SKILL / JOB RECOMMENDER APPLICATION

IBM NALAIYATHIRAN (HX8001)

PROJECT REPORT

SUBMITTED BY

TEAM_ID:PNT2022TMID26502

KAMALESH PATHY VA	**211719106035**
HARSHAVARDHAN V	**211719106027**
KOVI RAHUL	**211719106037**
LOKESH N	**211719106043**
MAHESH P	**211719106044**

in

partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

RAJALAKSHMI INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"SKILL / JOB RECOMMENDER APPLICATION"** is the bonafide work of **"KAMALESH PATHY VA (211719106035), HARSHAVARDHAN V (211719106027), KOVI RAHUL (211719106037), LOKESH N (211719106043) and MAHESH P (211719106044)",** who carried out the project work under my supervision.

SIGNATURE:	**SIGNATURE:**
Dr. S.MANJULA, M.E., Ph.D.,	**Mr. A.BALAJI, M.E., (Ph.D).,**
HEAD OF THE DEPARTMENT,	**MENTOR,**
Dept. of Electronics and	Dept. of Electronics and
Communication Engg.,	Communication Engg.,
Rajalakshmi Institute of	Rajalakshmi Institute of
Technology,	Technology,
Kuthambakkam Post,	Kuthambakkam Post,
Chennai - 600 124	Chennai - 600 124

The viva-voce is held on ___________________.

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

TABLE OF FIGURES

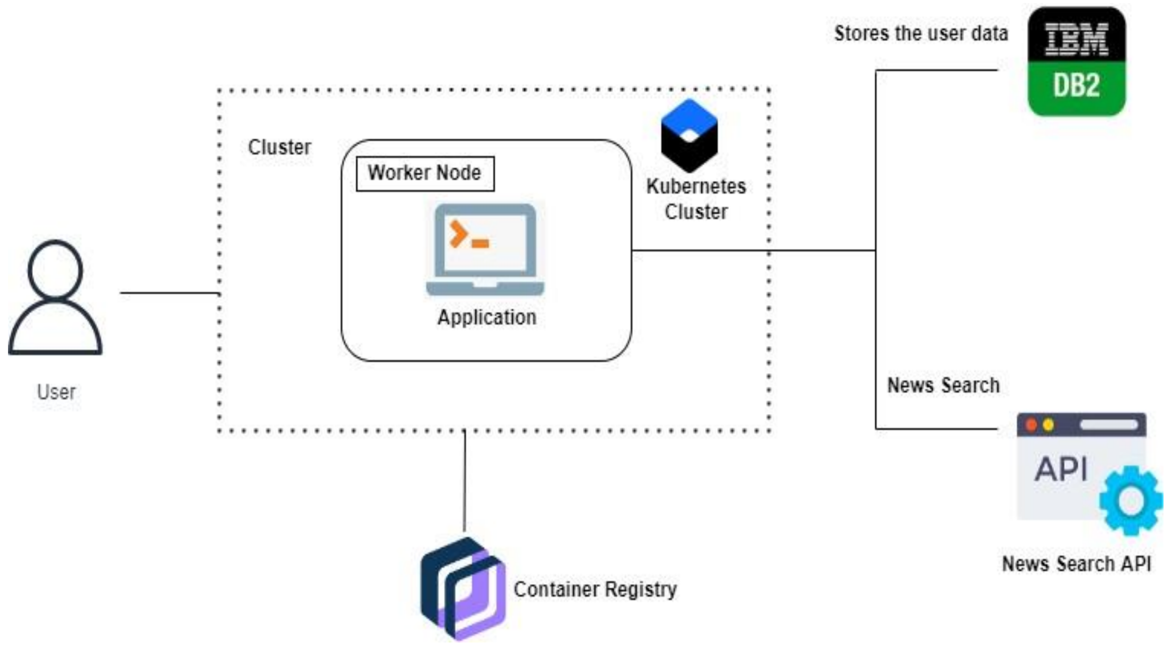
1. INTRODUCTION :

Having lots of skills but wondering which job will best suit you? Don't need to worry! We have come up with a skill recommender solution through which the fresher or the skilled person can log in and find the jobs by using the search option or they can directly interact with the chatbot and get their dream job.To develop an end-to-end web application capable of displaying the current job openings based on the user skillset. The user and their information are stored in the Database. An alert is sent when there is an opening based on the user skillset. Users will interact with the chatbot and can get the recommendations based on their skills. We can use a job search API to get the current job openings in the market which will fetch the data directly from the webpage.

1.1 Project Overview

Skill/Job Recommender Application

Technical Architecture:



Project Workflow:

- First the User login into the website
- User enters the details and the necessary information for the job.
- The details of user will be saved in db2.
- Then Organization posts the Job opening's table.
- User enter their skills.
- Based on the skills the page auto populates and recommend jobs.
- Therefore Jobs are recommended based on the Users Skills.

1.2 Purpose

First, job offers are collected from job search websites then they are prepared to extract meaningful attributes such as job titles and technical skills. Job offers with common features are grouped into clusters. As job seeker like one job belonging to a cluster, he will probably find other jobs in that cluster that he will like as well. A list of top n recommendations is suggested after matching data from job clusters and job seeker behavior, which consists on user interactions such as applications, likes and rating.

2. LITERATURE SURVEY

2.1 Existing problem

Recommendation engines are a big investment, not only financially, but in terms of time, too: it takes a long time and deep expertise to build an effective recommendation engine in-house. Alternatively, you could employ an off-the-shelf solution from a third-party company, but with so many options available on the market, how do you know which is the right one for your business? Evaluating different solutions can be enormously time consuming, as you need to evaluate their case studies, the technology, how the solution will be integrated into your current company setup, and so on. Bringing a recommendation engine into your business can be a complex affair. Sometimes, it might not be worth the effort, especially if it does not fit into your business vertical. Like all AI-based technologies, 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.

.

2.2 References

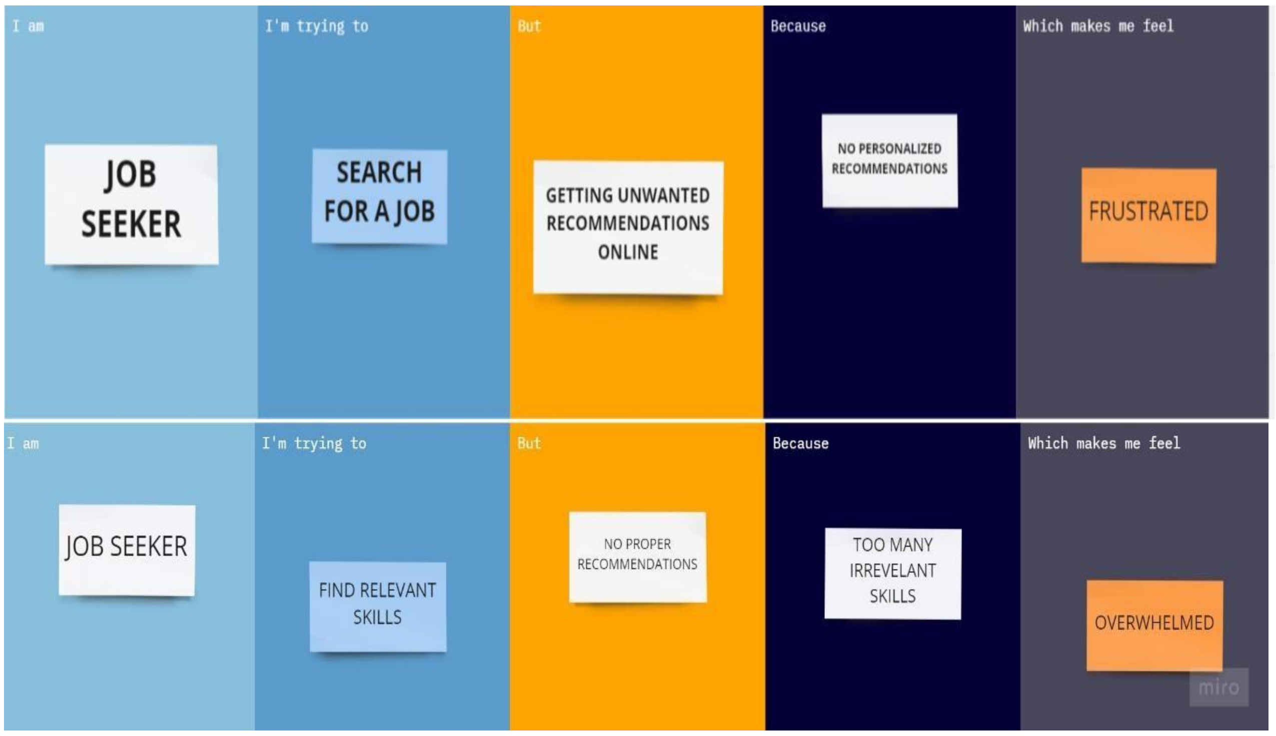
BOOK/ JOURNAL	**AUTHOR'S NAME**	**INFERENCE**
Job, Recommendation based on Job Seeker Skills: An Empirical Study, 2018.	Jorge Valverde Rebaza, Ricardo Puma,PaulBustios, Nathalia C. Silva.	Job search is a task commonly done on the Internet using job search engine sites like Linked In, Indeed, and others. Commonly, a job seeker has two ways to search a job using these sites: 1) doing a query based on keywords related to the job vacancy that he/she is looking for, or 2) creating and/or updating a professional profile containing data related to his/her education, professional experience, professional skills and other, and receive personalized job recommendations based on this data.
A survey of job recommender systems, 2012.	Shaha T. Al- Otaibiand Mourad Ykhlef	The fast growth of the Internet caused a matching growth of the amount of available online information that increased the need to expand the ability of users to manage all this information. This encourages a substantial interest in specific research fields and technologies that could benefit the managing of this information overload. The most important fields are Information retrieval and Information filtering. Information retrieval deals with automatically matching users information and Information filtering aims to assist users eliminating unwanted information

BOOK/ JOURNAL	AUTHOR’S NAME	INFERENCE
Skill-based Career Path Modeling and Recommendation	Aritra Ghosh, Beverly Woolf, Shlomo Zilberstein, Andrew Lan	Newskills andknowledge are needed for jobs in the future due in part to the rapid development of workplace technology such as artificial intelligence and internet of things. Jobs in the future will likely require skills that are not taught in schools nor in standard training programs. Instead, workers will have to either upskill as they move to new jobs within the same industry, or reskill themselves through the lifelong learning process to move to another industry.
Job Recommendation through Progression of Job Selection	AmberNigam, Aakash Roy, Hartaran Singh, Harsimran Waila	Through this paper, we are introducing a novel machine learning model which uses the candidates job preference over time to incorporate the dynamics associated with highly volatile job market. In addition to that, our approach comprises several other smaller recommendations that contribute to problems of 2 a)generating serendipitous recommendations b) solving the cold-start problem for new jobs and new candidates.
Job Recommendation System Using Machine Learning And Natural Language Processing	Jeevankrishna	In this paper scrape data from Job board to create offline Job dataset.develop a user profile based on stack overflow survey data.construct a recommender model that can address cold start issue. devise recommender model which recommends Job to the job seeker based on skills.

2.3 Problem Statement Definition

Mr. Mohan is a 24 years man, who is having lot of Skills, he is searching for an job and needs to get help with queries about what type of job to do, is it necessary to learn any other skills etc… Through Skill\Job Recommender Application.

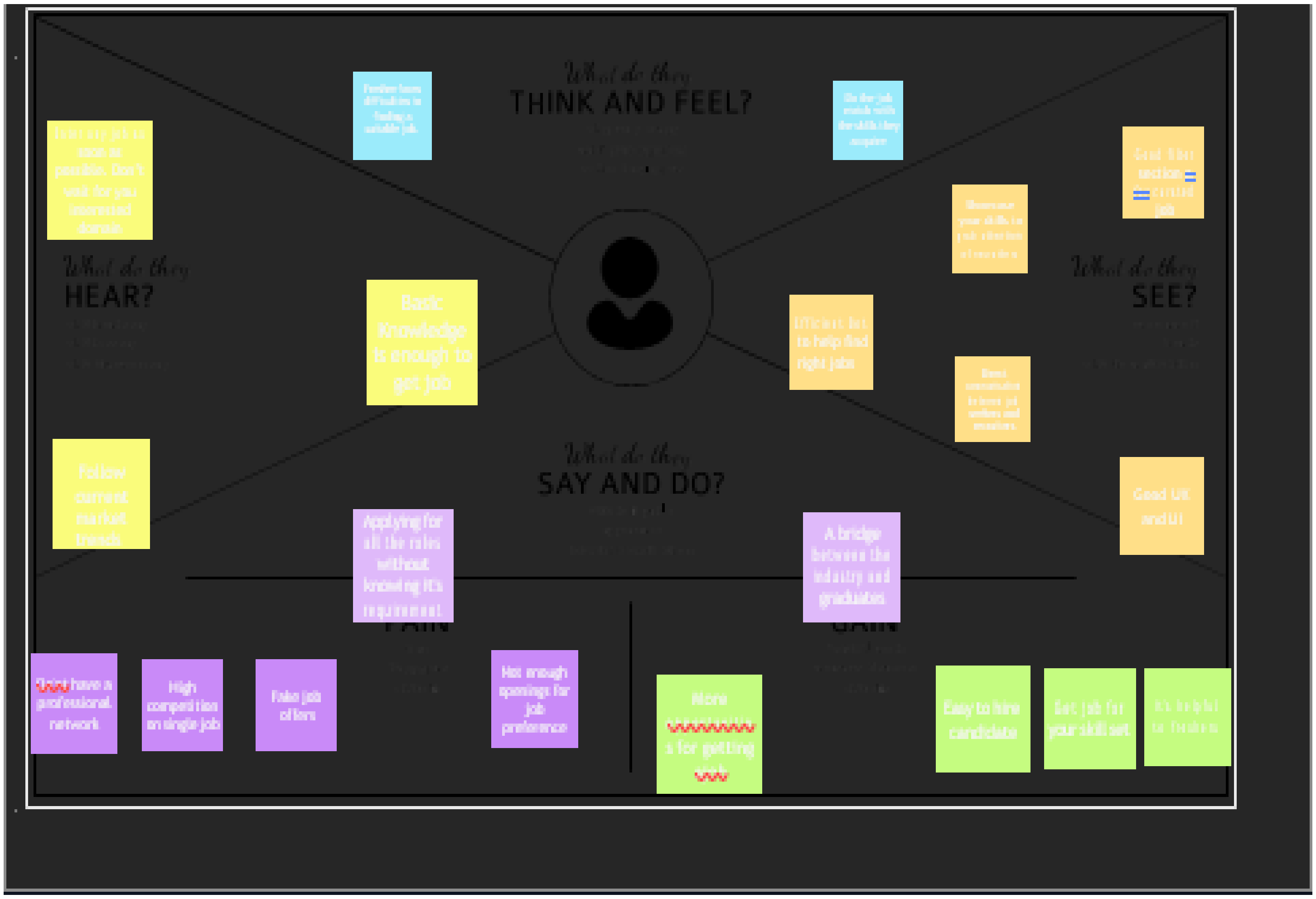
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	JOB SEEKER	SEARCH FOR A JOB	GETTING UNWANTED RECOMMENDATIONS ONLINE	NO PERSONALIZED RECOMMENDATIONS	FRUSTRATED
PS-2	JOB SEEKER	FIND RELEVANT SKILLS	NO PROPER RECOMMENDATIONS	TOO MANY IRREVELANT SKILLS	OVERWHELMED



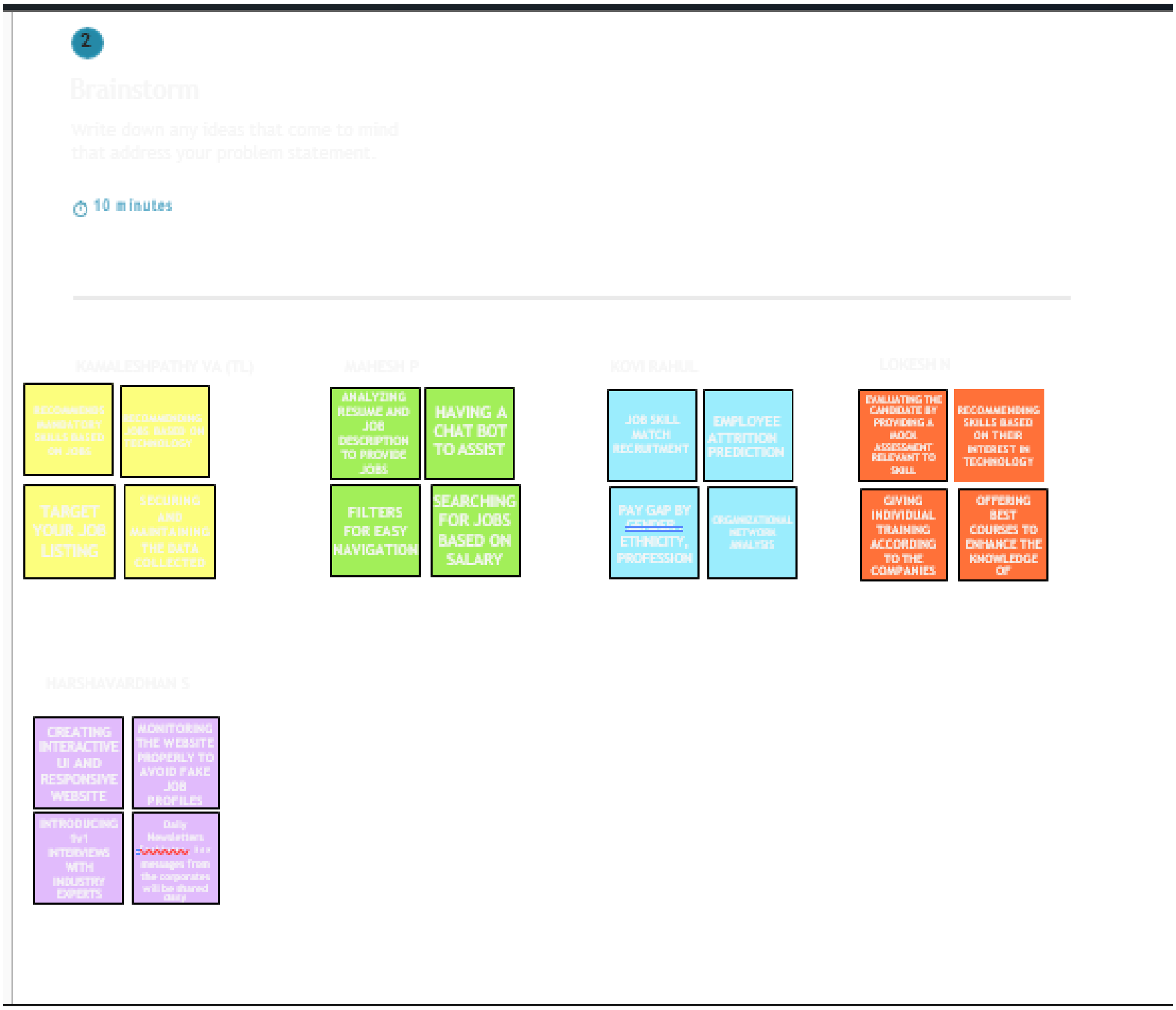
I am	I'm trying to	But	Because	Which makes me feel
JOB SEEKER	SEARCH FOR A JOB	GETTING UNWANTED RECOMMENDATIONS ONLINE	NO PERSONALIZED RECOMMENDATIONS	FRUSTRATED
JOB SEEKER	FIND RELEVANT SKILLS	NO PROPER RECOMMENDATIONS	TOO MANY IRREVELANT SKILLS	OVERWHELMED

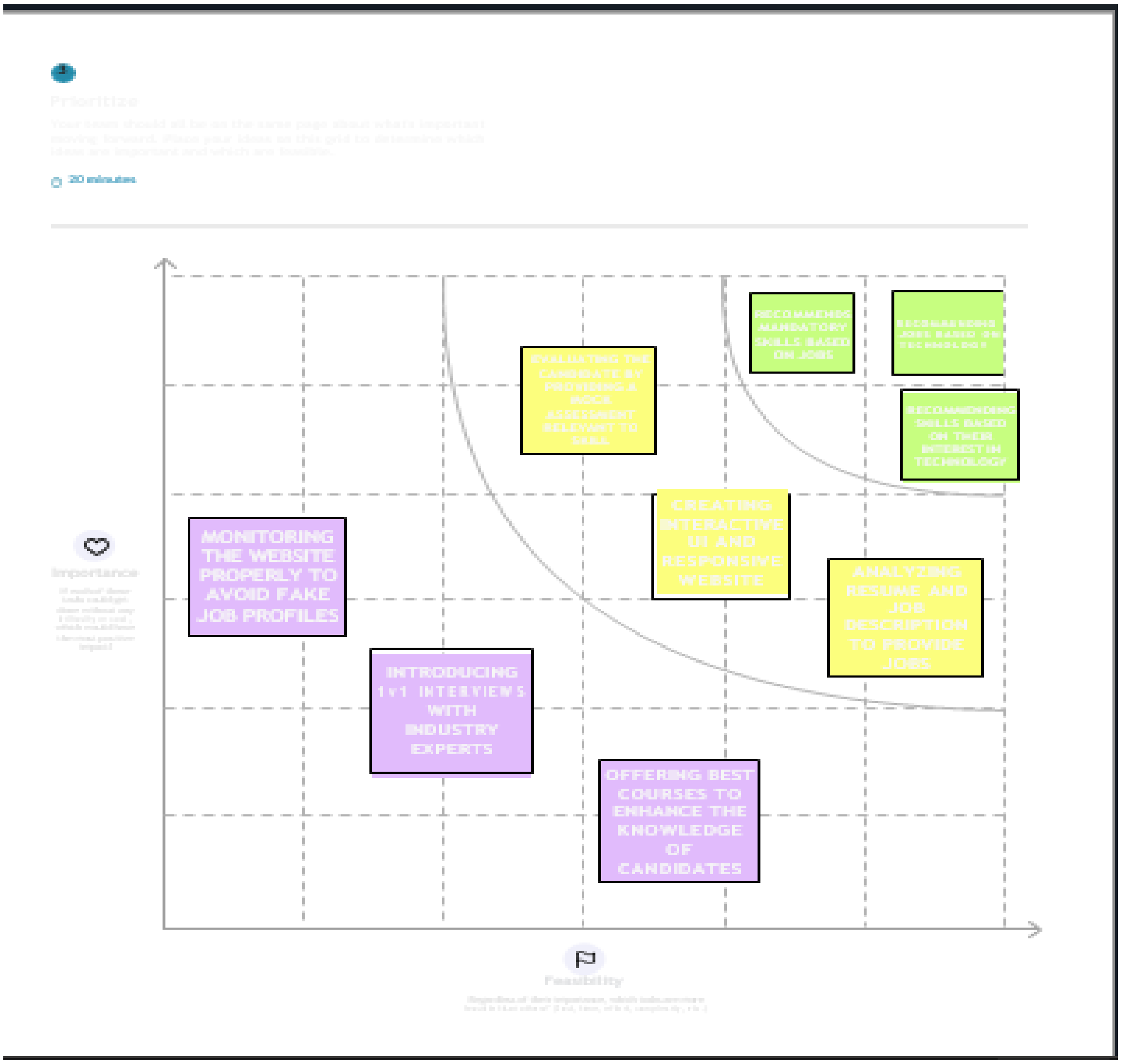
3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas


What do they
THINK AND FEEL?
What do they
HEAR?
What do they
SEE?
What do they
SAY AND DO?
PAIN
GAIN
Basic Knowledge is enough to get job
Follow current market trends
Applying for all the roles without knowing it's requirement
A bridge between the industry and graduates
Efficient list to help find right jobs
Good UX and UI
Don't have a professional network
High competition on single job
Fake job offers
Not enough openings for job preference
More opportunities for getting work
Easy to hire candidate
Get job for your skill set
It's helpful to freshers

3.2 Ideation & Brainstorming


2
Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
KAMALESHPATHY VA (TL)
RECOMMENDS MANDATORY SKILLS BASED ON JOBS
RECOMMENDING JOBS BASED ON TECHNOLOGY
TARGET YOUR JOB LISTING
SECURING AND MAINTAINING THE DATA COLLECTED
MAHESH P
ANALYZING RESUME AND JOB DESCRIPTION TO PROVIDE JOBS
HAVING A CHAT BOT TO ASSIST
FILTERS FOR EASY NAVIGATION
SEARCHING FOR JOBS BASED ON SALARY
KOVI RAHUL
JOB SKILL MATCH RECRUITMENT
EMPLOYEE ATTRITION PREDICTION
PAY GAP BY GENDER, ETHNICITY, PROFESSION
ORGANIZATIONAL NETWORK ANALYSIS
LOKESH N
EVALUATING THE CANDIDATE BY PROVIDING A MOCK ASSESSMENT RELEVANT TO SKILL
RECOMMENDING SKILLS BASED ON THEIR INTEREST IN TECHNOLOGY
GIVING INDIVIDUAL TRAINING ACCORDING TO THE COMPANIES
OFFERING BEST COURSES TO ENHANCE THE KNOWLEDGE OF
HARSHAVARDHAN S
CREATING INTERACTIVE UI AND RESPONSIVE WEBSITE
MONITORING THE WEBSITE PROPERLY TO AVOID FAKE JOB PROFILES
INTRODUCING 1v1 INTERVIEWS WITH INDUSTRY EXPERTS
Daily Newsletters [illegible] messages from the corporates will be shared daily



Prioritize

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

20 minutes

Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible in terms of cost, time, effort, complexity, etc.?
RECOMMENDING MANDATORY SKILLS BASED ON JOBS
RECOMMENDING JOBS BASED ON TECHNOLOGIES
RECOMMENDING SKILLS BASED ON THEIR INTEREST IN TECHNOLOGY
EVALUATING THE CANDIDATE BY PROVIDING A MOCK ASSESSMENT RELEVANT TO SKILL
CREATING INTERACTIVE UI AND RESPONSIVE WEBSITE
ANALYZING RESUME AND JOB DESCRIPTION TO PROVIDE JOBS
MONITORING THE WEBSITE PROPERLY TO AVOID FAKE JOB PROFILES
INTRODUCING 1v1 INTERVIEWS WITH INDUSTRY EXPERTS
OFFERING BEST COURSES TO ENHANCE THE KNOWLEDGE OF CANDIDATES

3.3Proposed Solution

ABSTRACT

In the last years, job recommender systems have become popular since they successfully reduce information overload by generating personalized job suggestions. Although in the literature exists a variety of techniques and strategies used as part of job recommender systems, most of them fail to recommending job vacancies that fit properly to the job seekers profiles. Thus, the contributions of this work are threefold, we: i) made publicly available a new dataset formed by a set of job seekers profiles and a set of job vacancies collected from different job search engine sites; ii) put forward the proposal of a framework for job recommendation based on professional skills of job seekers; and iii) carried out an evaluation to quantify empirically the recommendation abilities of two state-of-the-art methods, considering different configurations, within the proposed framework. We thus present a general panorama of job recommendation task aiming to facilitate research and real-world application design regarding this important issue..

PROPOSED SYSTEM

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	● People are having the skill but they are unable to get the desired job for skills they have. ● People may want dream job but may not have the required skills. ● The main motive is to develop an end-to-end web application capable of displaying the current job openings based on the user skillset.
2.	Idea / Solution description	● We are proposing an application which will help the people to get suggestion on the jobs based on their skills. People can also enquire about skills that are required to their desired job. ● The user and their information are stored in the Database. An alert is sent when there is an opening based on the user skillset. 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.
3.	Novelty / Uniqueness	● Users will interact with the chatbot and gets the recommendations based on their skills. ● HR’s can get best candidates as per their requirements.

4.	Social Impact / Customer Satisfaction	● People will be benefited by knowing which jobs suits them based on their skill set. ● People will learn new skills required for their desired 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 best profiles.
6.	Scalability of the Solution	● IBM Cloud is used to make our web app continue to function well when it is changed in size or volume in order to meet a user need.

3.4 Problem Solution fit

Project Title: Skill/Job recommender Application

Project Design Phase-I – Problem Solution Fit

Team ID: PNT2022TMID26502

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

The main customers for our project are :

- Persons who are seeking employment
- Persons that recruit job candidates

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

- Concern about misuse of personal information
- Worry about unreliable connections
- Inadequate product knowledge
- Potential Scam
- Time consuming

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Pros	Cons
Promotion of people's skillset	Delivering false information
Marketing of company infrastructure	Occurrence of fraudulent activity
Cultivate commercial relationship	Intense competition

Explore AS, differentiate

Focus on J&P, tap into

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

- Create a platform to facilitate job searching
- A platform to make it simpler to identify people with the necessary skills
- Make the job-filtering process simpler
- Profile with safe personal data

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists? What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

- Jobs that are listed on unreliable platforms may be fraudulent
- Companies fail to disclose their true infrastructure
- Some job portals want payment in advance of the job starting.
- Users post false credentials
- Users pretend to have expertise in a skillset they lack

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

- When Users apply for fraudulent jobs, they get unhappy due to wasted time

  Users were not satisfied when platforms allowed hirers to post jobs that were not real
- Cheating during online recruitment process
- When candidates with inadequate qualifications apply for a position, employers become irritated.

Focus on J&P, tap int C

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

- Employment opportunities
- Endorsement and connections

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.

To develop an end-to-end web application which in default have a lot of current job openings through job search API out of which appropriate job will be recommended based on user skill set. At the same time students can develop their skills side by side with various courses and webinars offered by reputed organization. In addition to this a smart chat bot will be available for 24*7 which can help users in finding the right job.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

- Apply for jobs
- Review job applications
- Attend initial level assessment

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

- Final level interview
- Checkout location and infrastructure of company
- Finalize paperwork

Extract online & offline CH of BE

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	**User Registration (mobile users)**	Registration through Web application, mobile number Registration through Gmail
FR-2	**User accessibility**	The users need to enable location, storage, media allowance
FR-3	**User profile**	The users have to create a profile which has some basic information relevant to the application
FR-4	**User uploads**	The users have to upload the softcopy of their mark sheet, identity card and resume of their original database.
FR-5	**User verification**	The user has to verify whether the given information are correct or not.
FR-6	**End user benefits**	This makes the recruit process in an easy manner. It helps us to know the educational information in an effective way.

4.2 Non-Functional requirements

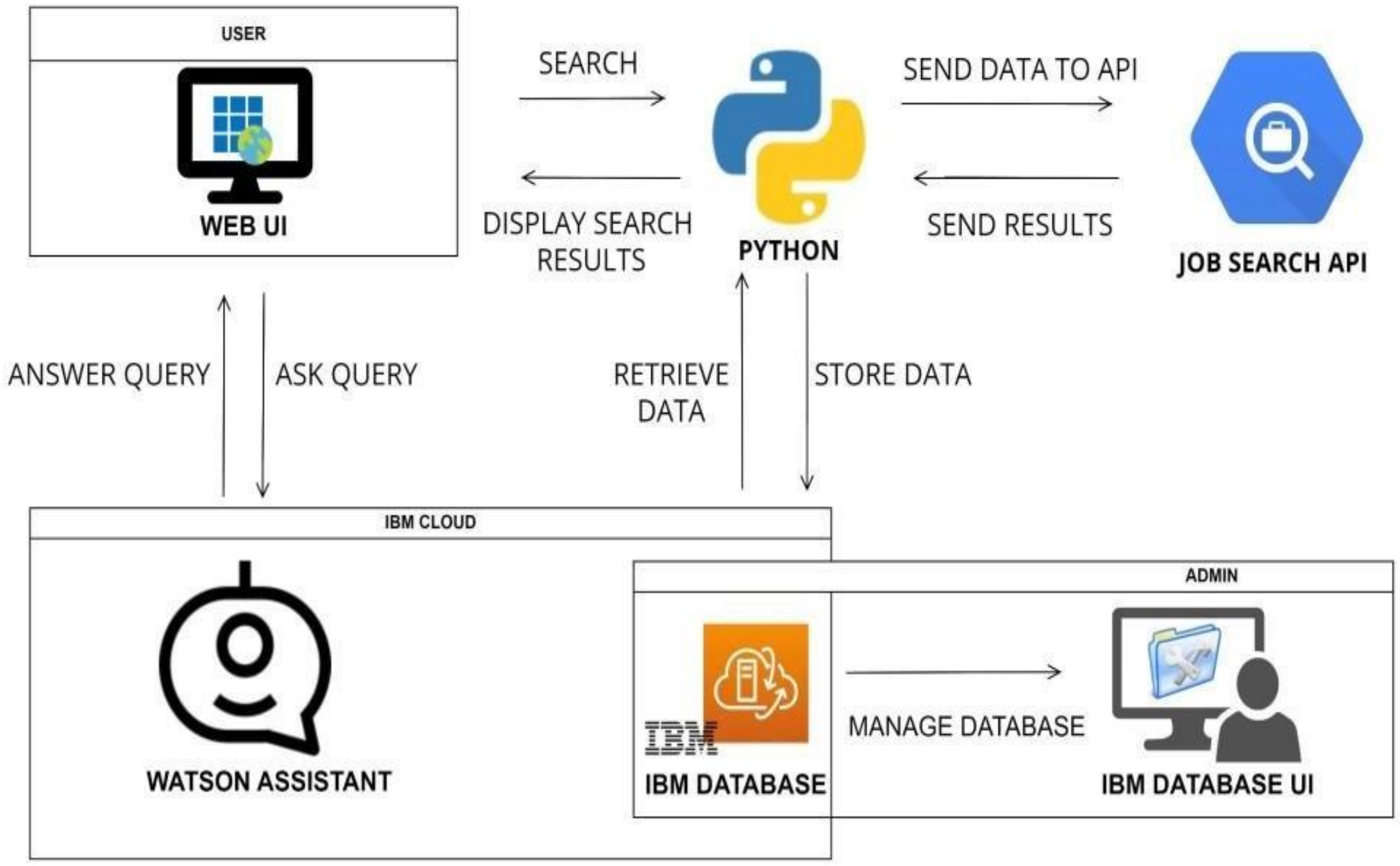
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It is effective way to achieve the skill and job recommendation. It is easily access by everyone.
NFR-2	**Security**	The privacy of the users should be guaranteed in thesystem.
NFR-3	**Reliability**	Integrity and consistency of the recommender engine andall its transactions should be ensured.
NFR-4	**Performance**	The recommender engine should generate recommendation within a time frame of 500 milliseconds.
NFR-5	**Availability**	It is always available in all platforms through websites.
NFR-6	**Scalability**	It may convenient for the user to use the application andalso this app have been considered as user friendly.
NFR-7	**learnability**	A new user should be able to use the recommender engine without putting too much efforts on learning how to use it, and in case of doubt, there must be some helpto solve their doubts.

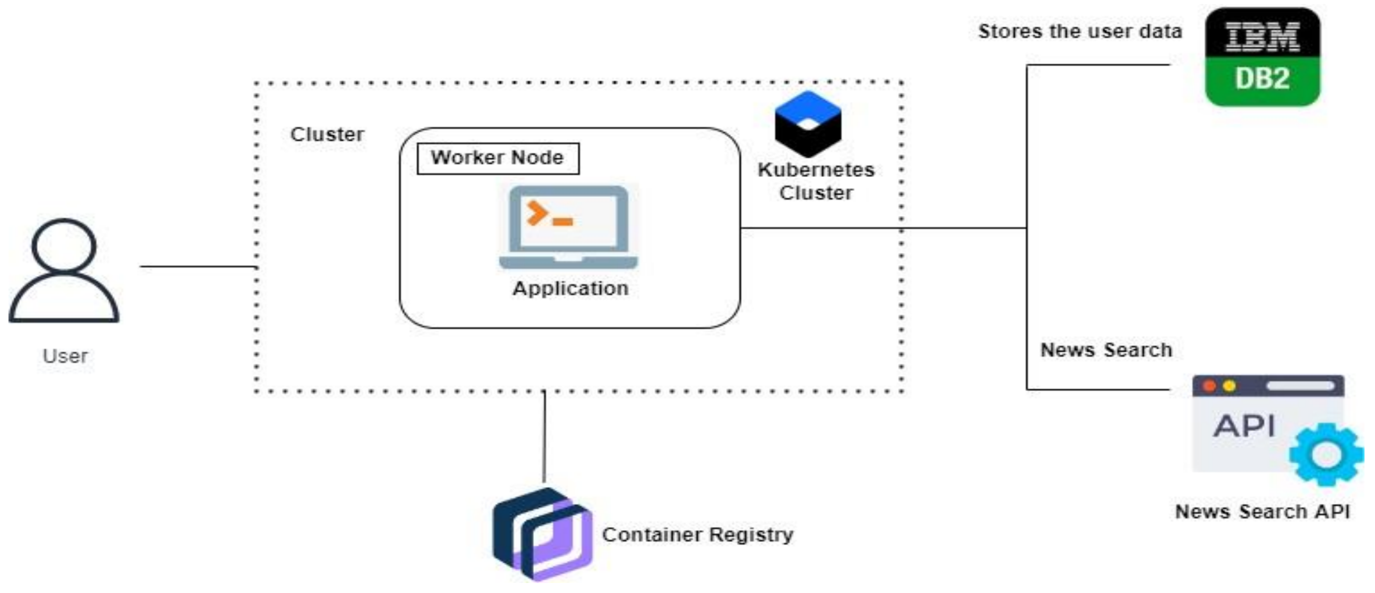
5. PROJECT DESIGN

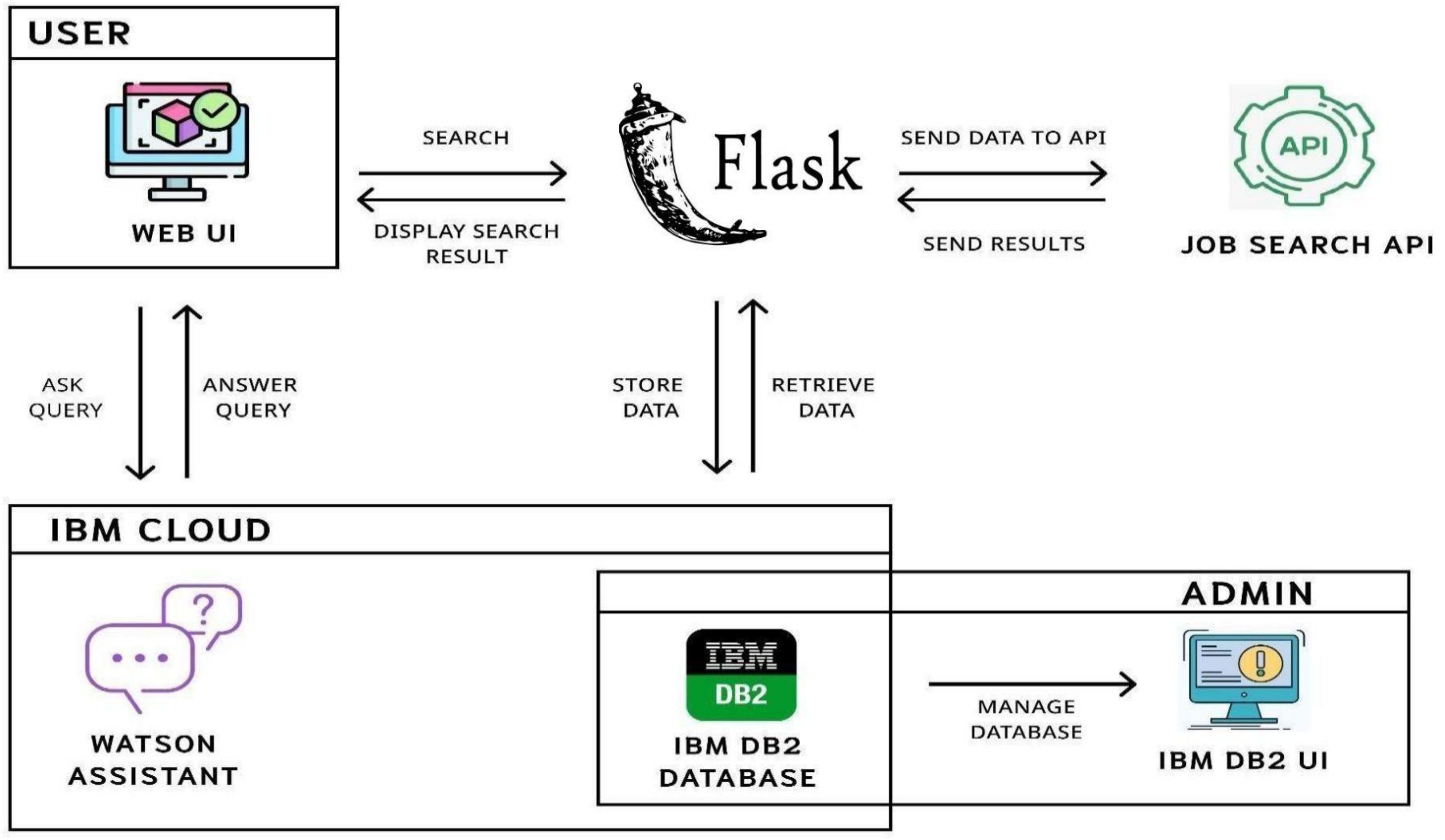
5.1 Data Flow Diagrams

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



5.2 Solution & Technical Architecture





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
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm
		USN-3	As a user, I can register for the application through online websites	I can register & access the dashboard with online website Login
		USN-4	As a user, I can register for the application through Gmail	I can receive confirmation Gmail & click confirm
	Login	USN-5	As a user, I can log into the application by entering email & password	I can receive confirmation email & click confirm
	Dashboard			
Customer (Web user)		USN-6	As a user, I can able to take up the skill assessment and view the appropriate test score. Based on the skill sets I can able to get personalised job recommendations.	I can receive job recommendations
Customer Care Executive		USN-7	As a customer care executive, we provide 24/7 chatbot support.	24/7 chatbot support
Administrator		USN-8	As an administrator, I can able to view the progress and make required changes in the project	Deploy user specific and personalised job recommendations

		USN-2	As a user, I will receive confirmation email once I have registered for the application through Gmail.	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Gmail.		Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password.		High	Sprint-1
	Dashboard	USN-5	As a user, I can enter the interests & choices of news I want to see for the first time in dashboard.		High	Sprint-2
		USN-6	As a user I can go through the feed of news filtered according to my wish.		High	Sprint-3
		USN-7	As a user, I can logout my account in settings.	I can click confirm to log out and end the session	Medium	Sprint-3

	Settings	USN-8	As a user, I can update my interests and choices in account settings.		Medium	Sprint-4
Customer Care Executive	Chat Bot / Query Section	USN-9	Solve issues brought up by client.		Medium	Sprint-4
Admin		USN10	Roll out updates and bug fixes.		High	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

. Title	Description	Date
Literature Survey and Information Gathering	Gathering Information by referring the technical papers, research publications etc	2 SEPTEMBER 2022
Empathy Map	To capture user pain and gains. Prepare List of ProblemStatement	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	19 SEPTEMBER 2022
Solution Architecture	Solution Architecture	19 SEPTEMBER 2022
Problem Solution Fit	Solution fit document	1 OCTOBER 2022
Technology Architecture	Technology Architecture diagram	3 OCTOBER 2022
Customer Journey	To Understand User Interactions and experiences with application	8 OCTOBER 2022
Functional Requirement	Prepare functional Requirement	12 OCTOBER 2022
Data flow diagram	Data flow diagram	15 OCTOBER 2022
Milestone & sprint delivery plan	Activity what we done &further plans	31 OCTOBER 2022

Project Development Delivery of sprint 1,2,3 & 4	Develop and submit the developed code by testing it	28 OCTOBER 2022 – 19 NOVEMBER 2022

6.2 Sprint Delivery 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	Medium	KAMALESH MAHESH RAHUL
Sprint-1	Database Connectivity	USN-2	Viewing and applying jobs Connecting UI with Database	10	High	HARSHA LOKESH
Sprint-2	SendGrid Integration	USN-3	SendGrid Integration with Python Code	10	Low	RAHUL HARSHA LOKESH
Sprint-2	Chatbot Development	USN-4	Building a chatbot	10	High	KAMALESH MAHESH
Sprint-3	Integration and Containerisation	USN-5	Integrating chatbot to the HTML page and containerizing the app.	20	Medium	LOKESH HARSHA KAMALESH
Sprint-4	Upload Image and deployment	USN-6	Upload the image to the IBM Registry and deploy it in the Kubernetes Cluster.	20	High	MAHESH RAHUL

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

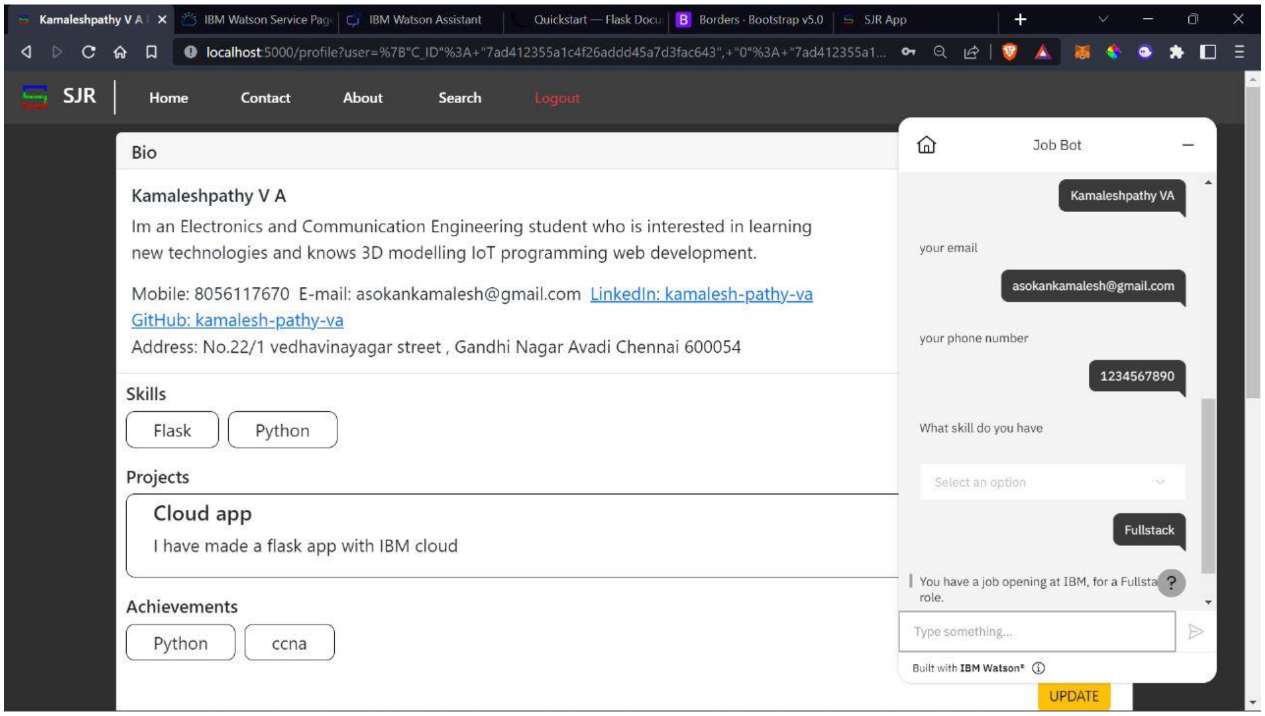
7.1 Feature:

IBM Watson

```
<script>
    window.watsonAssistantChatOptions = {
      integrationID: "c74dff4b-0b3e-405a-ae10-9a0d71f50fc9", // The ID of this integration.
      region: "jp-tok", // The region your integration is hosted in.
      serviceInstanceID: "0feb80be-a66e-4dc4-86ab-b96a3fa9bb30", // 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>
```

IBM Object storage

```
<img src="https://sjr-app.s3.jp-tok.cloud-object-storage.appdomain.cloud/sjr_symbol.png" alt="sjr_logo" width="50">
```



Bio

Kamaleshpathy V A

Im an Electronics and Communication Engineering student who is interested in learning new technologies and knows 3D modelling IoT programming web development.

Mobile: 8056117670 E-mail: asokankamalesh@gmail.com LinkedIn: kamalesh-pathy-va
GitHub: kamalesh-pathy-va
Address: No.22/1 vedhavinayagar street , Gandhi Nagar Avadi Chennai 600054

Skills

Flask Python

Projects

Cloud app

I have made a flask app with IBM cloud

Achievements

Python ccna

Job Bot

Kamaleshpathy VA

your email

asokankamalesh@gmail.com

your phone number

1234567890

What skill do you have

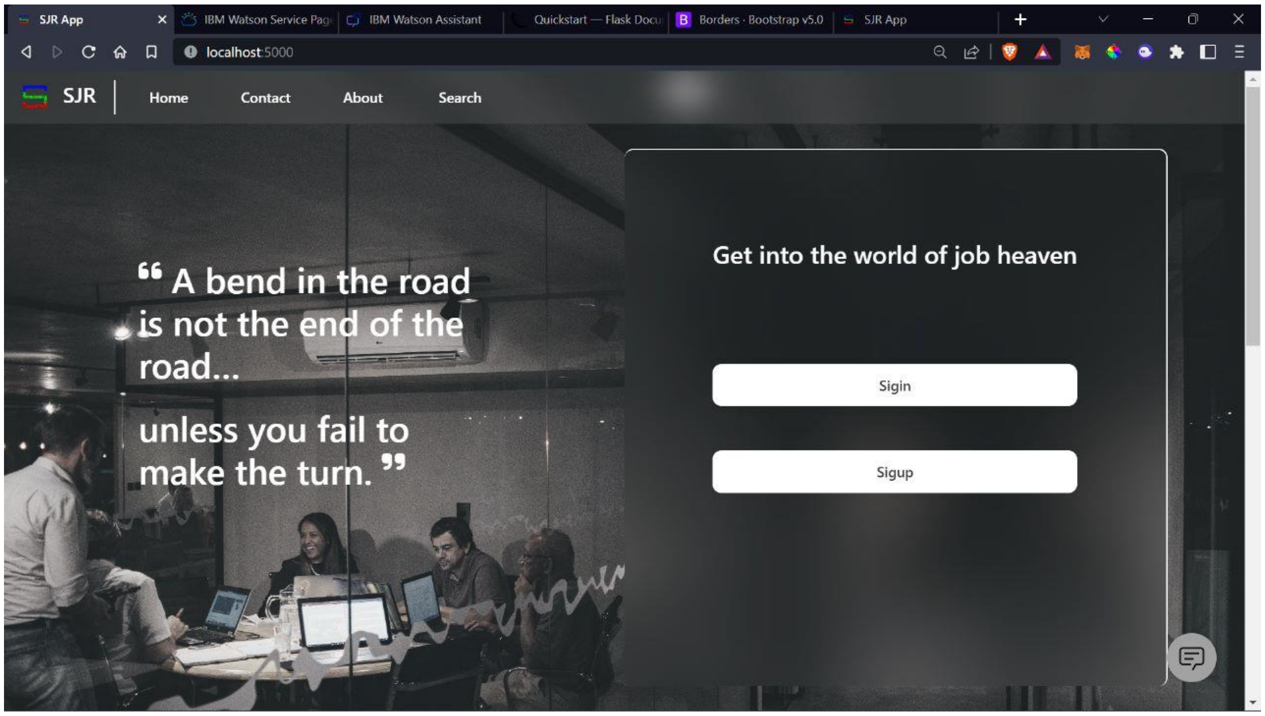
Select an option

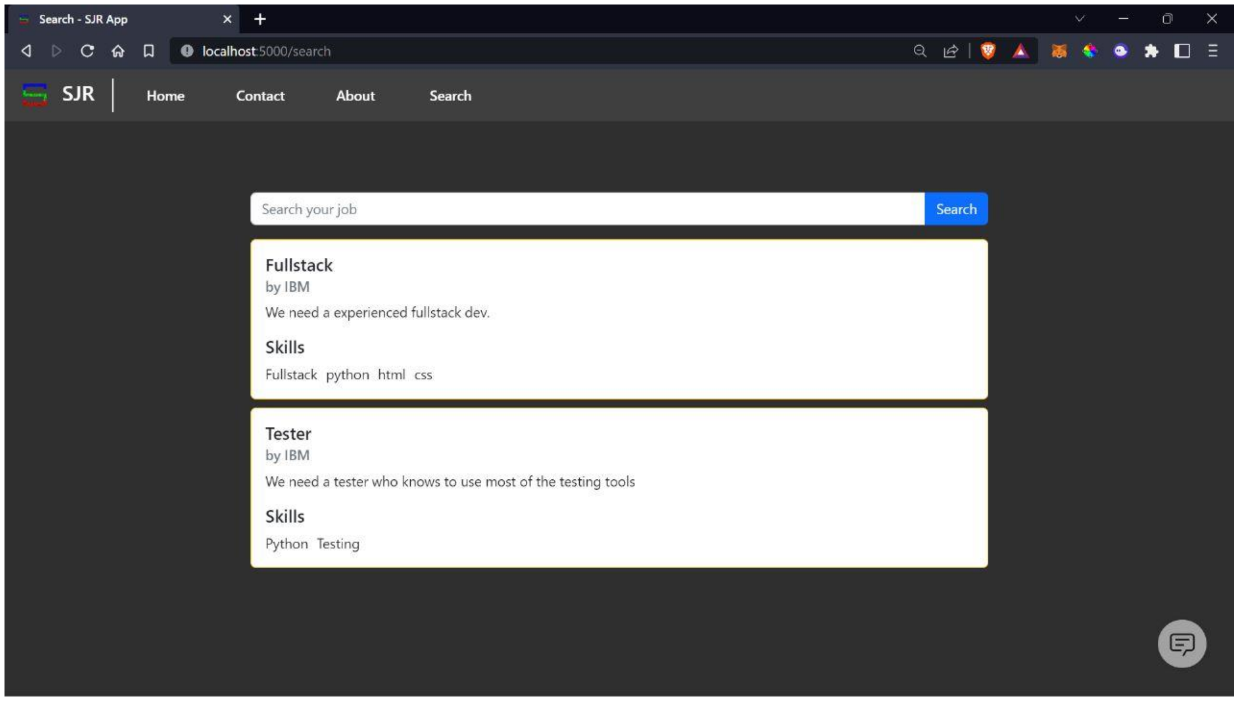
Fullstack

You have a job opening at IBM, for a Fullsta role.

8. TESTING

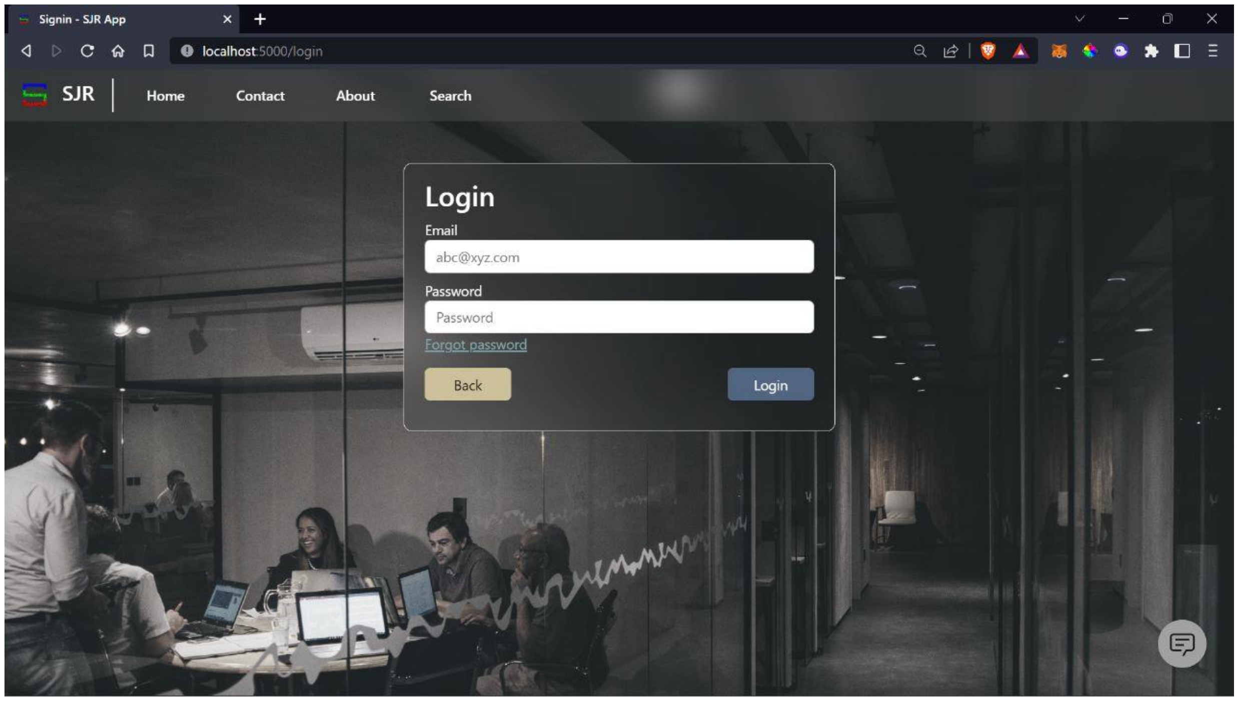
8.1 User Acceptance Testing

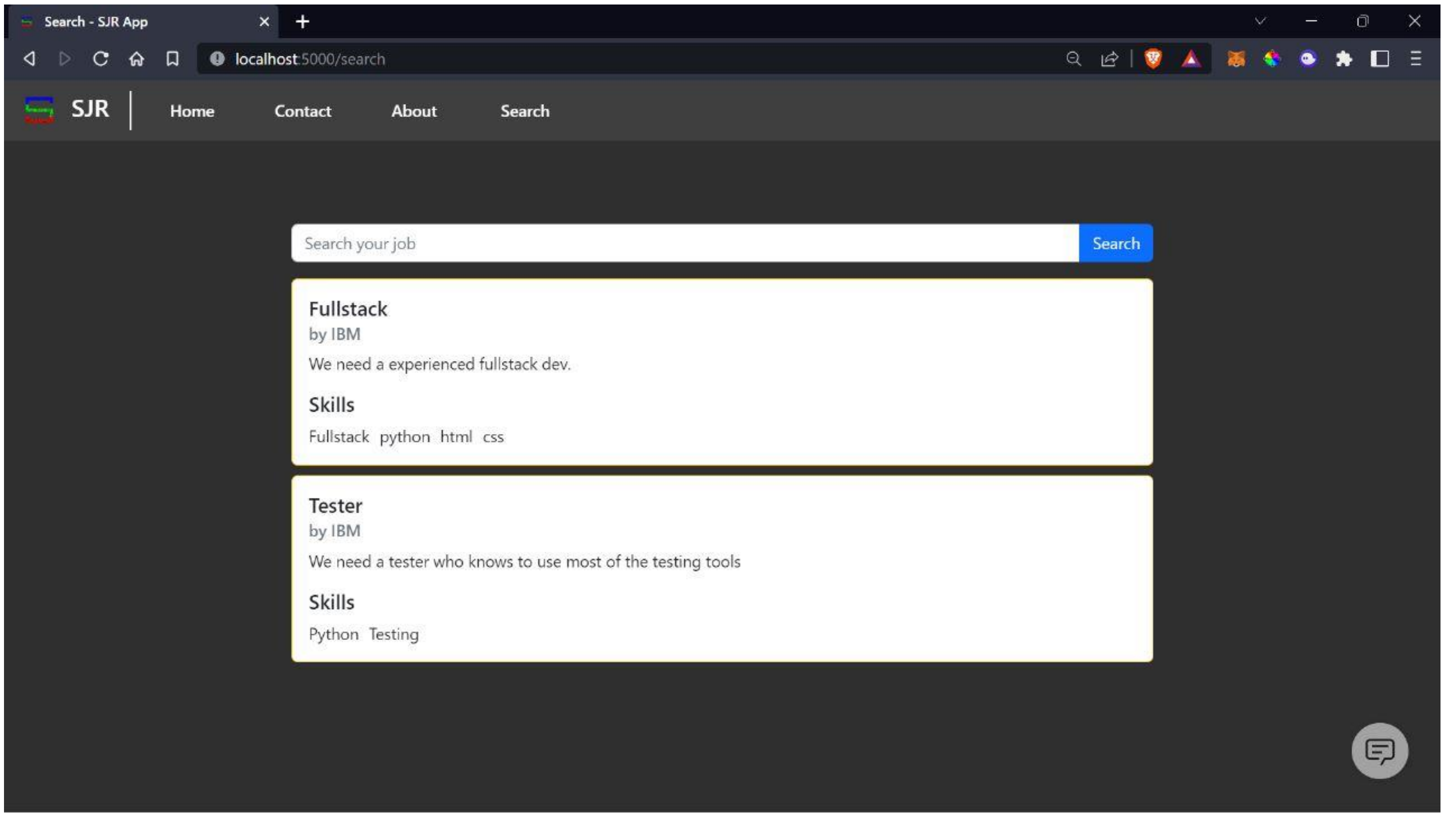




Fullstack
by IBM
We need a experienced fullstack dev.
Skills
Fullstack python html css

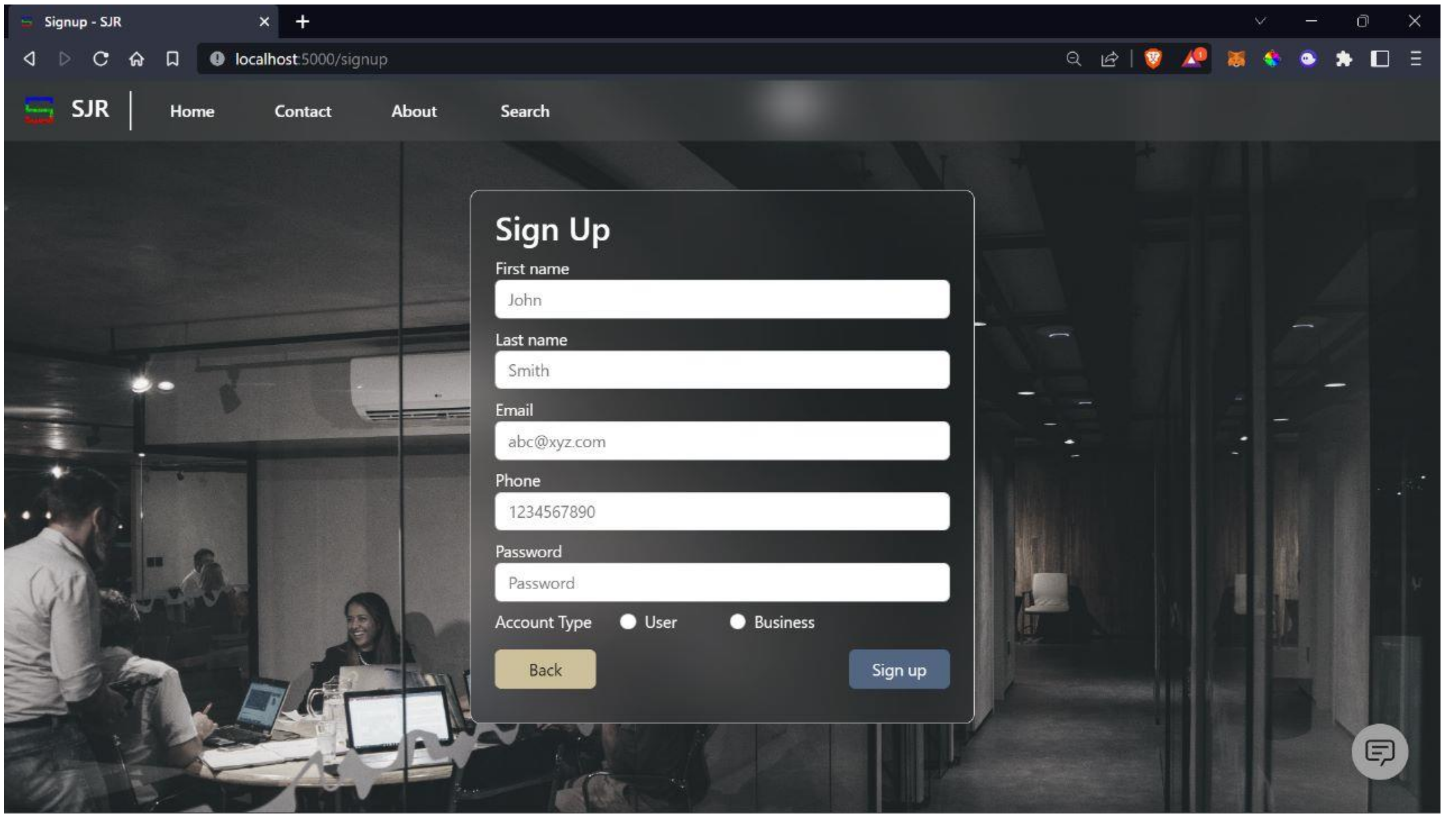
Tester
by IBM
We need a tester who knows to use most of the testing tools
Skills
Python Testing





Fullstack
by IBM
We need a experienced fullstack dev.
Skills
Fullstack python html css

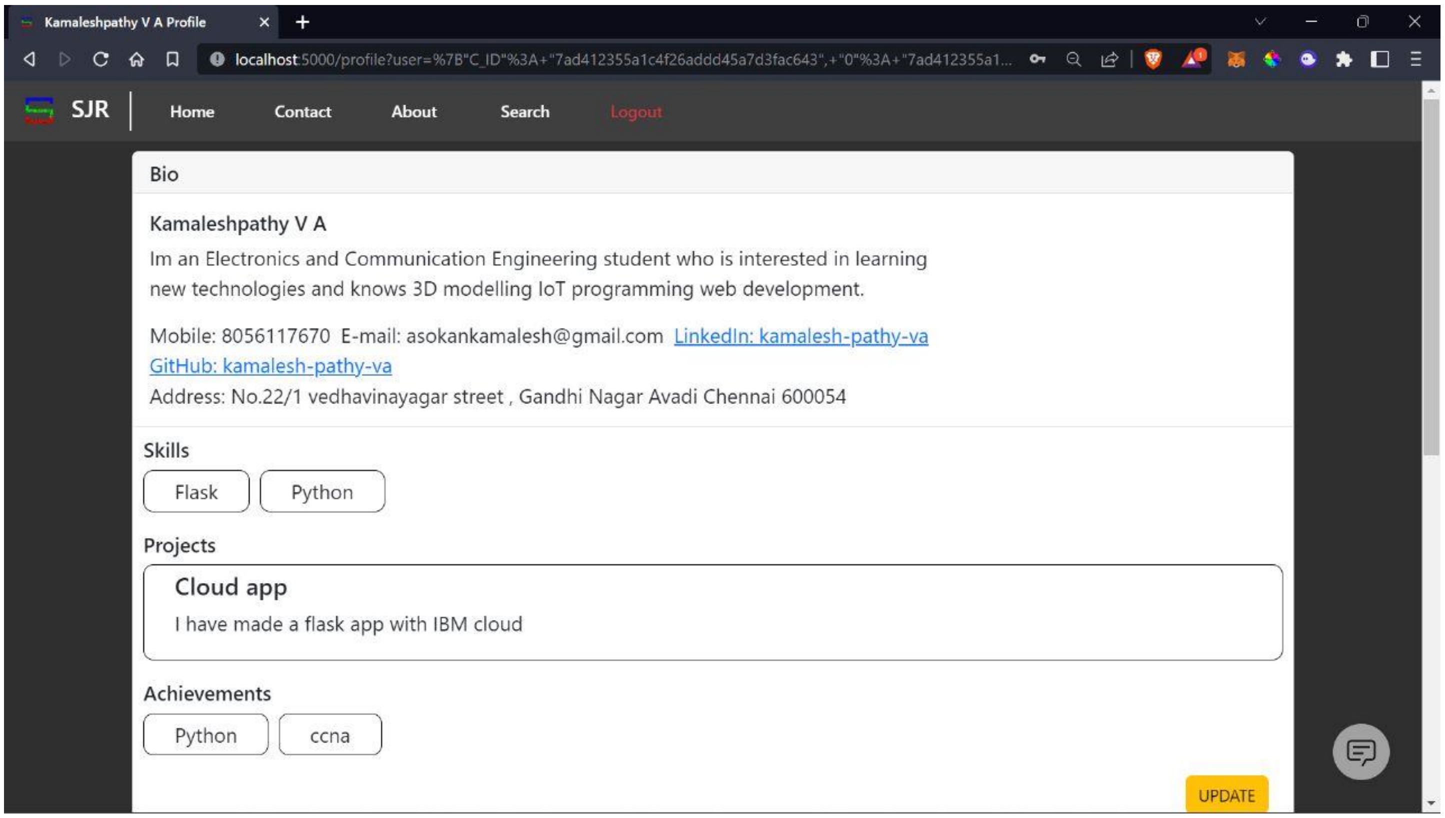
Tester
by IBM
We need a tester who knows to use most of the testing tools
Skills
Python Testing



Sign Up

First name: John
Last name: Smith
Email: abc@xyz.com
Phone: 1234567890
Password: Password
Account Type ○ User ○ Business

User:


Kamaleshpathy V A Profile
localhost:5000/profile?user=%7B"C_ID"%3A+"7ad412355a1c4f26addd45a7d3fac643",+"0"%3A+"7ad412355a1...
SJR
Home
Contact
About
Search
Logout

Bio

Kamaleshpathy V A

Im an Electronics and Communication Engineering student who is interested in learning new technologies and knows 3D modelling IoT programming web development.

Mobile: 8056117670 E-mail: asokankamalesh@gmail.com LinkedIn: kamalesh-pathy-va
GitHub: kamalesh-pathy-va
Address: No.22/1 vedhavinayagar street , Gandhi Nagar Avadi Chennai 600054

Skills

Flask Python

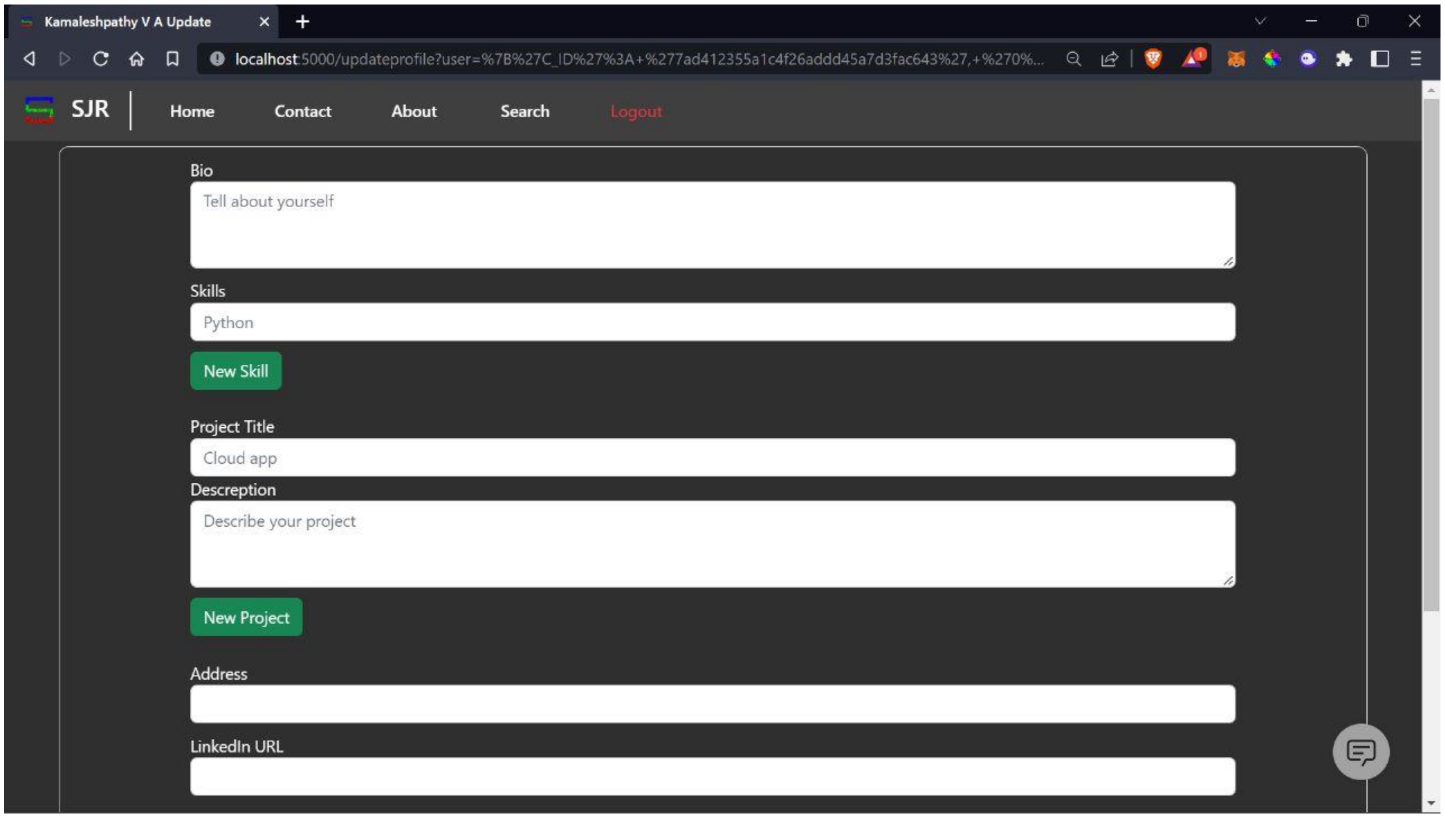
Projects

Cloud app

I have made a flask app with IBM cloud

Achievements

Python ccna


Kamaleshpathy V A Update
localhost:5000/updateprofile?user=%7B%27C_ID%27%3A+%277ad412355a1c4f26addd45a7d3fac643%27,+%270%...
SJR
Home
Contact
About
Search
Logout

Bio: Tell about yourself

Skills: Python

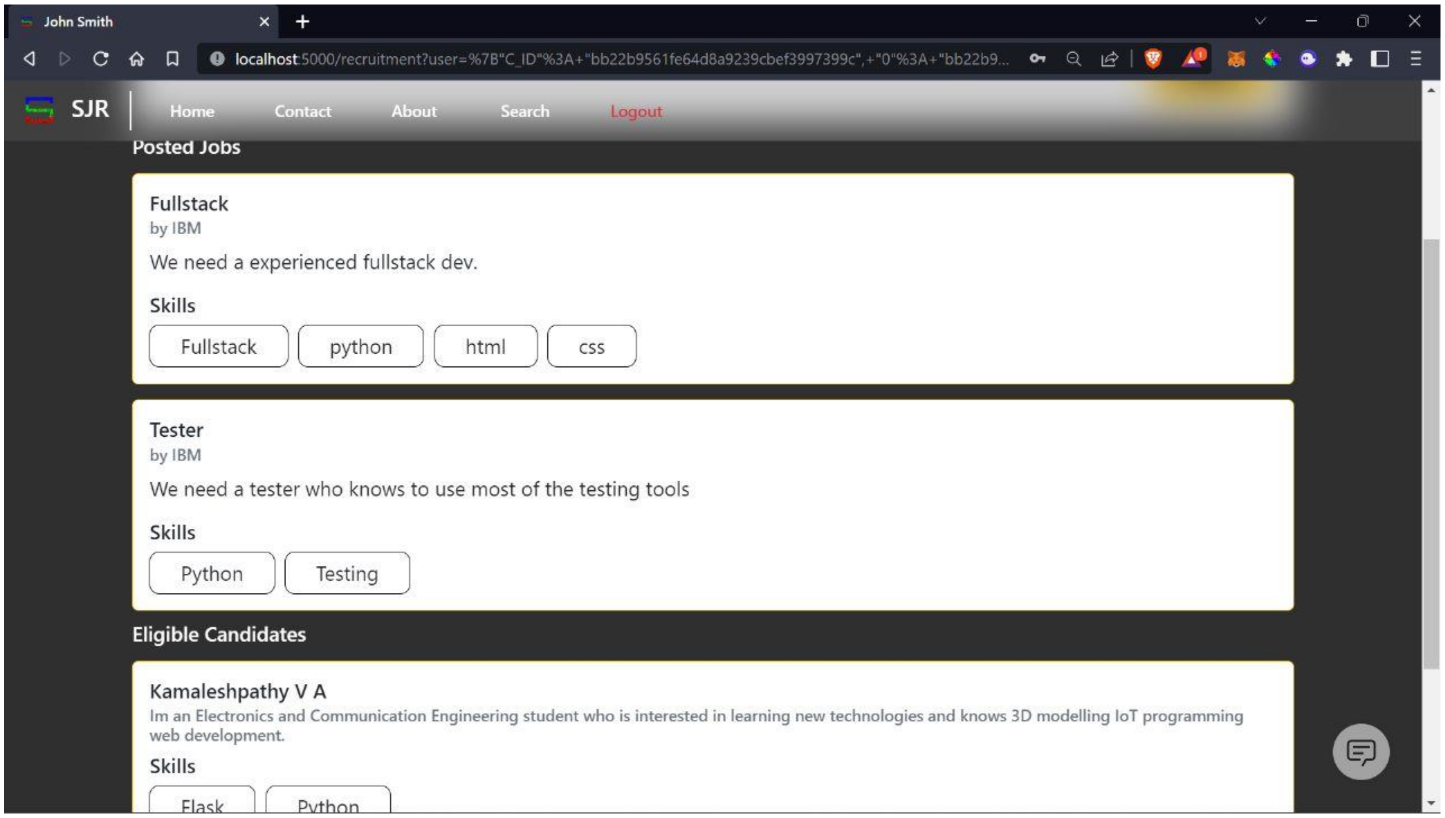
Project Title: Cloud app

Descreption: Describe your project

Address:

LinkedIn URL:

Recruiters:


John Smith

localhost:5000/recruitment?user=%7B"C_ID"%3A+"bb22b9561fe64d8a9239cbef3997399c",+"0"%3A+"bb22b9...

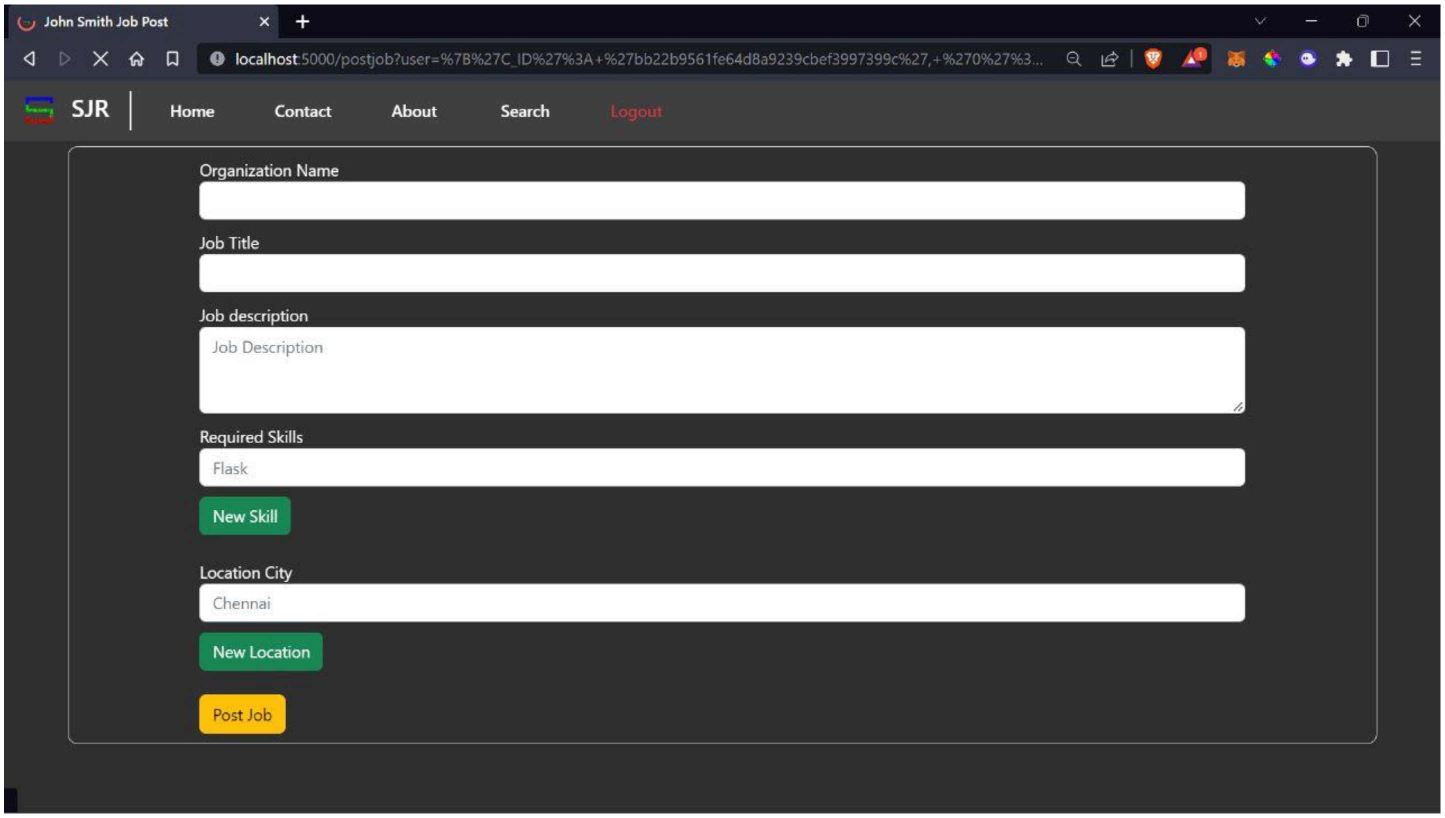
Posted Jobs

Fullstack
by IBM
We need a experienced fullstack dev.
Skills
Fullstack python html css

Tester
by IBM
We need a tester who knows to use most of the testing tools
Skills
Python Testing

Eligible Candidates

Kamaleshpathy V A
Im an Electronics and Communication Engineering student who is interested in learning new technologies and knows 3D modelling IoT programming web development.
Skills
Flask Python


John Smith Job Post

localhost:5000/postjob?user=%7B%27C_ID%27%3A+%27bb22b9561fe64d8a9239cbef3997399c%27,+%270%27%3...

Organization Name

Job Title

Job description
Job Description

Required Skills
Flask

Location City
Chennai

9. RESULTS

Skills/Job seeker application using cloud is developed and executed at the level of completed progress.

10. ADVANTAGES & DISADVANTAGES

Advantages:

- For customers, recommender systems can help them find items which they are interested in.
- High stability compare to Existing system.
- It is an user-friendly application.
- For enterprises, recommender systems can improve the loyalty of their customers by enhancing the user experience and further convert more browsers to consumers.
- Easily accessible and portable.
- Better user experience.

Disadvantage:

- Need a lot of data to effectively make recommendations.
- It works only through internet.
- Device fault may affect the application.

11. CONCLUSION:

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 profifiles 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.

12. FUTURE SCOPE:

In the future we will further explore the design of adaptive interfaces, in order to be in a position to demonstrate a complete adaptive mobile job seeker framework.

13. APPENDIX

Source Code

deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sjr-app

spec:
  replicas: 5
  selector:
   matchLabels:
    app: sjr-app
  template:
   metadata:
    labels:
     app: sjr-app

   spec:
    containers:
    - name: sjr-app-container
      image: jp.icr.io/sjr_final/sjr-app
      imagePullPolicy: Always
      ports:
      - containerPort: 5000
        protocol: TCP
```

ibm-sjrapp-ingress.yaml:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: sjr-app-ingress
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-redirect: "false"

spec:
  # ingressClassName: nginx
  rules:
    - http:
        paths:
          - backend:
              service:
                name: sjr-app-service
                port:
                  number: 5000
            path: /
            pathType: Prefix
```

ibm-sjrapp-service.yaml:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: sjr-app-service
spec:
  type: ClusterIP
  ports:
    - port: 5000
  selector:
        app: sjr-app
```

addjob.css:

```css
.post-job-form {
  display: flex;
  flex-direction: column;
  align-items: center;
  width: 100%;
  margin-top: 65px;
  margin-bottom: 10px;
}

.Org,
.title-job,
.description,
.req_skills,
.locations {
  display: flex;
  flex-direction: column;
}
```

```
.bg {
  backdrop-filter: blur(40px);
  border-radius: 10px;
  padding-block: 10px;
}
.navigation-action {
  display: flex;
  justify-content: space-between;
}
.b-wid {
  width: 20%;
}

.Org,
.title-job,
.description,
.req_skills,
.locations {
  display: flex;
  flex-direction: column;
  margin-bottom: 10px;
}
```

base.css:

```
body {
  background-color: #2f2f2f;
  color: white;
}
.topnav {
  position: fixed;
  top: 0;
  width: 100%;
  height: 60px;
  padding: 10px;
  display: flex;
  align-items: center;
  background-color: #6666664b;
  backdrop-filter: blur(18px);
  z-index: 10;
}

.topnav > * {
  margin-right: 10px;
}

.topnav-title {
  border-right: 2px solid white;
  padding-right: 20px;
  height: 40px;
  display: flex;
  align-items: center;
}

.topnav-title > a {
  text-decoration: none;
  font-size: x-large;
  font-weight: 600;
  color: white;
}

.topnav-item {
  position: relative;
  text-decoration: none;
  color: white;
  font-weight: 500;
  padding: 10px;
  border-radius: 5px;
  width: 100px;
  text-align: center;
  transition: background-color 250ms ease;
}

.topnav-item:hover {
  background-color: rgba(203, 203, 203, 0.516);
  color: black;
}

/* Hide the link that should open and close the topnav on small screens */
.topnav .icon {
  display: none;
}

@media screen and (max-width: 600px) {
  .topnav {
    justify-content: space-between;
  }
  .topnav a:not(:first-child) {
    display: none;
  }
  .topnav a.icon {
    display: block;
    position: relative;
    text-decoration: none;
    color: white;
    font-weight: 500;
    padding: 10px;
    border-radius: 5px;
    width: 100px;
    text-align: center;
    transition: background-color 250ms ease;
  }
  .topnav a.icon:hover {
    background-color: rgba(203, 203, 203, 0.516);
    color: black;
  }
}

/* The "responsive" class is added to the topnav with JavaScript when the user clicks on the icon. This class makes the topnav
look good on small screens (display the links vertically instead of horizontally) */
@media screen and (max-width: 600px) {
  .topnav.responsive {
    position: fixed;
    display: flex;
    flex-direction: column;
    height: 250px;
  }
  .topnav.responsive .topnav-title {
    border: none;
  }
  .topnav.responsive .topnav-item {
    width: 90%;
  }
  .topnav.responsive a.icon {
    text-align: center;
    position: absolute;
    right: 0;
    top: 0;
  }
  .topnav.responsive a {
    float: none;
    display: block;
    text-align: left;
  }
}
```

index.css:

```
.main {
  padding-block-start: 60px;
  background-image: url("https://sjr-app.s3.jp-tok.cloud-object-storage.appdomain.cloud/jobBG.jpg");
  background-repeat: no-repeat;
  background-size: cover;
  filter: grayscale(80%);
  height: 100vh;
}

.content {
  display: flex;
  height: 100%;
  padding: 2%;
}
.left {
  padding: 100px;
  width: 50%;
  height: 100%;
}

.right {
  border-top: 2px solid white;
  border-right: 2px solid white;
  /* border-left: 1px solid white;
  border-bottom: 1px solid white; */
  border-radius: 10px;
  backdrop-filter: blur(40px);
  padding: 100px;
  display: flex;
  flex-direction: column;
}

.ss-button {
  display: flex;
  padding-block-start: 100px;
  flex-direction: column;
  gap: 50px;
}
```

```
.log-button {
  text-decoration: none;
  width: 100%;
  /* height: 5%; */
  background-color: #fff;
  color: #2f2f2f;
  padding: 12px;
  font-weight: 600;
  border-radius: 10px;
  text-align: center;
}

.about-sec {
  display: flex;
  padding: 2%;
  flex-direction: column;
}

.about-p {
  font-size: x-large;
}

.contact-sec {
  display: flex;
  padding: 2%;
  height: 60%;
  flex-direction: column;
}

.contact-form {
  padding-inline: 10%;
  padding-block-start: 20px;
}

@media screen and (max-width: 600px) {
  .content {
    padding: 50px;
  }
  .left {
    display: none;
  }
  .right {
    padding-block-start: 25%;
    padding-inline: 10%;
  }
}
```

recruitment.css:

```
.job-cards {
  margin-top: 10px;
  display: flex;
  flex-direction: column;
}
```

userprofile.css:

```
.main {
  padding-block-start: 50px;
}
.main-content {
  max-width: 1150px;
  margin: auto;
  padding-top: 20px;
}
.contact-details {
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
}
.card-wid {
  width: 70%;
}
.sub-title {
  padding: 10px;
}
.skill-list {
  display: flex;
  flex-wrap: wrap;
  gap: 10px;
}
.ind-skill {
  border: 1px solid black;
  border-radius: 10px;
  padding: 5px 30px 5px 30px;
}
.recommended {
  margin-top: 15px;
}
.job-cards {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  gap: 15px;
}
.apply-btn {
  margin-top: 10px;
  display: flex;
  flex-direction: row-reverse;
}
```

userupdate.css:

```
.update-form {
  display: flex;
  flex-direction: column;
  align-items: center;
  width: 100%;
  margin-top: 65px;
  margin-bottom: 10px;
}

.bg {
  backdrop-filter: blur(40px);
  border-radius: 10px;
  padding-block: 10px;
}
.navigation-action {
  display: flex;
  justify-content: space-between;
}
.b-wid {
  width: 20%;
}

.skills,
.pro-item,
.addresses,
.descreption,
.achievements,
.github-url,
.linkedin-url {
  display: flex;
  flex-direction: column;
  margin-bottom: 10px;
}

/* .skills,
.pro-item,
.addresses,
.descreption,
.achievements,
.github-url,
.linkedin-url > input {
  margin-inline-end: 20%;
} */
```

base.js:

```
function myFunction() {
    var x = document.getElementById("myTopnav");
    if (x.className === "topnav") {
        x.className += " responsive";
    } else {
        x.className = "topnav";
    }
}

document.getElementById('logout').onclick = function logout() {
    localStorage.removeItem('C_ID');
    location.href = '/login?msg=Logged+Out';
}
```

base.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="icon" type="image/x-icon" href="https://sjr-app.s3.jp-tok.cloud-object-storage.appdomain.cloud/sjr_symbol.ico">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">
        <link rel="stylesheet" href="{{ url_for('static', filename='css/base.css') }}">
        {% block head %}{% endblock %}
    </head>
    <body>
        <div class="topnav" id="myTopnav">
            <div class="topnav-title">
                <a href="/">
                <img src="https://sjr-app.s3.jp-tok.cloud-object-storage.appdomain.cloud/sjr_symbol.png" alt="sjr_logo" width="50">
                SJR</a>
            </div>
            <a href="/#" class="topnav-item">Home</a>
            <a href="/#contact" class="topnav-item">Contact</a>
            <a href="/#about" class="topnav-item">About</a>
            <a href="/search" class="topnav-item">Search</a>
            {% block nav %} {% endblock %}
            <a href="javascript:void(0);" class="icon" onclick="myFunction()">
                <i class="fa-solid fa-bars"></i>
            </a>
        </div>
        <div class="main" id="main-content">
            {% block body %}{% endblock %}
        </div>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3" crossorigin="anonymous"></script>
        <script src="https://kit.fontawesome.com/a2758c9efd.js" crossorigin="anonymous"></script>
        <script src="{{ url_for('static', filename='js/base.js') }}"></script>
        <script>
            window.watsonAssistantChatOptions = {
              integrationID: "c74dff4b-0b3e-405a-ae10-9a0d71f50fc9", // The ID of this integration.
              region: "jp-tok", // The region your integration is hosted in.
              serviceInstanceID: "0feb80be-a66e-4dc4-86ab-b96a3fa9bb30", // 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>
    </body>
    </html>
```

forgotpass.html:

```
    {% extends 'base.html' %}

    {% block head %}
    <title>Signin - SJR App</title>
    <style>
        .navigation-action {
            display: flex;
            justify-content: space-between;
        }
```

```
</style>
{% endblock %}

{% block body %}
<main class="container" style="margin-top: 120px;">
    <div class="mx-auto mt-5 border border-2 rounded-3 bg" style="width: 500px;">
        <h2 class="mx-4 mt-2">Forgot Password</h2>
        <form action="" method="post">
            <div class="my-2 mx-4">
                <label for="email">Email</label>
                <input type="email" class="form-control" placeholder="abc@xyz.com" name="email" required id="email"/>
                {% if msg != '' %}
                {{msg}}
                {% endif %}
            </div>
            <div class="navigation-action">
                <a href="/login" class="btn btn-warning my-4 mx-4 mt-2 b-wid">Back</a>
                <input type="button" id="eemail" value="Send mail" class="btn btn-primary my-4 mx-4 mt-2 b-wid"/>
            </div>
        </form>
    </div>
</main>

<script>
    document.getElementById('eemail').onclick = function () {
        const email = document.getElementById('email').value;
        // alert(`/sendemail/${email}`)
        fetch(`/sendmail/${email}`).then(location.href = '/login?msg=Mail+Sent')
}
</script>
{% endblock %}
```

index.html:

```
{% extends 'base.html' %}

{% block head %}
<title>SJR App</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/index.css') }}">
{% endblock %}

{% block body %}
<section class="content">
    <div class="left">
        <h1 class="m-4 pe-5"><sup><i class="fa-solid fa-quote-left"></i></sup> A bend in the road is not the end of the
road…</h1>
        <h1 class="m-4 pe-4">unless you fail to make the turn. <sup><i class="fa-solid fa-quote-right"></i></sup></h1>
    </div>
    <div class="right">
        <h3>Get into the world of job heaven</h3>
        <div class="ss-button">
            <a href="/login" class="log-button">Sigin</a>
            <a href="/signup" class="log-button">Sigup</a>
        </div>
    </div>
</section>
<section class="about-sec" id="about">
    <h1>About</h1>
```

```
        <p class="about-p">We built this app because we know the pain of finding a right job for your skills</p>
        <p class="about-p">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.</p>
        <div class="row row-cols-1 row-cols-md-3 g-4">
            <div class="col">
              <div class="card">
                <img src="https://sjr-app.s3.jp-tok.cloud-object-storage.appdomain.cloud/c_img1.jpg" class="card-img-top" alt="...">
              </div>
            </div>
            <div class="col">
              <div class="card">
                <img src="https://sjr-app.s3.jp-tok.cloud-object-storage.appdomain.cloud/c_img2.jpg" class="card-img-top" alt="...">
              </div>
            </div>
            <div class="col">
              <div class="card">
                <img src="https://sjr-app.s3.jp-tok.cloud-object-storage.appdomain.cloud/c_img3.jpg" class="card-img-top" alt="...">
              </div>
            </div>
          </div>
    </section>
    <section class="contact-sec" id="contact">
        <h1>Contact</h1>
        <div class="contact-form">
            <form action="">
                <div class="mb-3">
                    <label for="exampleFormControlInput1" class="form-label">Email address</label>
                    <input type="email" class="form-control" id="exampleFormControlInput1" placeholder="name@example.com">
                </div>
                <div class="mb-3">
                    <label for="exampleFormControlTextarea1" class="form-label">Tell us about your queries</label>
                    <textarea class="form-control" id="exampleFormControlTextarea1" rows="4" placeholder="I would like to know
about..."></textarea>
                </div>
                <button type="submit" class="btn btn-primary">Submit</button>
            </form>
        </div>
    </section>
    {% endblock %}
```

postjob.html:

```
{% extends 'base.html' %}

{% block head %}
<title>{{user['FIRSTNAME']+" "+user['LASTNAME']+" "}} Job Post</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/addjob.css') }}">
{% endblock %}

{% block nav %}
<a class="topnav-item text-danger"  id="logout">Logout</a>
{% endblock %}
{% block body %}
<main class="container">
    <div class="post-job-form border bg">
        <form action="{{ url_for('addjob') }}" method="post" style="width: 80%;">
            <input type="text" name="c_id" value="{{ user['C_ID'] }}" hidden>
            <div class="Org">
                <label for="organization">Organization Name</label>
                <input class="form-control" type="text" name="organization" id="organization">
            </div>
            <div class="title-job">
                <label for="title">Job Title</label>
                <input class="form-control" type="text" name="title" id="title">
            </div>
            <div class="description">
                <label for="des">Job description</label>
                <textarea class="form-control" name="des" id="des" cols="40" rows="3" placeholder="Job Description"></textarea>
            </div>
            <div>
                <div class="req_skills" id="req_skills">
                    <label for="req_skill">Required Skills</label>
                    <input class="form-control" type="text" name="req_skill" placeholder="Flask">
                </div>
                <a class="btn btn-success mb-4" id="add-skill">New Skill</a>
                <!-- <button id="add-skill" type="button">Add Skill</button> -->
            </div>
            <div>
                <div class="locations" id="locations">
                    <label for="location">Location City</label>
                    <input class="form-control" type="text" name="location" placeholder="Chennai">
                </div>
                <!-- <button id="add-location" type="button">Add Location</button> -->
                <a class="btn btn-success mb-4" id="add-location">New Location</a>
            </div>
            <button class="btn btn-warning" id="submit">Post Job</button>
        </form>
    </div>
</main>

<script>
    document.getElementById('add-skill').onclick = function () {
        const input_tag = document.createElement('input');
        input_tag.setAttribute('name', 'req_skill');
        input_tag.setAttribute('placeholder', 'Flask');
        input_tag.setAttribute('type', 'text');
        const skills_div = document.getElementById('req_skills');
        skills_div.appendChild(input_tag);
    }
    document.getElementById('add-location').onclick = function () {
        const input_tag = document.createElement('input');
        input_tag.setAttribute('name', 'location');
        input_tag.setAttribute('placeholder', 'Chennai');
        input_tag.setAttribute('type', 'text');
        const skills_div = document.getElementById('locations');
        skills_div.appendChild(input_tag);
    }
</script>
{% endblock %}
```

recruitmentpage.html:

```
{% extends 'base.html' %}

{% block head %}
<title>{{user['FIRSTNAME']+" "+user['LASTNAME']+" "}}</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/userprofile.css') }}">
<link rel="stylesheet" href="{{ url_for('static', filename='css/recruitment.css') }}">
{% endblock %}

{% block nav %}
<a class="topnav-item text-danger"  id="logout">Logout</a>
{% endblock %}

{% block body %}
<div class="main-content fs-5">
    <div class="card">
        <h5 class="card-header text-dark">Details - Business</h5>
        <div class="card-body card-wid">
            <h5 class="card-title text-dark">{{user['FIRSTNAME']+" "+user['LASTNAME']}}</h5>
            <div class="contact-details">
                <div class="mob-no text-dark">Mobile: {{ user['PHONE'] }}</div>
                <div class="email-contact text-dark">E-mail: {{ user['EMAIL'] }}</div>
            </div>
        </div>
        <div class="apply-btn pe-4 pb-4">
            <a href="{{ url_for('postjob', user=user) }}" class="btn btn-warning">Post new job</a>
        </div>
    </div>
    <div class="job-cards">
        <h5 class="card-header">Posted Jobs</h5>
        {% if data|length != 0%}
        {% for job in data %}
        <div class="card border-warning" style="width: 100%;">
            <div class="card-body">
                <h5 class="card-title text-dark">{{job['TITLE']}}</h5>
                <h6 class="card-subtitle mb-2 text-muted">by {{job['ORGANIZATION']}}</h6>
                <p class="card-text text-dark">{{job['DES']}}</p>
                {% if job['REQ_SKILL'] != '' %}
                    <h5 class="text-dark">Skills</h5>
                    <div class="skill-list text-dark">
                        {% for skill in job['REQ_SKILL'].split(',') %}
                            <div class="ind-skill">{{skill}}</div>
                        {% endfor %}
                    </div>
                {% endif %}
            </div>
        </div>
        {% endfor %}
        {% else %}
        <h6 class="card-subtitle mb-2">Post new job to see your openings.</h6>
        {% endif %}
    </div>
    <div class="job-cards">
        <h5 class="card-header">Eligible Candidates</h5>
        {% if candidates|length != 0 %}
        {% for x in range(candidates|length) %}
        <div class="card border-warning" style="width: 100%;">
            <div class="card-body">
```

```
                    <h5 class="card-title text-dark">{{info[x]['FIRSTNAME'] + " " + info[x]['LASTNAME']}}</h5>
                    <h6 class="card-subtitle mb-2 text-muted">{{candidates[x]['DES']}}</h6>
                    {% if candidates[x]['SKILLS'] != '' %}
                        <h5 class="text-dark">Skills</h5>
                        <div class="skill-list text-dark">
                            {% for skill in candidates[x]['SKILLS'].split(',') %}
                                <div class="ind-skill">{{skill}}</div>
                            {% endfor %}
                        </div>
                    {% endif %}
                </div>
            </div>
            {% endfor %}
            {% else %}
            <h6 class="card-subtitle mb-2">No eligible candidates yet</h6>
            {% endif %}

    <script>
        localStorage.setItem("C_ID", "{{ user['C_ID'] }}");
        document.getElementById('logout').onclick = function logout() {
            let token = localStorage.removeItem('C_ID');
            location.href = '/';
        }
    </script>
    {% endblock %}
```

searchpage.html:

```
{% extends 'base.html' %}

{% block head %}
<title>Search - SJR App</title>
<style>
    .main-content {
        padding-block-start: 60px;
        padding: 10% 20% 10% 20%;
    }
    .card-list {
        margin-top: 10px;
        display: flex;
        flex-direction: column;
        gap: 10px;
    }
    .skill-list {
        display: flex;
        flex-wrap: wrap;
        gap: 10px;
    }
</style>
{% endblock %}

{% block body %}
<div class="main-content">
    <form action="{{ url_for('searchbackend') }}" method="get">
        <div class="input-group mb-3">
            <input type="text" name="searchval" class="form-control" placeholder="Search your job" aria-label="Recipient's username"
aria-describedby="button-addon2">
```

```
            <button class="btn btn-outline-primary text-light bg-primary" type="submit" id="button-addon2">Search</button>
        </div>
    </form>
    <div class="card-list">
        {% if offers|length != 0%}
        {% for job in offers %}
        <div class="card border-warning" style="width: 100%;">
            <div class="card-body">
                <h5 class="card-title text-dark">{{job['TITLE']}}</h5>
                <h6 class="card-subtitle mb-2 text-muted">by {{job['ORGANIZATION']}}</h6>
                <p class="card-text text-dark">{{job['DES']}}</p>
                {% if job['REQ_SKILL'] != '' %}
                    <h5 class="text-dark">Skills</h5>
                    <div class="skill-list text-dark">
                        {% for skill in job['REQ_SKILL'].split(',') %}
                            <div class="ind-skill">{{skill}}</div>
                        {% endfor %}
                    </div>
                {% endif %}
            </div>
        </div>
        {% endfor %}
        {% endif %}
    </div>
</div>
{% endblock %}
```

signin.html:

```
{% extends 'base.html' %}

{% block head %}
<title>Signin - SJR App</title>
<style>
    .main {
        padding-block-start: 60px;
        background-image: url("https://sjr-app.s3.jp-tok.cloud-object-storage.appdomain.cloud/jobBG.jpg");
        background-repeat: no-repeat;
        background-size: cover;
        filter: grayscale(80%);
        height: 100vh;
    }
    .bg {
        backdrop-filter: blur(40px);
        border-radius: 10px;
        padding-block: 10px;
    }
    .navigation-action {
        display: flex;
        justify-content: space-between;
    }
    .b-wid {
        width: 20%;
    }
</style>
{% endblock %}

{% block body %}
```

```
<main class="container">
    <div class="mx-auto mt-5 border bg" style="width: 500px;">
        <h2 class="mx-4 mt-2">Login</h2>
        <form action="{{ url_for('accessbackend') }}" method="get">
            <div class="my-2 mx-4">
                <label for="email">Email</label>
                <input type="email" class="form-control" placeholder="abc@xyz.com" name="email" required/>
            </div>
            <div class="my-2 mx-4">
                <label for="password">Password</label>
                <input type="password" class="form-control" placeholder="Password" name="password" required/>
                <a href="/forgot" class="text-info">Forgot password</a>
            </div>
            {% if msg == "Account Created" %}
            <div class="my-2 mx-4 p-2 bg-success d-flex justify-content-center rounded rounded-3">
                {{msg}}
            </div>
            {% elif msg != '' %}
            <div class="my-2 mx-4 p-2 bg-danger d-flex justify-content-center rounded rounded-3">
                {{msg}}
            </div>
            {% endif %}
            <div class="navigation-action">
                <a href="/" class="btn btn-warning my-4 mx-4 mt-2 b-wid">Back</a>
                <input type="submit" value="Login" class="btn btn-primary my-4 mx-4 mt-2 b-wid"/>
            </div>
        </form>
    </div>
</main>
{% endblock %}
```

signup.html:

```
{% extends 'base.html' %}

{% block head %}
<title>Signup - SJR</title>
<style>
    .main {
        padding-block-start: 60px;
        background-image: url("https://sjr-app.s3.jp-tok.cloud-object-storage.appdomain.cloud/jobBG.jpg");
        background-repeat: no-repeat;
        background-size: cover;
        filter: grayscale(80%);
        height: 100vh;
    }
    .bg {
        backdrop-filter: blur(40px);
        border-radius: 10px;
        padding-block: 10px;
    }
    .navigation-action {
        display: flex;
        justify-content: space-between;
    }
    .b-wid {
        width: 20%;
    }
</style>
{% endblock %}

{% block body %}
<main class="container">
    <div class="mx-auto mt-5 border bg" style="width: 500px;">
        <h2 class="mx-4 mt-2">Sign Up</h2>
        <form action="{{ url_for('accessbackend') }}" method="post">
            <div class="my-2 mx-4">
                <label for="firstname">First name</label>
                <input type="text" class="form-control" placeholder="John" name="firstname" required/>
            </div>
            <div class="my-2 mx-4">
                <label for="lastname">Last name</label>
                <input type="text" class="form-control" placeholder="Smith" name="lastname" required/>
            </div>
            <div class="my-2 mx-4">
                <label for="email">Email</label>
                <input type="email" class="form-control" placeholder="abc@xyz.com" name="email" required/>
            </div>
            <div class="my-2 mx-4">
                <label for="phone">Phone</label>
                <input type="number" class="form-control" placeholder="1234567890" name="phone" required/>
            </div>
            <div class="my-2 mx-4">
                <label for="password">Password</label>
                <input type="password" class="form-control" placeholder="Password" name="password" required/>
            </div>
            <div class="my-2 mx-4">
                <label for="dob" class="me-4">Account Type</label>
                <input class="form-check-input me-1" type="radio" name="accounttype" id="flexRadioDefault1" value="user">
                <label class="form-check-label me-5" for="flexRadioDefault1">User</label>
                <!-- <input type="radio" class="form-control" name="accounttype" value="user" required> User -->
                <input class="form-check-input me-1" type="radio" name="accounttype" id="flexRadioDefault1" value="business">
                <label class="form-check-label" for="flexRadioDefault1">Business</label>
                <!-- <input type="radio" class="form-control" name="accounttype" value="business" required/> -->
            </div>
            <div class="navigation-action">
                <a href="/" class="btn btn-warning my-4 mx-4 mt-2 b-wid">Back</a>
                <input type="submit" value="Sign up" class="btn btn-primary my-4 mx-4 mt-2 b-wid"/>
            </div>
        </form>
    </div>
</main>
{% endblock %}
```

userprofile.html:

```
{% extends 'base.html' %}

{% block head %}
<title>{{user['FIRSTNAME']+" "+user['LASTNAME']+" "}}Profile</title>
```

```
<link rel="stylesheet" href="{{ url_for('static', filename='css/userprofile.css') }}">
{% endblock %}

{% block nav %}
<a class="topnav-item text-danger"  id="logout">Logout</a>
{% endblock %}

{% block body %}
<div class="main-content fs-5">
    <div class="card">
        <h5 class="card-header text-dark">Bio</h5>
        <div class="card-body card-wid">
            <h5 class="card-title text-dark">{{user['FIRSTNAME']+" "+user['LASTNAME']}}</h5>
            {% if data['DES'] != '' %}
                <p class="card-text text-dark">{{data['DES']}}</p>
            {% endif %}
            <div class="contact-details">
                <div class="mob-no text-dark">Mobile: {{ user['PHONE'] }}</div>
                <div class="email-contact text-dark">E-mail: {{ user['EMAIL'] }}</div>
                {% if data['LINKEDIN'] != '' %}
                <div class="linked-link">
                    <a href="https://{{data['LINKEDIN']}}">LinkedIn: {{data['LINKEDIN'].split('/')[2]}}</a>
                </div>
                {% endif %}
                {% if data['GITHUB'] != '' %}
                <div class="github-link">
                    <a href="https://{{data['GITHUB']}}">GitHub: {{data['GITHUB'].split('/')[1]}}</a>
                </div>
                {% endif %}
                {% if data['ADDRESS'] != '' %}
                <div class="address text-dark">
                    Address: {{data['ADDRESS']}}
                </div>
                {% endif %}
            </div>
        </div>
        <div style="border-bottom: 1px solid #cbcbcb;"></div>
        {% if data['SKILLS'] != '' %}
        <div class="sub-title">
            <h5 class="text-dark">Skills</h5>
            <div class="skill-list text-dark">
                {% for skill in data['SKILLS'].split(',') %}
                    <div class="ind-skill">{{skill}}</div>
                {% endfor %}
            </div>
        </div>
        {% endif %}
        {% if data['PROJECT'] != '' %}
        <div class="sub-title">
            <h5 class="text-dark">Projects</h5>
            <div class="text-dark">
                {% set projects = data['PROJECT'].split(',') %}
                {% set project_des = data['PROJECT_DES'] %}
                {% set length = projects|length %}
                {% for i in range(length) %}
                    <div class="ind-skill">
                        <h4>{{projects[i]}}</h4>
                        <p>{{project_des}}</p>
                    </div>
                {% endfor %}
            </div>
        </div>
        {% endif %}
        {% if data['PROJECT'] != '' %}
        <div class="sub-title">
            <h5 class="text-dark">Achievements</h5>
            <div class="skill-list text-dark">
                {% for achieve in data['ACHIEVE'].split(',') %}
                    <div class="ind-skill">{{achieve}}</div>
                {% endfor %}
            </div>
        </div>
        {% endif %}
        <div class="apply-btn pe-4 pb-4">
            <a href="{{ url_for('updateprofile', user=user) }}" class="btn btn-warning">UPDATE</a>
        </div>
    </div>
    {% if openings|length != 0 %}
    <div class="recommended">
        <div class="card">
            <h5 class="card-header text-dark">Recommended Jobs</h5>
            <!-- <div class="job-cards"> -->
            {% for offer in openings %}
            <div class="card-body">
                <div class="card" style="width: 80%;">
                    <div class="card-body">
                        <h5 class="card-title text-dark">{{offer['TITLE']}}</h5>
                        <h6 class="card-subtitle mb-2 text-muted">From {{offer['ORGANIZATION']}}</h6>
                        <p class="card-text text-dark">Description: {{offer['DES']}}</p>
                        <p class="card-text text-dark">Location: {{offer['LOCATION']}}</p>
                        <div class="skill-list text-dark">
                            {% for skill in offer['REQ_SKILL'].split(',') %}
                                <div class="ind-skill">{{skill}}</div>
                            {% endfor %}
                        </div>
                        <div class="apply-btn">
                            <a href="#" class="btn btn-primary" style="width: 10%;">Apply</a>
                        </div>
                    </div>
                </div>
            </div>
            {% endfor %}
            <!-- </div> -->
        </div>
    </div>
    {% endif %}
</div>

<script>
    localStorage.setItem("C_ID", "{{ user['C_ID'] }}");
</script>
{% endblock %}
```

userupdate.html:

```
{% extends 'base.html' %}

{% block head %}
<title>{{user['FIRSTNAME']+" "+user['LASTNAME']+" "}} Update</title>
```

```
<link rel="stylesheet" href="{{ url_for('static', filename='css/userupdate.css')}}">
{% endblock %}

{% block nav %}
<a class="topnav-item text-danger"  id="logout">Logout</a>
{% endblock %}

{% block body %}
<main class="container">
    <div class="update-form border bg">
        <form action="{{ url_for('modifyskills') }}" id="the-form" method="post" style="width: 80%;">
            <input type="text" name="c_id" value="{{ user['C_ID'] }}" hidden>
            <div class="descreption">
                <label for="des">Bio</label>
                <textarea class="form-control" name="des" id="des" cols="40" rows="3" placeholder="Tell about yourself"></textarea>
            </div>
            <div>
                <div class="skills" id="skills">
                    <label for="name">Skills</label>
                    <input class="form-control" type="text" name="skill" placeholder="Python">
                </div>
                <a class="btn btn-success mb-4" id="add-skill">New Skill</a>
                <!-- <button id="add-skill" type="button">New Skill</button> -->
            </div>
            <div>
                <div class="projects" id="projects">
                    <div class="pro-item">
                        <label for="project">Project Title</label>
                        <input class="form-control" type="text" name="project" placeholder="Cloud app">
                        <label for="project_des">Descreption</label>
                        <textarea class="form-control" name="project_des" cols="40" rows="3" placeholder="Describe your
project"></textarea>
                    </div>
                </div>
                <a class="btn btn-success mb-4" id="add-des">New Project</a>
                <!-- <button id="add-des" type="button">New Project</button> -->
            </div>
            <div class="addresses">
                <label for="address">Address</label>
                <input class="form-control" type="text" name="address" id="address">
            </div>
            <div class="linkedin-url">
                <label for="linkedin">LinkedIn URL</label>
                <input class="form-control" type="text" name="linkedin" id="linkedin">
            </div>
            <div class="github-url">
                <label for="github">Github URL</label>
                <input class="form-control" type="text" name="github" id="github">
            </div>
            <div>
                <div class="achievements" id="achievements">
                    <label for="achievement">Achievements</label>
                    <input class="form-control" type="text" name="achievement" placeholder="CCNA">
                </div>
                <a class="btn btn-success mb-4" id="add-achieve">New achievement</a>
                <!-- <button id="add-achieve" type="button">add element</button> -->
            </div>
            <button class="btn btn-warning" id="submit">Update</button>
        </form>
    </div>
</main>
```

```
<script>
document.getElementById('add-skill').onclick = function () {
    const input_tag = document.createElement('input');
    input_tag.setAttribute('name', 'skill');
    input_tag.setAttribute('placeholder', 'Python');
    input_tag.setAttribute('type', 'text');
    const skills_div = document.getElementById('skills');
    skills_div.appendChild(input_tag);
}

document.getElementById('add-achieve').onclick = function () {
    const input_ach = document.createElement('input');
    input_ach.setAttribute('name', 'achievement');
    input_ach.setAttribute('placeholder', 'CCNA');
    input_ach.setAttribute('type', 'text');
    const achievements_div = document.getElementById('achievements');
    achievements_div.appendChild(input_ach);
}

document.getElementById('add-des').onclick = function () {
    const lab_title = document.createElement('label');
    lab_title.setAttribute('for', 'project');
    lab_title.innerText = "Project Title";
    const input_pro = document.createElement('input');
    input_pro.setAttribute('name', 'project');
    input_pro.setAttribute('placeholder', 'Cloud app');
    input_pro.setAttribute('type', 'text');
    const lab_des = document.createElement('label');
    lab_des.setAttribute('for', 'project_des');
    lab_des.innerText = "Description";
    const input_des = document.createElement('textarea');
    input_des.setAttribute('name', 'project_des');
    input_des.setAttribute('placeholder', 'Describe your project');
    input_des.setAttribute('cols', '40');
    input_des.setAttribute('rows', '3');
    const div_pro = document.createElement('div');
    div_pro.setAttribute('class', 'pro-item');
    div_pro.append(lab_title, input_pro, lab_des, input_des);
    const master_div = document.getElementById('projects');
    master_div.appendChild(div_pro);
}
</script>
{% endblock %}
```

Dockerfile:

```
FROM python:3.10-buster

WORKDIR /app

COPY . .

RUN pip install --no-cache-dir -r requirements.txt

CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"]
```

app.py:

```
import json
from flask import Flask, render_template, request, redirect, url_for, session
import ibm_db
import uuid
import sendemail

app = Flask(__name__)
app.secret_key = "password"

conn =
ibm_db.connect("DATABASE=<database>;HOSTNAME=<hostname>;PORT=<port>;SECURITY=SSL;SSLServerCertificate=Digi
CertGlobalRootCA.crt;UID=<user>;PWD=<password>", "", "")

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

@app.route("/login")
def signin():
    try:
        msg = request.args['msg']
    except:
        msg = ""
    return render_template('signin.html', msg=msg)

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

@app.route("/forgot")
def forgot():
    return render_template('forgotpass.html')

@app.route("/profile")
def profile():
    user = request.args['user']
    user=json.loads(user)

    # fetch user professional profile
    query = f"SELECT * FROM skillset WHERE c_id='{user['C_ID']}'"
    stmt = ibm_db.exec_immediate(conn, query)
    data = ibm_db.fetch_both(stmt)

    # fetch all eligible jobs
    openings = []
    skillssss = data['SKILLS'].split(',')
    print(skillssss)
    for i in skillssss:
        if i == '':
            continue
        query = f"SELECT * FROM openings WHERE REGEXP_LIKE (req_skill, '\\b{i.strip()}\\b', 'i')"
        stmt = ibm_db.exec_immediate(conn, query)
        dictionary = ibm_db.fetch_both(stmt)
        while dictionary != False:
            if dictionary not in openings:
                openings.append(dictionary)
            dictionary = ibm_db.fetch_both(stmt)

    return render_template('userprofile.html', user=user, data=data, openings=openings)

@app.route("/recruitment")
def recruitment():
    user = request.args['user']
    user=json.loads(user)

    # fetch all posted job opennings
    data = []
    query = f"SELECT * FROM openings WHERE c_id='{user['C_ID']}'"
    stmt = ibm_db.exec_immediate(conn, query)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        data.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    # fetch all eligible candidates
    candid = []
    for job in data:
        if job == '':
            continue
        skillls = job['REQ_SKILL'].split(',')

        for i in skillls:
            query = f"SELECT * FROM skillset WHERE REGEXP_LIKE (SKILLS, '\\b{i.strip()}\\b', 'i')"
            stmt = ibm_db.exec_immediate(conn, query)
            dictionary = ibm_db.fetch_both(stmt)

            while dictionary != False:
                if dictionary not in candid:
                    candid.append(dictionary)
                dictionary = ibm_db.fetch_both(stmt)

    candid_info = []
    for i in candid:
        query = f"SELECT c_id, firstname, lastname FROM customer where c_id='{i['C_ID']}'"
        stmt = ibm_db.exec_immediate(conn, query)
        dictionary = ibm_db.fetch_both(stmt)

        while dictionary != False:
            if dictionary not in candid:
                candid_info.append(dictionary)
            dictionary = ibm_db.fetch_both(stmt)
    print(candid_info)
    return render_template('recruitmentpage.html', user=user, data=data, candidates=candid, info=candid_info)

@app.route("/postjob")
def postjob():
    user = request.args.get('user')
    user = user.replace("\'", '\"')
    user = json.loads(user)
    return render_template('postjob.html', user=user)

@app.route("/updateprofile")
def updateprofile():
    user = request.args.get('user')
    user = user.replace("\'", '\"')
    user = json.loads(user)
    return render_template('userupdate.html', user=user)

@app.route("/modifyskills", methods=['POST', 'GET'])
def modifyskills():
    if request.method == 'POST':
        c_id = request.form['c_id']
        skills = request.form.getlist('skill')
        project = request.form.getlist('project')
        project_des = request.form.getlist('project_des')
        address = request.form['address'].replace('\'', " ").replace('\"', " ")
        linkedin = request.form['linkedin'].replace('\'', " ").replace('\"', " ")
        github = request.form['github'].replace('\'', " ").replace('\"', " ")
        achieve = request.form.getlist('achievement')
        des = request.form['des'].replace('\'', " ").replace('\"', " ")
        query = f"UPDATE skillset SET skills='{','.join(skills)}', project='{','.join(project)}', project_des='{','.join(project_des)}',
address='{address}', linkedin='{linkedin}', github='{github}', achieve='{','.join(achieve)}', des='{des}' WHERE c_id='{c_id}'"
        print(query)
        update_stmt = ibm_db.prepare(conn, query)
        ibm_db.execute(update_stmt)
        query = f"SELECT * FROM customer WHERE c_id='{c_id}'"
        stmt = ibm_db.exec_immediate(conn, query)
        dictionary = ibm_db.fetch_both(stmt)
        dictionary = json.dumps(dictionary)
    return redirect(url_for('profile', user=dictionary))

@app.route("/addjobs", methods=['POST', 'GET'])
def addjob():
    if request.method == 'POST':
        c_id = request.form['c_id']
        organization = request.form['organization'].replace('\'', " ").replace('\"', " ")
        title = request.form['title'].replace('\'', " ").replace('\"', " ")
        des = request.form['des'].replace('\'', " ").replace('\"', " ")
        req_skill = request.form.getlist('req_skill')
        location = request.form.getlist('location')
        query = f"INSERT INTO openings VALUES ('{c_id}', '{organization}', '{title}', '{des}', '{','.join(req_skill)}',
'{','.join(location)}')"
        update_stmt = ibm_db.prepare(conn, query)
        ibm_db.execute(update_stmt)
        query = f"SELECT * FROM customer WHERE c_id='{c_id}'"
        stmt = ibm_db.exec_immediate(conn, query)
        dictionary = ibm_db.fetch_both(stmt)
        dictionary = json.dumps(dictionary)
    return redirect(url_for('recruitment', user=dictionary))

@app.route("/accessbackend", methods=['POST', 'GET'])
def accessbackend():
    if request.method == 'POST':
        firstname = request.form['firstname']
        lastname = request.form['lastname']
        email = request.form['email']
        phone = request.form['phone']
        password = request.form['password']
        account_type = request.form['accounttype']

        query = "SELECT * FROM customer WHERE email =?"
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if account:
            return "user exists"
        else:
            insert_query = "INSERT INTO customer VALUES (?,?,?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_query)
            c_id = str(uuid.uuid4().hex)
            ibm_db.bind_param(prep_stmt, 1, c_id)
            ibm_db.bind_param(prep_stmt, 2, str(firstname))
            ibm_db.bind_param(prep_stmt, 3, str(lastname))
            ibm_db.bind_param(prep_stmt, 4, str(email))
            ibm_db.bind_param(prep_stmt, 5, str(phone))
            ibm_db.bind_param(prep_stmt, 6, str(password))
            ibm_db.bind_param(prep_stmt, 7, str(account_type))
            ibm_db.execute(prep_stmt)
            if account_type == 'user':
                query = f"INSERT INTO skillset VALUES ('{c_id}', '','','','','','','')"
                stmt = ibm_db.prepare(conn, query)
                ibm_db.execute(stmt)

        return redirect(url_for('signin', msg="Account Created"))

    else:
        temp_user_email = request.args.get('email')
        temp_user_password = request.args.get('password')
        msg = ''

        query = f"SELECT * FROM customer WHERE email='{temp_user_email}'"
        stmt = ibm_db.exec_immediate(conn, query)
        dictionary = ibm_db.fetch_both(stmt)

        if dictionary:
            if temp_user_password == dictionary['PASSWORD']:
                if dictionary['ACCOUNT_TYPE'] == 'user':
                    dictionary = json.dumps(dictionary)
                    return redirect(url_for('profile', user=dictionary))
                else:
                    dictionary = json.dumps(dictionary)
                    return redirect(url_for('recruitment', user=dictionary))
            else:
                print(dictionary['PASSWORD'])
                msg = "wrong password"
                return redirect(url_for('signin', msg=msg))
        msg = "No user found"
        return redirect(url_for('signin', msg=msg))

@app.route("/sendmail/<mail>")
def sendmail(mail):
    query = f"SELECT password FROM customer WHERE email='{mail}'"
    stmt = ibm_db.exec_immediate(conn, query)
    dictionary = ibm_db.fetch_both(stmt)
    if dictionary:
        sendemail.send_mail(mail, dictionary['PASSWORD'])
    return 'sent'

@app.route("/search")
def search():
    offers = []
    query = f"SELECT * FROM openings"
    stmt = ibm_db.exec_immediate(conn, query)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        if dictionary not in offers:
            offers.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    return render_template('searchpage.html', offers=offers)

@app.route("/searchbackend", methods=['GET'])
def searchbackend():
    offers = []
    if request.method == 'GET':
        print(request.form)
        search = request.args.get('searchval')
        query = f"SELECT * FROM openings WHERE REGEXP_LIKE (req_skill, '\\b{search.strip()}\\b', 'i')"
        stmt = ibm_db.exec_immediate(conn, query)
        dictionary = ibm_db.fetch_both(stmt)
        while dictionary != False:
            if dictionary not in offers:
                offers.append(dictionary)
            dictionary = ibm_db.fetch_both(stmt)
        query = f"SELECT * FROM openings WHERE REGEXP_LIKE (title, '\\b{search.strip()}\\b', 'i')"
        stmt = ibm_db.exec_immediate(conn, query)
        dictionary = ibm_db.fetch_both(stmt)
        while dictionary != False:
            if dictionary not in offers:
                offers.append(dictionary)
            dictionary = ibm_db.fetch_both(stmt)
    return render_template('searchpage.html', offers=offers)

if __name__ == '__main__':
        app.run(debug=True, port=5000, host="0.0.0.0")
```

sendemail.py:

```
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

def send_mail(to, msg):
    message = Mail(
        from_email='asokankamalesh@gmail.com',
        to_emails= to,
        subject='Your password',
        html_content=f'<strong>Your password is {msg}</strong>')
    try:
        sg = SendGridAPIClient('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')
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e.message)
```

GitHub & Project Demo Link

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-22660-1659855811

Project Demo Link:

https://drive.google.com/file/d/1aT7NBG5de0EVgTqOMi4Os9FtALWdE6zm/view?usp=share_link

Deployment Link:

http://169.51.203.139:31754/