seekers in a subscription based. We can share the profiles with companies and generate the revenue by providing them bestprofiles. -Updating the new technologies and joboffers regularly.
6.	Scalability of the Solution	In this system, we demonstrate a chatbot that uses Artificial Intelligence to produce dynamic responses to online client enquiries. Data can be scaled up and scaled down according to number of current job openings available

PROBLEM SOLUTION FIT:



CHAPTER 4

REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENTS:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Email.
FR-2	User Confirmation	Confirmation via EmailConfirmation via OTP
FR-3	User Login	Login through Email Sign in through username and password
FR-4	Setting up User Profile	Complete user profile by providing personal detailsUpload resume and certificates Update the Skills and Experience
FR-5	Searching jobs	Search jobs based on the skillset, experience and qualification
FR-6	Job Recommendation	Recommending Job by querying data from DB2Database
FR-7	User Acceptance	Confirmation of the job

NON-FUNCTIONAL REQUIREMENTS:

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

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Simple and Easy to use interface
NFR-2	Security	This application is secure with the personalized Login credentials. Using SSL Certificate for DB2.
NFR-3	Reliability	Data cannot be leaked easily. More updates on the software and the system.
NFR-4	Performance	Well Optimized application. Compatible and portable to every device.
NFR-5	Availability	Chatbot available to help 24/7 hrs. Avail every feature without any subscriptions.

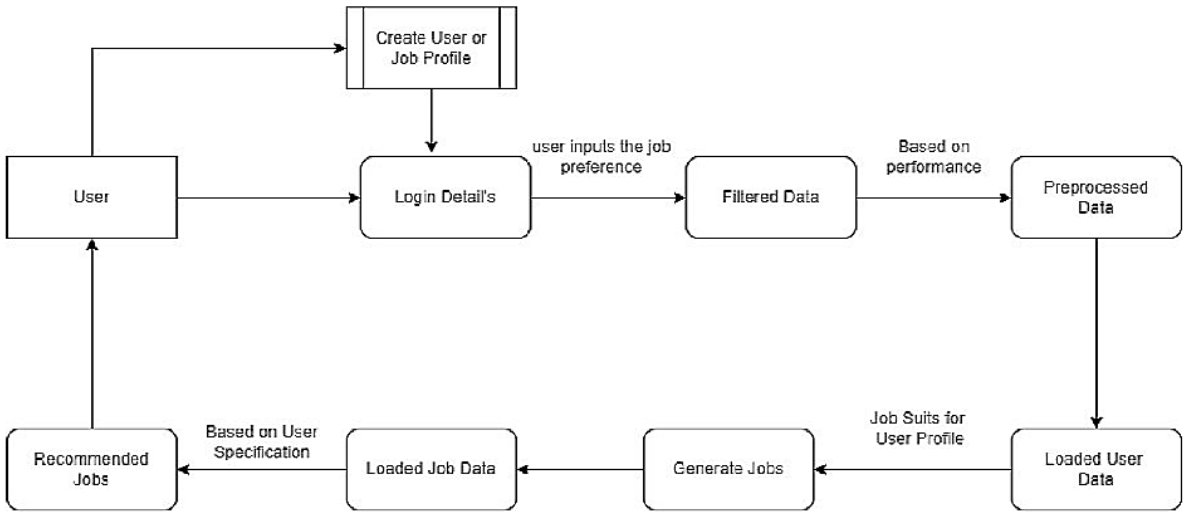
NFR-6	Scalability	Can easily extend the app as we are using kubernetes and docker containers

CHAPTER 5

PROJECT DESIGN

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 datais stored.



TECHNICAL ARCHITECTURE:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Guidelines:

1. Every process must be included as a part of application logic or technology block

2. Establish infrastructure boundaries (local / cloud)
3. Identify external interfaces (such as third-party APIs).
4. List the components and services for data storage.
5. Describe the machine learning model interface (if applicable)

Table-1: Components & Technologies:

S.No	Component	Description	Technology
1.	Registration	How user interacts with application by entering email, password and confirming password.	HTML, CSS, JavaScript
2.	Login	A user can log into the application by entering email& password.	HTML, CSS, JavaScript
3.	User Details	A User can fill personal details like Mobile number,Blood group, Gender, Address, etc.	HTML, CSS, JavaScript
4.	User Experience	Interaction between user and application	HTML, CSS, JavaScript/Angular JS/React JS
5.	Creating an Interface	Making an application for the job searching	Java/Python
6.	Voice Support	Using voice commands for searching rather thantyping	IBM Watson STT service
7.	Chat bot Support	Interface for interaction with user	IBM Watson Assistant
8.	Database	Data type, configurations, etc.	MySQL, NoSQL, etc.
9.	Online Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
10.	Keeping Files	File storage requirements	IBM Block Storage or Other StorageService or Local File system
11.	Machine Learning Model	Usage of Machine Learning Model	Object Recognition Model, etc.
12.	Infrastructure	Application Deployment on Local System / Cloud:Local	Local, Cloud Foundry, Kubernetes, etc.

		Server Configuration Cloud Server Configuration	

USER STORIES:

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**	**Priority**	**Release**
Customer (Web)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / Dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email &click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & accessthe dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the	I can register & access the dashboard	Medium	Sprint-1

			application through Gmail	with GmailLogin		
	Login	USN-5	As a user, I can log into the application byentering email & password	I can access the dashboard	High	Sprint-1
	Search	USN-6	As a user, I can search for the desired companies	Companies related to the search terms are listed	High	Sprint-2
	Apply	USN-7	As a user, I can apply for a compa ny	Application is submittedto the company	High	Sprint-2
	Review	USN-8	As an admin, I must forward the applications to the respective companies	Application is submittedto the company	Medium	Sprint-2
Administr ator	Forward	USN-9	As an admin, I must forward the applications to the respective companies	The application is received by the company	High	Sprint-1
	Send Confirma tion	USN-10	A confirm ation mail is sent from the respect	Confirmation is receivedby the user	High	Sprint-2

			ed compa ny			
	Manage review	USN-11	As an admin, I must make the reviews appear on the company ’s profile	Reviews appear on the company’s page	Low	Sprint-2

Table-2: Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Describe the utilized open-source frameworks	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,Micro-services)	Artificial intelligence (AI)
4.	Availability	Justify the application's availability	RAID (redundant array of independentdisks)
5.	Performance	Application performance (number of requests per second, use of Cache, use ofCDNs, etc.) was taken into account duringdesign.	DRAM or flash memory

References:

https://c4model.com/

https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/

https://www.ibm.com/cloud/architecture

https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda

CHAPTER 6

PROJECT PLANNING & SCHEDULING

SPRINT PLANNING & ESTIMATION:

Product Backlog, Sprint Schedule, and Estimation:

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	UI Creation Creating Registration page,Login page	10	High	Janani Hemaroshini Kaviya Malavika
Sprint-1	Database Connectivity	USN-2	Viewing and applying jobs Connecting UIwith Database	10	High	Janani Hemaroshini Malavika
Sprint-2	Send Grid Integration	USN-3	Send Grid Integration with PythonCode	10	Low	Kaviya Malavika
Sprint-2	Chatbot Development	USN-4	Building a chatbot	10	Medium	Janani Hemaroshini
Sprint-3	Integration and Containerization	USN-5	Integrating chatbot to the HTML pageand containerizing the app.	20	High	Janani Hemaroshini Kaviya Malavika
Sprint-4	Upload Image and deployment	USN-6	Upload the image to the IBM Registry anddeploy it in the Kubernetes Cluster	20	High	Janani Hemaroshini Kaviya Malavika

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

Velocity:

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

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Burn down Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies suchas Scrum . However, burn down charts can be applied to any project containing measurable progress over time.

https://www.visual-paradigm.com/scrum/scrum-burndown-chart/https://www.atlassian.com/agile/tutorials/burndown-char

Reference:

https://www.atlassian.com/agile/project-management

https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software

https://www.atlassian.com/agile/tutorials/epics

https://www.atlassian.com/agile/tutorials/sprints

MILESTONE AND ACTIVITY LIST:

Title	Description	Date
Literature Survey and Information Gathering	Gathering Information by referring the technical papers, research publications etc.	19 SEPTEMBER 2022
Prepare Empathy Map	To capture user pain and gains Prepare List of Problem Statement	19 SEPTEMBER 2022
Ideation	Prioritize a top 3 ideas based on feasibility and Importance	19 SEPTEMBER 2022
Proposed Solution	Solution include novelty, feasibility, business model, social impact and scalability of solution	24 SEPTEMBER 2022
Problem Solution Fit	Solution fit document	1 OCTOBER 2022
Solution Architecture	Solution Architecture	1 OCTOBER 2022
Customer Journey	To Understand User Interactions and experiences with application	8 OCTOBER 2022
Functional Requirement	Prepare functional Requirement	15 OCTOBER 2022
Data flow Diagrams	Data flow diagram	15 OCTOBER 2022
Technology Architecture	Technology Architecture diagram	15 OCTOBER 2022
Milestone & sprint delivery plan	Activity what we done &further plans	28 OCTOBER 2022
Project Development Delivery of sprint 1,2,3 & 4	Develop and submit the developed code by testing it	28 OCTOBER 2022 – 19NOVEMBERS 2022

CHAPTER 7

TESTING

TEST CASES:

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
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.Scroll down 3.Verify login/Singup popup displayed or not	http://169.51.204.215:30106/	Login/Signup popup should display	Working as expected	PASS	Successfull			Janani Hemaroshini Kaviya Malavika
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup	1. Enter URL and click go 2.Click on Sigup button for User 3. Verify login/Singup popup with below UI elements: a.id text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link	http://169.51.204.215:30106/	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	Successful			Janani Hemaroshini Kaviya Malavika
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials	1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter valid password in password text box 5. Click on login button	ID: 5342 password: Testing123	User should navigate to user account homepage	Working as expected	PASS	Successful			Janani Hemaroshini Kaviya Malavika
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter valid password in password text box 5. Click on login button	ID: 5342 password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful			Janani Hemaroshini
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter Invalid password in password text box 5.Click on login button	ID: 5342 password: Testing123678687 6786876876	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful			Kaviya Malavika
LoginPage_TC_OO6	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter Invalid password in password text box 5. Click on login button	ID: 5342 password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	PASS	Successful			Kaviya Malavika

LoginPage_TC_OO7	Functional	Login page	Verify User is able to log into application with Valid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter Invalid password in password text box 5.Click on login button	ID: 5434 password: Testing123	Application should show 'correct email or password ' validation message.	Working as expected	PASS	Successful			Janani Hemaroshini
LoginPage_TC_OO8	Functional	Login page for ADMIN	Verify User is able to log into application with Valid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter valid password in password text box 5.Click on login button	ID: 1111 password: 5678	Application should show 'correct email or password ' validation message.	Working as expected	PASS	Successful			Janani Hemaroshini
LoginPage_TC_OO9	UI	ADMIN PAGE	Verify all the Customer database is visible	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter Invalid password in password text box 5.Click on login button	http://169.51.204.215:30106/	Customer database is visible	Working as expected	PASS	Successful			Hemaroshini Kaviya

LoginPage_TC_OO7	Functional	Login page	Verify User is able to log into application with Valid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter Invalid password in password text box 5.Click on login button	ID: 5434 password: Testing123	Application should show 'correct email or password ' validation message.	Working as expected	PASS	Successful			Janani Hemaroshini
LoginPage_TC_OO8	Functional	Login page for ADMIN	Verify User is able to log into application with Valid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter Valid ID in ID text box 4. Enter valid password in password text box 5.Click on login button	ID: 1111 password: 5678	Application should show 'correct email or password ' validation message.	Working as expected	PASS	Successful			Janani Hemaroshini
LoginPage_TC_OO9	UI	ADMIN PAGE	Verify all the Customer database is visible	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter Invalid password in password text box 5.Click on login button	http://169.51.204.215:30106/	Customer database is visible	Working as expected	PASS	Successful			Hemaroshini Kaviya

LoginPage_TC_O10	Functional	USER REGISTER	Verify Id sent to customer email address	1. Enter URL(http://169.51.204.215:30106/) and click go 1.Register the account by giving credentials 2. Click on button Submit	http://169.51.204.215:30106/	Email sent successfully	Working as expected	PASS	Successful			Kaviya Janani
LoginPage_TC_O11	Functional	AGENT REGISTER	Verify AGENT is able to log into application with Valid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter Invalid password in password text box 5. Click on login button	ID: 5342 password: Testing123	ID sent successfully	Application should show 'correct email or password ' validation message.	PASS	Successful			Malavika Hemaroshini
LoginPage_TC_O12	Functional	Login page for ADMIN	Verify User is able to log into application with InValid Credentials	1.Enter URL(http://169.51.204.215:30106/) and click go 2.Click on My Account dropdown button 3. Enter InValid ID in ID text box 4. Enter Invalid password in password text box 5. Click on login button	ID: 1111 password: 5678	Application should show 'Incorrect ID or password ' validation message.	Working as expected	PASS	Successful			Malavika Janani
LoginPage_TC_O13	UI	Home page for Agent	Verify user is able to see the agent home page when user finish on submitting Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the Agent Login page and submit Your Credentials	ID: 1111 password: 5678	AGENT Home Page popup should display	Working as expected	PASS	Successful			Janani Hemaroshini Kaviya Malavika

LoginPage_TC_O14	UI	Home page for USER	Verify user is able to see the User home page when user finish on submitting Credentials	1 Enter URL(http://169.51.204.215:30106/) and click go 2 To the User Login page and submit Your Credentials	http://169.51.204.215:30106/	USER Home Page popup should display	Working as expected	PASS	Successful			Janani Hemaroshini Kaviya Malavika
LoginPage_TC_O15	UI	Home page for ADMIN	Verify user is able to see the ADMIN home page when user finish on submitting Credentials	1. Enter URL(http://169.51.204.215:30106/) and click go 2. To the User Login page and submit Your Credentials	http://169.51.204.215:30106/	ADMIN Home Page popup should display	Working as expected	PASS	Successful			Janani Hemaroshini Kaviya Malavika
LoginPage_TC_O16	Functional	AGENT PAGE	On delete Button the user Credentials will be delected	1. Enter URL(http://169.51.204.215:30106/) and click go 2 To the Admin Page and delect on User Credentials	http://169.51.204.215:30106/	ADMIN Home Page popup should display	Working as expected	PASS	Successful			Janani Hemaroshini Kaviya Malavika

USER ACCEPTANCE TESTING:

Purpose of Document:

The purpose of this document is to briefly explain the test coverage and open issues of the [Skill/Job Recommender Application] project at the time of the release to User Acceptance Testing (UAT).

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	3	1	2	17
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	40
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	13	12	25	78

Test Case Analysis:

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

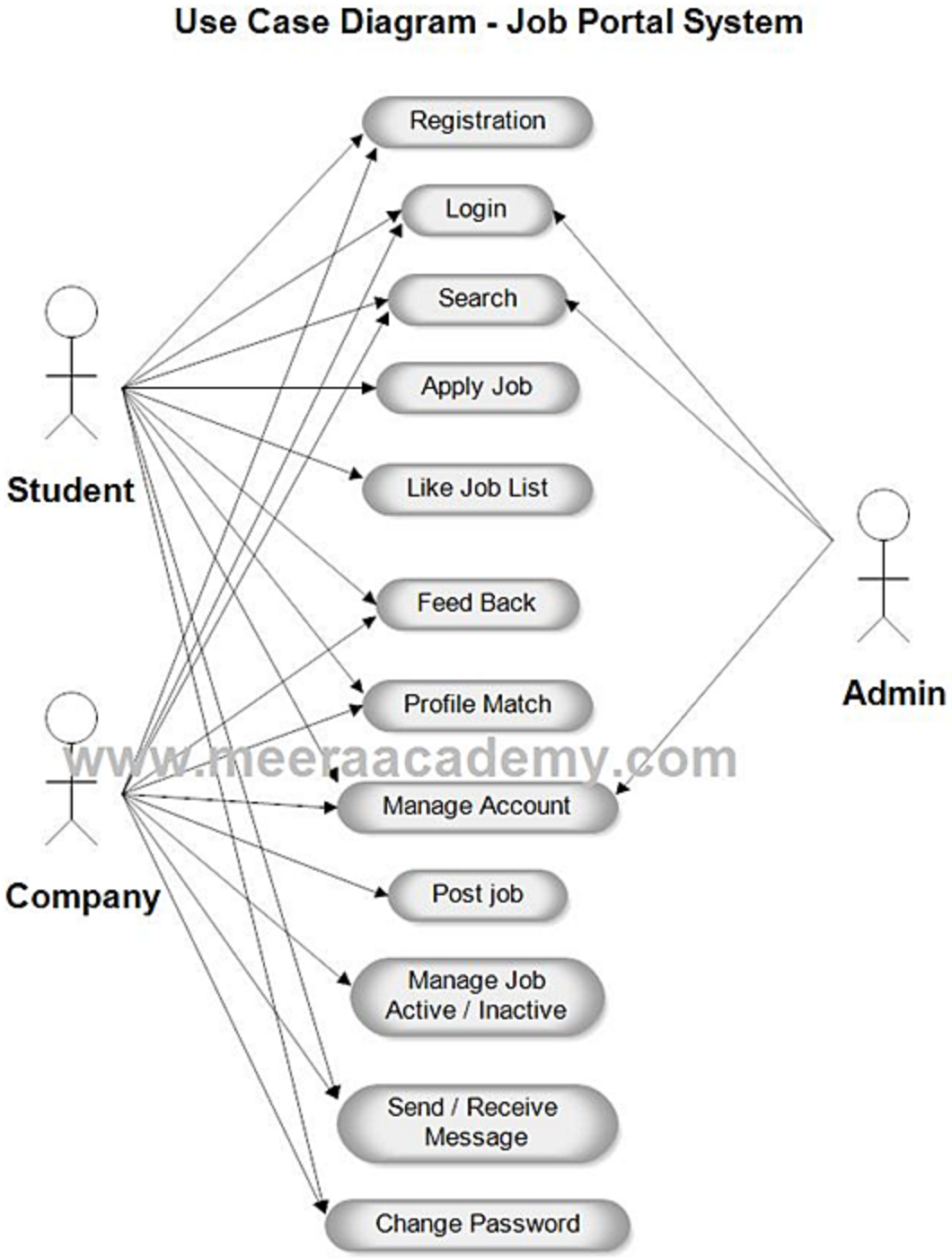
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	2	0	0	2
Outsource Shipping	3	0	0	3

Exception Reporting	8	0	0	8
Final Report Output	4	0	0	4
Version Control	2	0	0	2

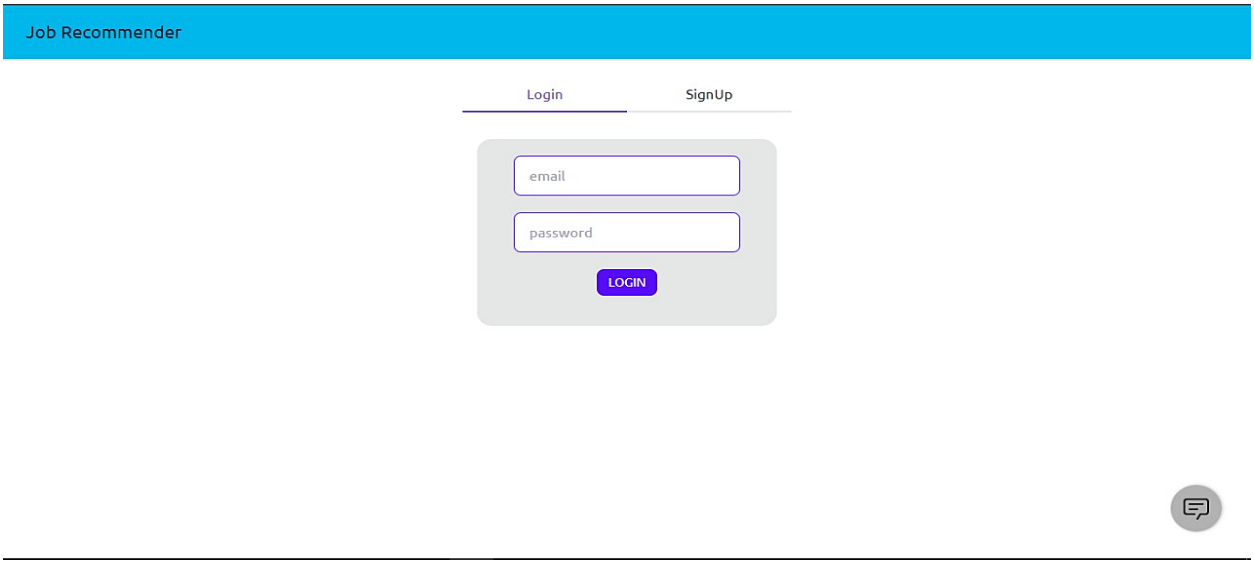
CHAPTER 8

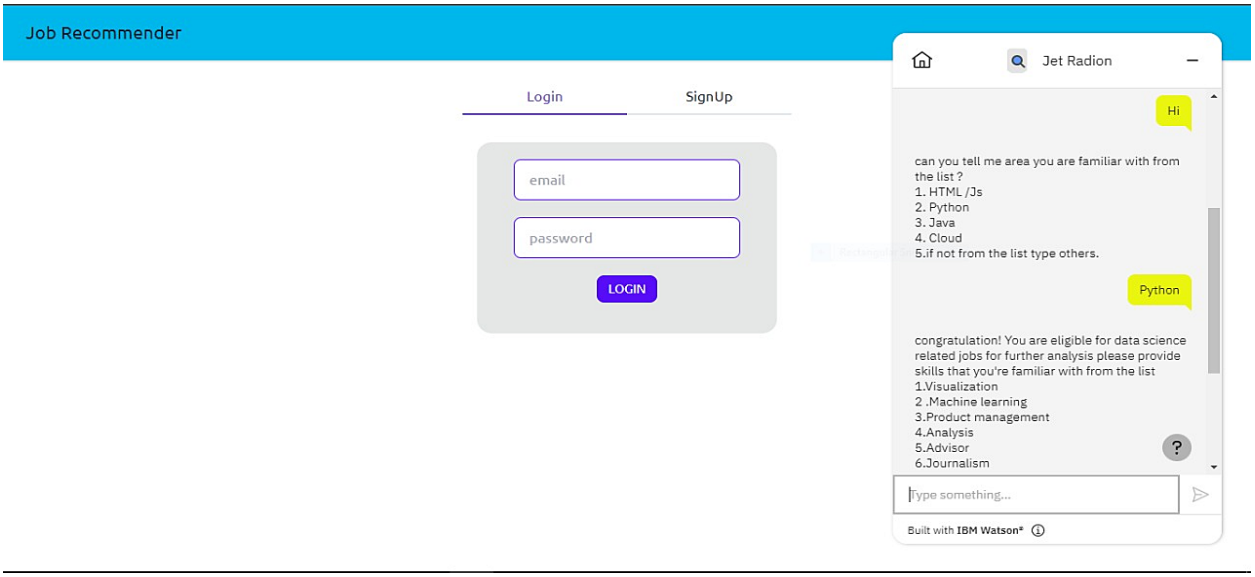
RESULT

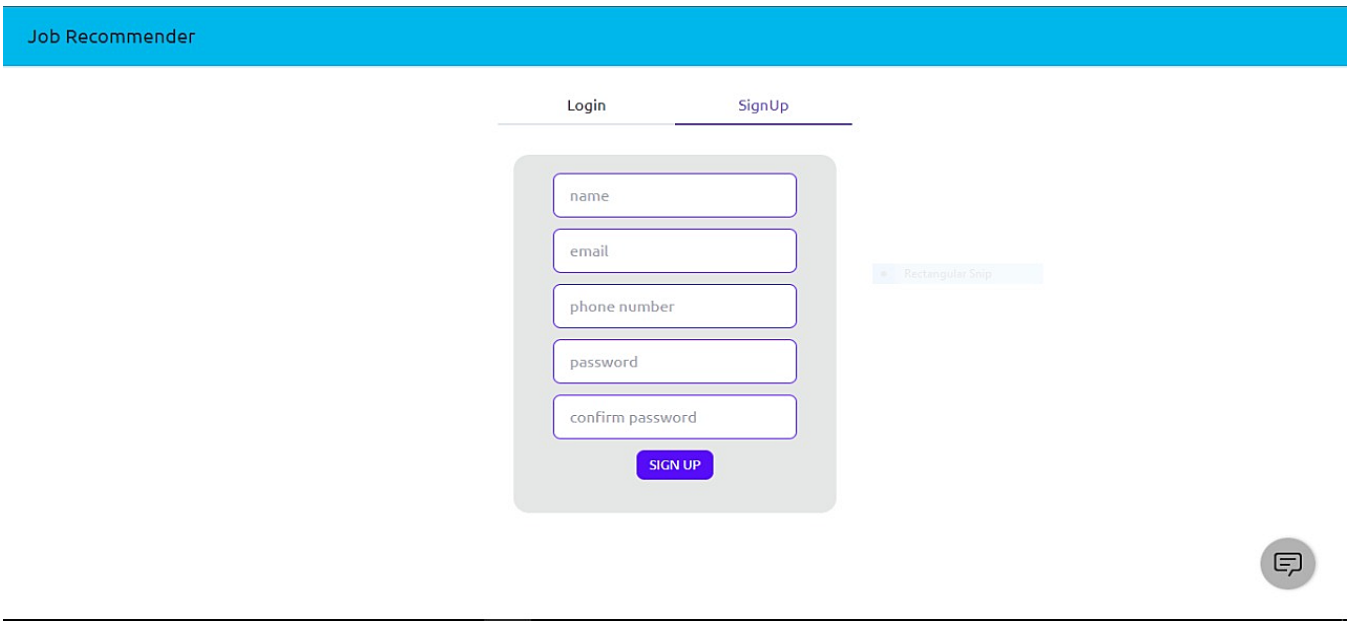
PERFORMANCE METRICES:

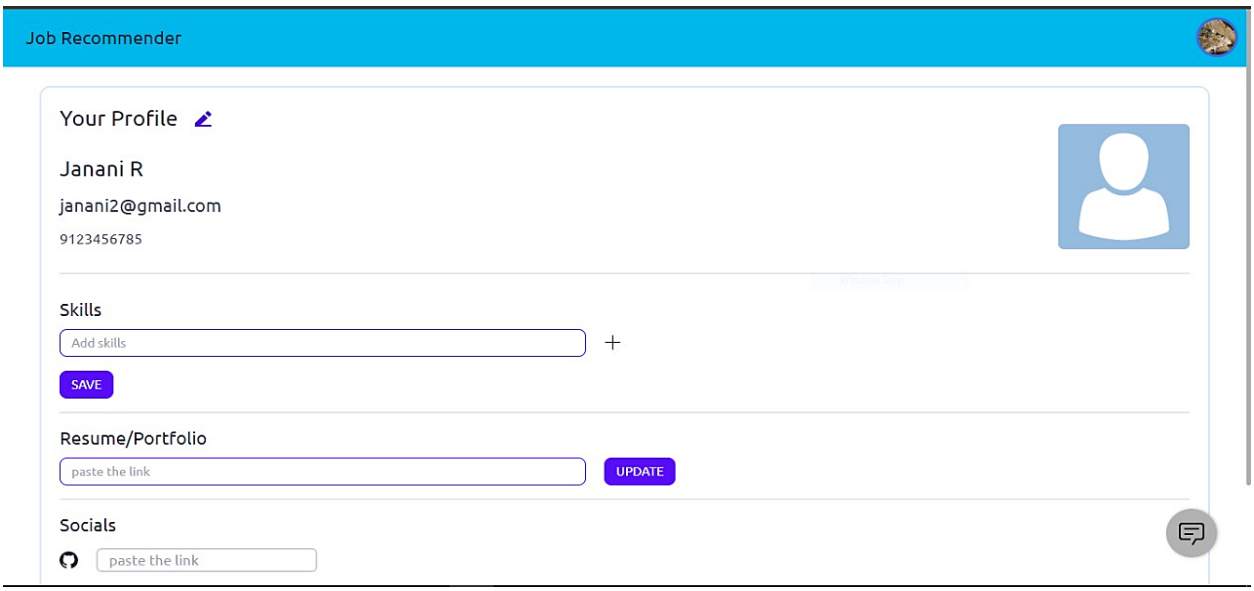


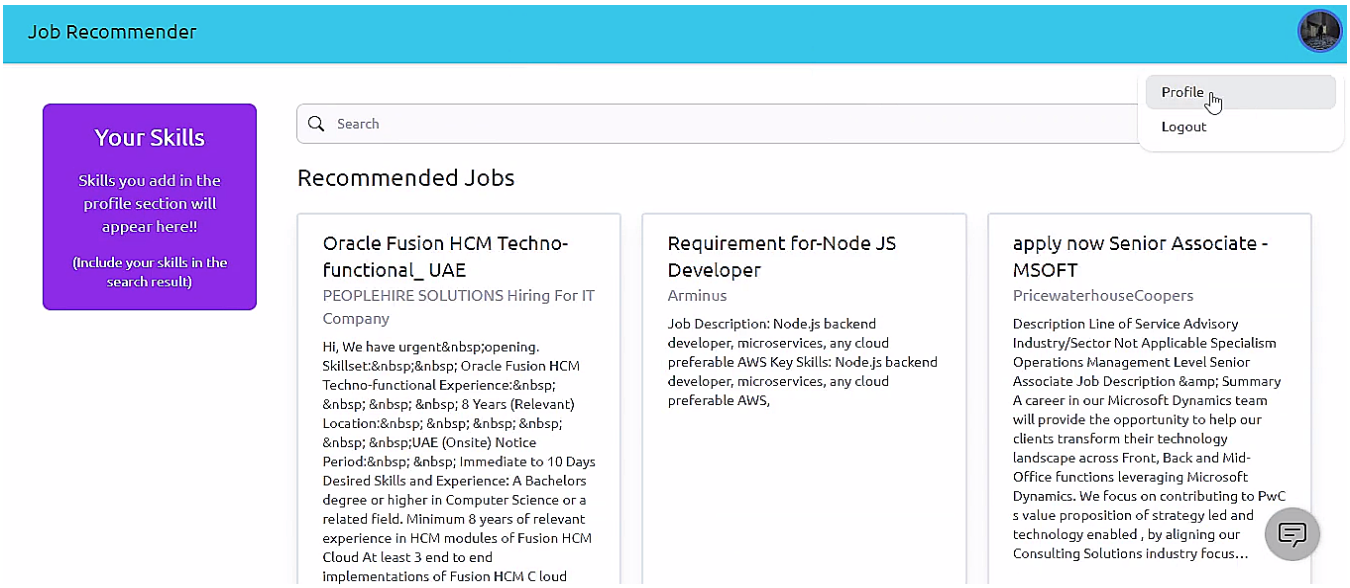
OUTCOME:




Job Recommender
Login
SignUp
email
password
LOGIN
Jet Radion
Hi
can you tell me area you are familiar with from the list ?
1. HTML /Js
2. Python
3. Java
4. Cloud
5.if not from the list type others.
Python
congratulation! You are eligible for data science related jobs for further analysis please provide skills that you're familiar with from the list
1.Visualization
2 .Machine learning
3.Product management
4.Analysis
5.Advisor
6.Journalism
Type something...
Built with IBM Watson®


Job Recommender
Login
SignUp
name
email
phone number
password
confirm password
SIGN UP


Job Recommender
Your Profile
Janani R
janani2@gmail.com
9123456785
Skills
Add skills
SAVE
Resume/Portfolio
paste the link
UPDATE
Socials
paste the link


Job Recommender
Profile
Logout
Your Skills
Skills you add in the profile section will appear here!!
(Include your skills in the search result)
Search
Recommended Jobs
Oracle Fusion HCM Techno-functional_ UAE
PEOPLEHIRE SOLUTIONS Hiring For IT Company
Hi, We have urgent opening. Skillset: Oracle Fusion HCM Techno-functional Experience: 8 Years (Relevant) Location: UAE (Onsite) Notice Period: Immediate to 10 Days Desired Skills and Experience: A Bachelors degree or higher in Computer Science or a related field. Minimum 8 years of relevant experience in HCM modules of Fusion HCM Cloud At least 3 end to end implementations of Fusion HCM C loud
Requirement for-Node JS Developer
Arminus
Job Description: Node.js backend developer, microservices, any cloud preferable AWS Key Skills: Node.js backend developer, microservices, any cloud preferable AWS,
apply now Senior Associate - MSOFT
PricewaterhouseCoopers
Description Line of Service Advisory Industry/Sector Not Applicable Specialism Operations Management Level Senior Associate Job Description & Summary A career in our Microsoft Dynamics team will provide the opportunity to help our clients transform their technology landscape across Front, Back and Mid-Office functions leveraging Microsoft Dynamics. We focus on contributing to PwC s value proposition of strategy led and technology enabled , by aligning our Consulting Solutions industry focus…

CHAPTER 9

ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- Several industries use recommendation systems to serve as the best advice for a potential buyer. Recommendation helps businesses suggest the most relevant items to buy and boost the company's revenue. These suggestions are entirely based on users' behavior and history, including information on their past preferences.

- They can obtain the most valuable insights into the consumers' tastes allowing businesses to adjust their offerings and personalize them to satisfy users' needs.

- They provide a better experience for the users by giving them a broader exposure to many different roles they are interested in.

DISADVANTAGES:

- Recommendation engines rely on data.

- If you do not have high-quality data, or cannot crunch and analyze it properly, you will not be able to make the most of the recommendation engine.

- Perhaps the biggest issue facing recommender systems is that they need a lot of data to effectively make recommendations. It's no coincidence that the companies most identified with having excellent recommendations are those with a lot of consumer user data: Google, Amazon, Netflix, etc.

CHAPTER 10

CONCLUSION

We proposed a framework for job recommendation task. This framework facilitates the understanding of job recommendation process as well as it allows the use of a variety of text processing and recommendation methods according to the preferences of the job recommender system designer. Moreover, we also contribute making publicly available a new dataset containing job seekers profiles and job vacancies. Future directions of our work will focus on performing a more exhaustive evaluation considering a greater amount of methods and data as well as a comprehensive evaluation of the impact of each professional skill of a job seeker on the received job recommendation.

CHAPTER 11

FUTURE SCOPE

- The Recommender framework encourages us to get what we need from the huge measure of data.
- It recommends the specific item instead of an over-burden of information.
- As a domain of recommender system, Job recommender system provides opportunities for job seekers by suggesting relevant jobs according to their skills to them.
- This framework is utilized by numerous organizations to recommend suitable jobs to the job seekers.
- It additionally manages the issues of data over-burden.
- As one of the domain, online recruiting system utilizes the recommender systems by providing suitable jobs to users using the user's data.

CHAPTER 12

APPENDIX

SOURCE CODE:

app.py

```
import re
import os
import ibm_db
from flask import flash
from flask import Flask
from flask import request
from flask import redirect
from bs4 import BeautifulSoup
from flask import render_template
from urllib.request import urlopen
from urllib.request import Request
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

app = Flask (__name__)

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;SECURITY=SSL;S
SLServerCertificate=DigiCertGlobalRootCA.crt;UID=txz44264;PWD=0e7ZwhLpS1dJFh9f","","")

@app.route ("/")

def home ():

    return render_template ("/homepage.html")

@app.route ("/signup.html")

def signup ():

    return render_template ("/signup.html")

@app.route ("/signin.html")

def signin():
```

```
    return render_template("/signin.html")

@app.route("/user", methods=["POST", "GET"])

def user():

    if (request.method == "POST"):

        sem = request.form.get("sem")
        spw = request.form.get("spw")
        em_lst = []
        pw_lst = []

        try:

            data = ibm_db.exec_immediate(conn, "SELECT \"EMAIL\", \"PASSWORD\" FROM
\"TXZ44264\".\"SIGNUPUSERDETAILS\";")

            while ibm_db.fetch_row(data) != False:

                em_lst. append(ibm_db.result(data, 0).replace(" ", ""))
                pw_lst.append(ibm_db.result(data, 1).replace(" ", ""))

            if (sem in em_lst and spw in pw_lst):

                return render_template("/user.html")

            else:

                return redirect("/signin.html")

        except Exception as e:

            return render_template("/error.html")

@app.route("/app", methods=["POST", "GET"])

def welcome():

    if (request.method == "POST"):

        fn = request.form.get("fn")
        ln = request.form.get("ln")
        em = request.form.get("em")
```

```
        pw = request.form.get("pw")
        cpw = request.form.get("cpw")

        message = Mail(
            from_email = "312819104004@act.edu.in",
            to_emails = em,
            subject = "Job Recommender - Account created successfully",
            html_content = '<p>You can now sign in with your credentials,<br>email: {}<br>password:
{}</p>'.format(em, pw)
        )

        try:

            insert = ibm_db.exec_immediate(conn, "INSERT INTO
\"TXZ44264\".\"SIGNUPUSERDETAILS\" VALUES ('{}', '{}', '{}', '{}');".format(fn, ln, em, pw))
            f = open("./sendgrid.txt", "r")
            sg = SendGridAPIClient(f.read())
            response = sg.send(message)

            return render_template("/welcome.html")

        except Exception as e:

            print(e)

            return render_template("/error.html")

@app.route("/job", methods=["POST", "GET"])

def job():

    if (request.method == "POST"):

        search = request.form.get("search")
        link = "https://www.glassdoor.com/Job/india-{}-jobs-
SRCH_IL.0,5_IN115_KO6,11.htm".format(search)

        return render_template("/job.html", link=link)
if __name__ == "__main__":
    app.run(debug = True)
```

JobCard.jsx

```
import React, { useEffect } from "react";

const JobCard = ({ title, company, description, link }) => {

  return (

    <div className="max-w-sm flex flex-col rounded overflow-hidden shadow-lg border-2 border-slate-200">

      <>

        <div className="px-6 py-4">

          <div className="font-bold text-xl">{title}</div>

          <div className="text mb-2 text-gray-400">{company}</div>

          <p className="text-ellipsis overflow-hidden text-gray-800 text-sm">

            {description}

          </p>

        </div>

        <div className="px-6 pt-4 pb-2 mt-auto mb-2">

          <a

            href={link}

            target="__blank"

            className="bg-transparent hover:bg-purple-400 text-purple-400 font-semibold hover:text-white py-2 mb-0 mt-4 px-4 border border-purple-400 hover:border-transparent rounded"

          >

            Apply

          </a>

        </div>

      </>
```

```
    </div>
  );
};

export default JobCard;
```

Login.jsx

```
import { useToast } from "@chakra-ui/react";
import React, { useContext, useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import { AppContext } from "../context/AppContext";
import { loginUser } from "../proxies/backend_api";
import { emailRegex } from "../utils/helper";

const Login = () => {
  const toast = useToast();
  const { setUser } = useContext(AppContext);

  const navigate = useNavigate();

  const [inputs, setInputs] = useState({
    email: "",
    password: "",
  });
```

```
const [error, setErrors] = useState({
  email: "",
  password: "",
});
const handleChange = ({ target: { name, value } }) => {
  setErrors((prev) => {
    return { ...prev, [name]: "" };
  });
  setInputs((prev) => ({ ...prev, [name]: value }));
};
const checkInputErrors = () => {
  let status = true;
  if (inputs.email.trim() === "" || !emailRegex.test(inputs.email.trim())) {
    setErrors((prev) => {
      return { ...prev, email: "Enter a valid email" };
    });
    status = false;
  }
  if (inputs.password.trim() === "") {
    setErrors((prev) => {
      return { ...prev, password: "Enter a valid password" };
    });
    status = false;
```

```
  }

  if (inputs.password.trim().length < 6) {
    setErrors((prev) => {
      return { ...prev, password: "Minimum 6 characters" };
    });
    status = false;
  }
  return status;
};
const handleLogin = async () => {
  if (checkInputErrors()) {
    const data = await loginUser(inputs);
    if (data.error) {
      toast({
        title: data.error,
        status: "error",
        duration: 3000,
        isClosable: true,
        variant: "left-accent",
        position: "top",
      });
      return;
```

```
      }
      setUser(data);
      toast({
        title: `Welcome back ${data.name}`,
        status: "success",
        duration: 3000,
        isClosable: true,
        variant: "left-accent",
        position: "top",
      });
      localStorage.setItem("user", JSON.stringify(data));
      navigate("/dashboard");
    }
  };
  return (
    <>
      <div>
        <button className="bg-base-300 rounded-box flex flex-row justify-evenly items-center gap-10
px-10 py-5 w-fit mx-auto">
          <span>Sign in with Github</span>
          <img src={`github-dark.png`} alt="github" width="14%" />
        </button>
        <div className="divider max-w-xs">or</div>
        <form
```

```
      onSubmit={(e) => e.preventDefault()}
      className="card bg-base-300 rounded-box flex flex-col justify-center items-center gap-5 px-10 py-5 w-fit mx-auto"
    >
      <div>
        <input
          value={inputs.email}
          type="text"
          name="email"
          placeholder="email"
          className="input input-bordered input-primary w-full"
          onChange={handleChange}
        />
        {error.email !== "" && (
          <p className="text-sm text-red-500 mt-1 font-medium">
            {error.email}
          </p>
        )}
      </div>
      <div>
        <input
          value={inputs.password}
          type="password"
          name="password"
```

```
          placeholder="password"
          className="input input-bordered input-primary w-full"
          onChange={handleChange}
        />
        {error.password !== "" && (
          <p className="text-sm text-red-500 mt-1 font-medium">
            {error.password}
          </p>
        )}
      </div>
      <div className="text-center">
        <button
          type="submit"
          onClick={handleLogin}
          className="btn btn-sm btn-primary mb-4"
        >
          Login
        </button>
      </div>
    </form>
  </div>
 </>
);
```

```
};
export default Login;
```

Navbar.jsx

```
import { useToast } from "@chakra-ui/react";
import React, { useContext } from "react";
import { Link, useNavigate } from "react-router-dom";
import { AppContext } from "../context/AppContext";
  const Navbar = () => {
  const navigate = useNavigate();
  const toast = useToast();
  const { user, setUser, setSkills } = useContext(AppContext);
  const logout = () => {
    setUser(null);
    setSkills([]);
      toast({
      title: "Logged out successfully!",
      status: "info",
      duration: 3000,
      isClosable: true,
      variant: "left-accent",
      position: "top",
    });
```

```
  localStorage.removeItem("user");
  navigate("/");
};
return (
  <div className="navbar bg-base-100 border-b-2">
    <div className="flex-1">
      <Link
        className="btn btn-ghost normal-case text-xl"
        to={user ? "/dashboard" : "/"}
      >
        Job Recommender
      </Link>
    </div>
    {user && (
      <div className="flex-none gap-2">
        <div className="dropdown dropdown-end">
          <label tabIndex={0} className="btn btn-ghost btn-circle avatar ">
            <div className="w-10 rounded-full ring ring-opacity-50 ring-purple-700">
              <img src="https://placeimg.com/80/80/people" />
            </div>
          </label>
          <ul
            tabIndex={0}
            className="mt-3 p-2 shadow menu menu-compact dropdown-content bg-base-100 rounded-
```

```
box w-52"
          >
            <li>
              <a
                className="justify-between"
                onClick={() => navigate("/profile")}
              >
                Profile
              </a>
            </li>
            <li>
              <a onClick={logout}>Logout</a>
            </li>
          </ul>
        </div>
      </div>
    )}
  </div>
 );
};
```

SearchBar.jsx

```
import React from "react";
```

```
import { BsSearch } from "react-icons/bs";

const SearchBar = ({ setquery, onClick }) => {
  const handlesubmit = (e) => {
    e.preventDefault();
    onClick();
  };

  return (
    <form className="flex items-center" onSubmit={handlesubmit}>
      <label htmlFor="simple-search" className="sr-only">
        Search
      </label>
      <div className="relative w-full">
        <div className="flex absolute inset-y-0 left-0 items-center pl-3 pointer-events-none">
          <BsSearch />
        </div>
        <input
          onChange={(e) => setquery(e.target.value)}
          name="search"
          type="text"
          id="simple-search"
          className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full pl-10 p-2.5  dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white dark:focus:ring-blue-500 dark:focus:border-blue-500"
```

```
          placeholder="Search"
          required=""
        />
      </div>
      <button
        type="submit"
        className="p-2.5 ml-2 text-sm font-medium text-white bg-purple-700 rounded-lg border border-purple-700 hover:bg-purple-800 focus:ring-4 focus:outline-none focus:ring-purple-300"
      >
        <BsSearch />
        <span className="sr-only">Search</span>
      </button>
    </form>
  );
};
export default SearchBar;
```

Signup.jsx

```
import { useToast } from "@chakra-ui/react";
import React, { useContext, useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import { AppContext } from "../context/AppContext";
import { registerUser } from "../proxies/backend_api";
import { emailRegex } from "../utils/helper";
```

```
const SignUp = () => {
  const toast = useToast();
  const { setUser } = useContext(AppContext);
  const navigate = useNavigate();
  const [inputs, setInputs] = useState({
    name: "",
    email: "",
    phone_number: "",
    password: "",
    confirm_password: "",
  });
  const [error, setErrors] = useState({
    name: "",
    email: "",
    phone_number: "",
    password: "",
    confirm_password: "",
  });
  const handleChange = ({ target: { name, value } }) => {
    setErrors((prev) => {
      return { ...prev, [name]: "" };
    });
    setInputs((prev) => ({ ...prev, [name]: value }));
```

```
};
const checkInputErrors = () => {
  let status = true;
  if (inputs.email.trim() === "" || !emailRegex.test(inputs.email.trim())) {
    setErrors((prev) => {
      return { ...prev, email: "Enter a valid email" };
    });
    status = false;
  }
  if (inputs.name.trim() === "") {
    setErrors((prev) => {
      return { ...prev, name: "Enter a valid name" };
    });
    status = false;
  }
  if (inputs.phone_number.trim() === "") {
    setErrors((prev) => {
      return { ...prev, phone_number: "Enter a valid phone number" };
    });
    status = false;
  }

  if (inputs.confirm_password.trim() === "") {
```

```
  setErrors((prev) => {
    return { ...prev, confirm_password: "Enter a valid  password" };
  });
  status = false;
}
if (inputs.password.trim() === "") {
  setErrors((prev) => {
    return { ...prev, password: "Enter a valid password" };
  });
  status = false;
}
if (inputs.password.trim().length < 6) {
  setErrors((prev) => {
    return { ...prev, password: "Minimum 6 characters" };
  });
  status = false;
}
if (inputs.password.trim() !== inputs.confirm_password.trim()) {
  setErrors((prev) => {
    return { ...prev, confirmPassword: "Password don't match" };
  });
  status = false;
}
```

```
  return status;
};

const handleSignUp = async () => {
  if (checkInputErrors()) {
    const data = await registerUser(inputs);
    if (data.error) {
      toast({
        title: data.error,
        status: "error",
        duration: 3000,
        isClosable: true,
        variant: "left-accent",
        position: "top",
      });
      return;
    }
    setUser(data);
    toast({
      title: `Your journey starts here ${data.name}`,
      status: "success",
      duration: 3000,
      isClosable: true,
```

```
      variant: "left-accent",
      position: "top",
    });
    localStorage.setItem("user", JSON.stringify(data));
    navigate("/dashboard");
  }
};

return (
  <>
    <div>
      <button className="bg-base-300 rounded-box flex flex-row justify-evenly items-center gap-10 px-10 py-5 w-fit mx-auto">
        <span>Sign in with Github</span>
        <img src={`github-dark.png`} alt="github" width="14%" />
      </button>
      <div className="divider max-w-xs">or</div>
      <form
        onSubmit={(e) => {
          e.preventDefault();
          handleSignUp();
        }}
        className="card bg-base-300 rounded-box flex flex-col justify-center items-center gap-3 px-10 py-5 w-fit mx-auto"
```

```
>
  <div>
    <input
      value={inputs.name}
      type="text"
      name="name"
      placeholder="name"
      className="input input-bordered input-primary w-full"
      onChange={handleChange}
    />
    {error.name !== "" && (
      <p className="text-sm text-red-500  font-medium">{error.name}</p>
    )}
  </div>
  <div>
    <input
      value={inputs.email}
      type="text"
      name="email"
      placeholder="email"
      className="input input-bordered input-primary w-full"
      onChange={handleChange}
    />
```

```
  {error.email !== "" && (
    <p className="text-sm text-red-500  font-medium">{error.email}</p>
  )}
</div>
<div>
  <input
    value={inputs.phone_number}
    type="number"
    name="phone_number"
    placeholder="phone number"
    className="input input-bordered input-primary w-full"
    onChange={handleChange}
  />
  {error. Phone_number !== "" && (
    <p className="text-sm text-red-500  font-medium">
      {error. Phone_number}
    </p>
  )}
</div>
<div>
  <input
    value={inputs.password}
    type="password"
```

```
    name="password"
    placeholder="password"
    className="input input-bordered input-primary w-full"
    onChange={handleChange}
   />
   {error.password !== "" && (
    <p className="text-sm text-red-500  font-medium">
     {error.password}
    </p>
   )}
  </div>
  <div>
   <input
    value={inputs.confirm_password}
    type="password"
    name="confirm_password"
    placeholder="confirm password"
    className="input input-bordered input-primary w-full"
    onChange={handleChange}
   />
   {error. Confirm_password !== "" && (
    <p className="text-sm text-red-500  font-medium">
     {error.confirm_password}
```

```
            </p>
          )}
        </div>
        <div className="text-center">
          <button
            onClick={handleSignUp}
            className="btn btn-sm btn-primary mb-4"
          >
            Sign Up
          </button>
        </div>
      </form>
    </div>
  </>
  );
};
export default SignUp;
```

Skill.jsx

```
import React, { useEffect, useState } from "react";
const Skill = ({ skill, setSelectedSkills, disabled }) => {
  const [isSelected, setIsSelected] = useState(false);
  useEffect(() => {
```

```
    if (isSelected) {
      setSelectedSkills((prev) => [...prev, skill]);
    } else {
      setSelectedSkills((prev) => prev.filter((item) => item !== skill));
    }
  }, [isSelected]);
  return (
    <li className="hover:text-white flex gap-1 items-center justify-between p-1 rounded-sm">
      {skill}
      <button
        disabled={disabled}
        onClick={() => setIsSelected(!isSelected)}
        className={`cursor-pointer border-2 ${
          !isSelected ? "border-green-500" : "border-red-400"
        } p-1 rounded-lg`}
      >
        {!isSelected ? "Add" : "Remove"}
      </button>
    </li>
  );
};
export default Skill;
```

BIBILOGRPHY:

1. **[Flask. *flask*] https://flask.palletsprojects.com/en/1.1.x/foreword/**
2. **[Flask install]**
   https://www.twilio.com/docs/usage/tutorials/how-to-set-up-your-python-and-flask-development-environment

3. **[My SQL database creation] https://remotemysql.com/**

4. **[Creating a Flask Project]**
   https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-2297

5. **[Creating REST API]**
   https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask

6. **[Designing the UI] https://spring.io/guides/tutorials/react-and-spring-data-rest/**

7. **[Flask and MYSQL Database Connection]**
   https://www.askpython.com/python-modules/flask/flask-mysql-database

8. **[Dockers Image creation] https://www.geeksforgeeks.org/dockerize-your-flask-app/**

9. **[Deploying APP to TAS]**
   https://drive.google.com/file/d/1Y7V9XBm1iCsC37Fjv2UO6fnyGPZJcE6r/view?usp=sharing

10. **[installing cli foundry] https://docs.cloudfoundry.org/cf-cli/install-go-cli.html**

DEMO LINK:

https://drive.google.com/file/d/1zM990FpAbAO2Hi6tyCjrontw_x9fRYUH/view?usp=drivesdk

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-8558-1658924028/tree/main/Project%20Development%20Phase