1. INTRODUCTION

1.1 Project overview

This project perspective offers a summary of the job and skill recommendations for anyone considering a career in any industry. It talks about the crucial function that every industry plays in businesses and the different talents required for success in this industry. Additionally, it describes the many employment options and business models available to experts in every discipline.

1.2 Purpose

Having a variety of abilities but unsure of which position would be ideal for you? No need to be concerned! We have developed a talent recommender solution via which both experienced workers and recent graduates may login, browse for available positions, or speak with a chatbot directly to land their ideal position.

To build a complete online application that can show available jobs depending on the users' skill sets. The database contains both the user's data and their information. Based on the user's skill set, a notification is delivered when there is an opening. The chatbot will be used by the user, who may engage with it and get suggestions depending on his abilities. To obtain information about the available positions in the market, we may use a jobsearch API that will retrieve the necessary information straight from a website.

2. LITERATURE SURVEY

LITRATURESURVEY1:

NAMEOFTHEPAPER **:**JobRecommendationbasedonJobSeekerSkills.

NAMEOF THEAUTHOR: JorgeValverde-Rebaza,RicardoPuma,Paul Bustios,Nathalia C.Silva.

JOURNALPUBLISHED **:** First Workshop on Narrative Extraction From Textco-located with40thEuropean ConferenceonInformationRetrieval.

PUBLISHEDMONTH **:**March

PUBLISHEDYEAR **:**2018
OBJECTIVEOFTHEPROJECT:

- Inthis ,whenacandidatesubmitshis/herprofileatajobseeker engine.
- Their job recommendations are mostly suggested taking their academic qualificationandworkexperienceintoconsiderations.

LITRATURESURVEY2:

NAMEOF THEPAPER **:**Asurveyofjobrecommendersystems.

NAMEOF THEAUTHOR **:**ShahaAlotaibi.

JOURNALPUBLISHED :InternationalJournalofPhysical Sciences

PUBLISHEDMONTH :July

PUBLISHEDYEAR :2012

OBJECTIVEOFTHEPROJECT:

- The recommender system technology aims to help users in finding items that matchtheir personnel interests, it has a successful usage in e-commerce applications to dealwithproblems related to informationoverload efficiently.
- This article will present a survey of e-recruiting process and existing recommendationapproachesforbuildingpersonalizedrecommendersystemsfor candidate.

TECHNOLOGYUSED: Booleansearchmethods

LITRATURESURVEY3:

NAMEOF THEPAPER : A Research of Job Recommendation System Based on CollaborativeFiltering.

NAMEOF THEAUTHOR:ChengYang,YingyaZhang,ZhixiangNiu.

JOURNALPUBLISHED : 2014 Seventh International Symposium on ComputationalIntelligenceand Design.

PUBLISHEDMONTH :December

PUBLISHEDYEAR 2014

OBJECTIVEOFTHEPROJECT:

- It analyze the candidate's resume and the companies' recruitment guidelines.
- To compare and come to a better conclusion upon finding the best suited candidates for the job.

TECHNOLOGYUSED: Collaborativefilteringalgorithm.

LITRATURESURVEY4:

NAMEOFTHEPAPE :JobRecommendationthroughProgressionofJobSelection.

NAMEOF THEAUTHOR:AmberNigam,Aakash Roy,HartaranSingh,HarsimranWaila.

JOURNALPUBLISHED : 2019 IEEE 6th International Conference on Cloud Computing andIntelligenceSystems(CCIS).

PUBLISHEDMONTH :April

PUBLISHEDYEAR 2020

OBJECTIVEOFTHEPROJECT:

- It gradually incorporates the dynamics associated with a highly turbulent employment market by using the candidates' career preferences.
- Bidirectional Long Short Term Memory Networks (Bi-LSTM) with attention to machine learning job recommendation have produced the greatest outcomes.

TECHNOLOGYUSED: Filter-basedtechnique.

LITARTURESURVEY5:

NAMEOF THEPAPER **:**JobRecommenderSystems.
NAMEOF THEAUTHOR: JuhiDhameliya,NikitaDesai.

JOURNALPUBLISHED **:** 2019 Innovations in Power and Advanced Computing Technologies(i-PACT).

PUBLISHEDMONTH **:**March

PUBLISHEDYEAR 2019

OBJECTIVEOFTHEPROJECT:

- It is employed in the creation of systems that provide both recruiters and job seekers with customised recommendations.
- The fundamental problem with these portals is that they are unable to comprehend how difficult it is to connect candidates' preferences with organisational requirements.

TECHNOLOGY USED : Boolean search methods-Word matching algorithms.

2.1 Problem Statement Definition

Job skills recommended application

Problem Statement:

Goal:

For students, a job search has to be really intuitive so they may locate employment that fits their abilities, position, industry, role, and location by company name.

- The job skills recommended application is an illustration of a search where the documents are lengthy due to the information in applicant resumes.
- A vast array of fields must be available for the search provider over the candidate database.

Problem:

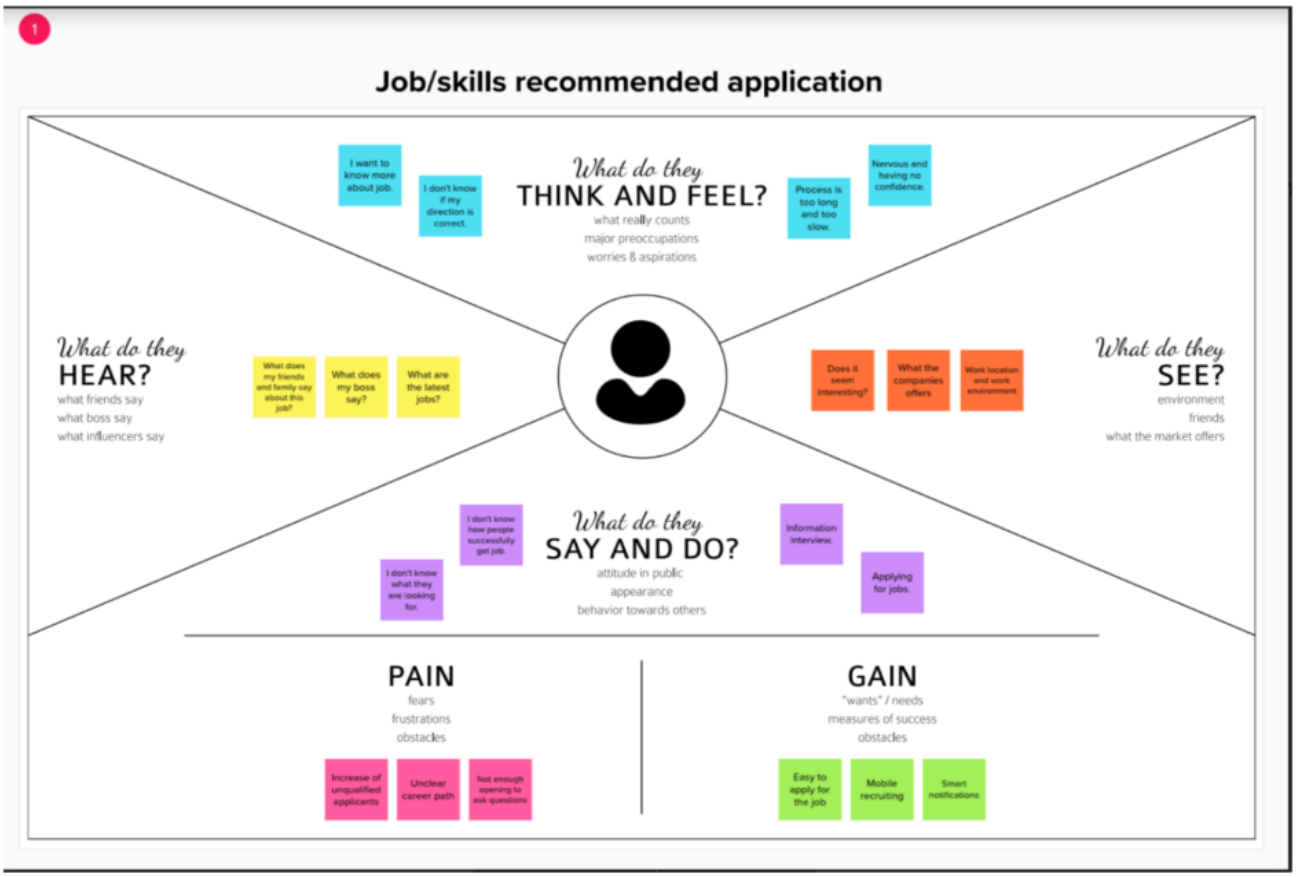
The majority of open positions in Nigeria can only be applied for at the agency, which forces job searchers to visit the agency to check the available employment there. This is an issue because recruiting is currently done manually.

Solution:

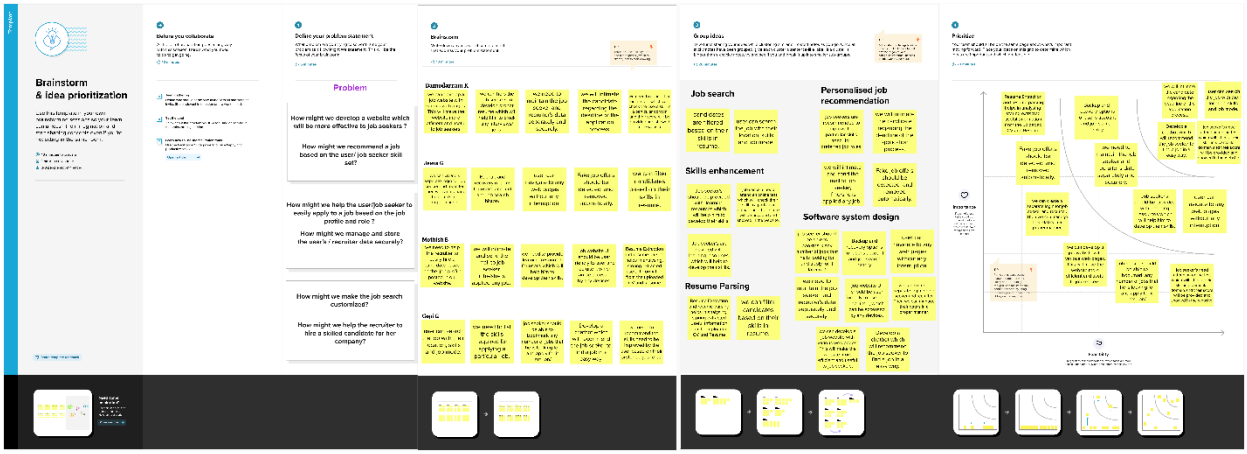
"The goal of a job-oriented application is to assist both recruiters and job seekers in locating the ideal company or employers."

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation and Brainstroming:



3.3 Proposed Solution

ProposedSolution:

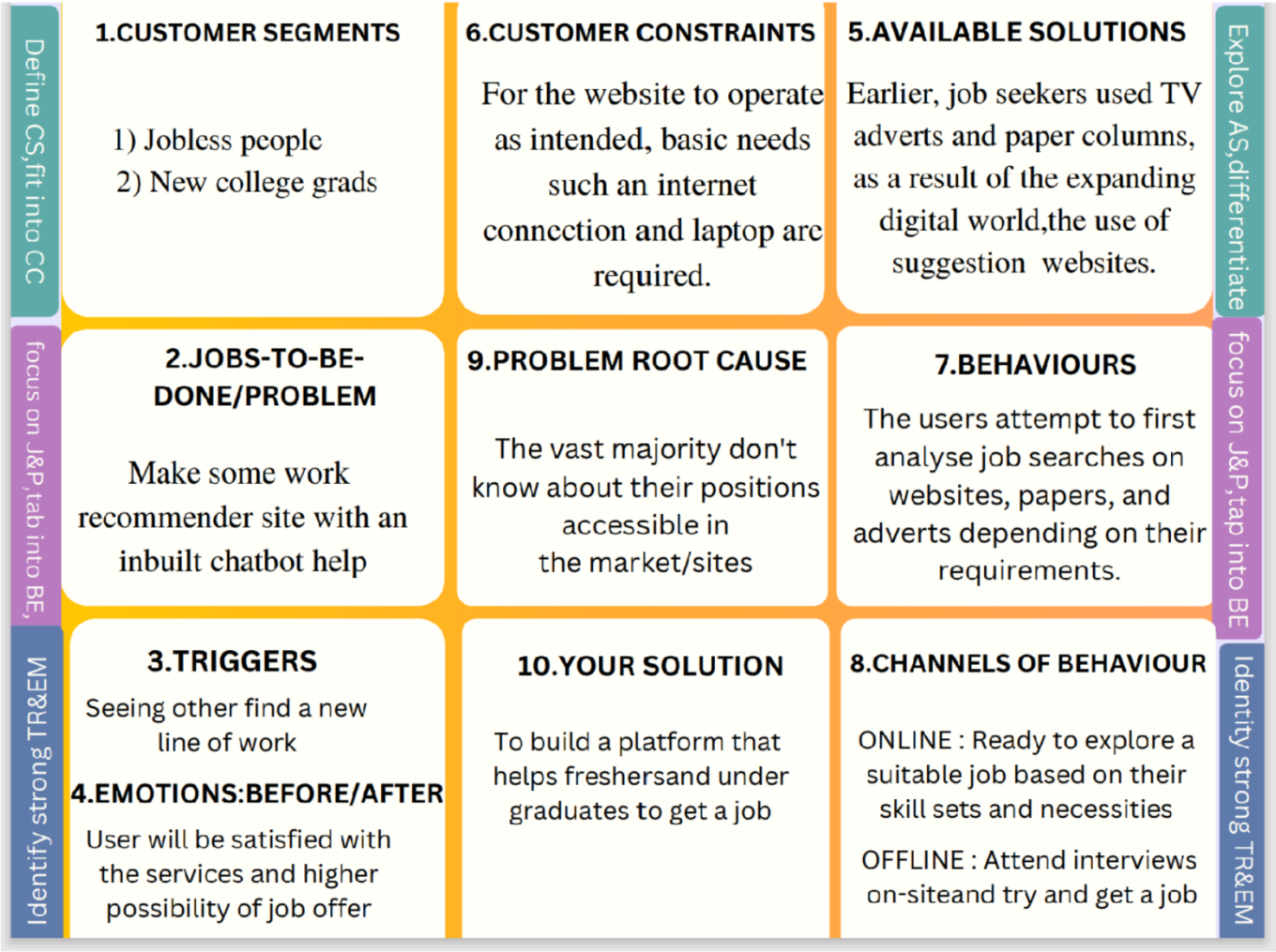
S.No.	Parameter	Description

1.	ProblemStatement(Problemtobesolved)	Having a variety of abilities but unsure of which position would be ideal for you? You shouldn't worry! We have developed a skill-recommender system that allows both novice and experienced users to log in, search for employment, or speak with a chatbot directly to land their ideal job. To build a complete web application that can show available jobs based on the user's skill set. The database contains the user's data as well. Alert users when an opening arises depending on their skill set. Users can obtain suggestions based on their talents by interacting with the chatbot. To obtain the most recent job vacancies on the market, we may utilize a job search API that will immediately pull data from a website.
2.	Idea/Solutiondescription	The three contributions of this study are as follows: i) We made a new dataset, composed of a collection of job seeker profiles and a set of job openings, publicly available. ii) present a framework for employment recommendations based on job seekers' professional skills iii) undertook an evaluation to gauge the recommendability of two cutting-edge approaches inside the suggested framework while taking various configurations into account. As a result, we provide a broad overview of the job suggestion assignment with the intention of facilitating research and real-world application creation addressing this significant subject.
3.	Novelty/Uniqueness	According to each person's abilities, the optimal position is recommended. It should be noted that there are typically many positions that are advised for a given set of talents, even when the position of known profiles is assumed.

		The most likely positions should be returned by a recommendation system, and any of them can be equally legitimate. Our recommendation strategy is based on finding the jobs with the most comparable vectors for each profile for both positions and profiles.
4.	Social Impact / CustomerSatisfaction	Students will profit because they will learn which jobs fit them best depending on their skill set, which will help to minimize unemployment.
5.	Business Model(RevenueModel)	We can offer job seekers an application on a subscription basis, share the profiles with businesses, and generate revenue by giving them the best profiles.
6.	ScalabilityoftheSolution	Data can be scaled up or scaled down depending on the number of available job openings right now.

3.4 Problem Solution Fit

Template:



4. REQUIREMENT ANALYSIS

4.1 Function Requirement

Software Required:

Python, Flask, Docker

4.2 Non-Function Requirement

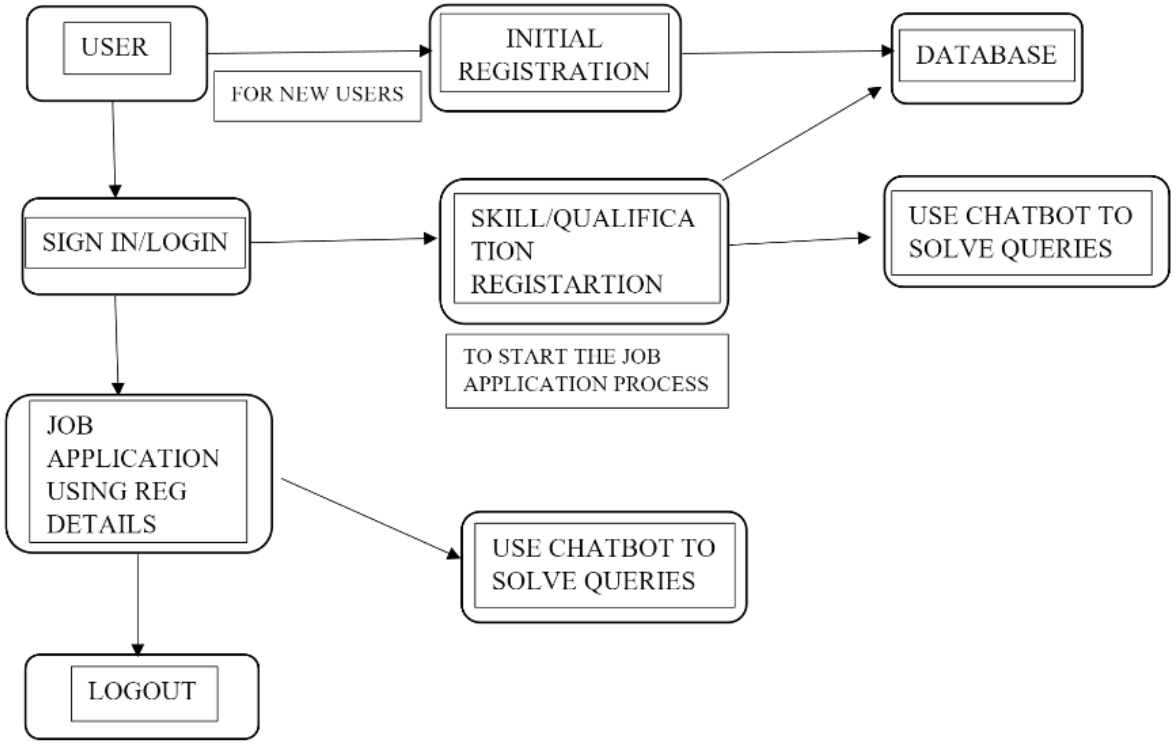
System Required:

8GB RAM, Intel Core i3, OS- Windows/Linux/MAC

,Laptop or Desktop

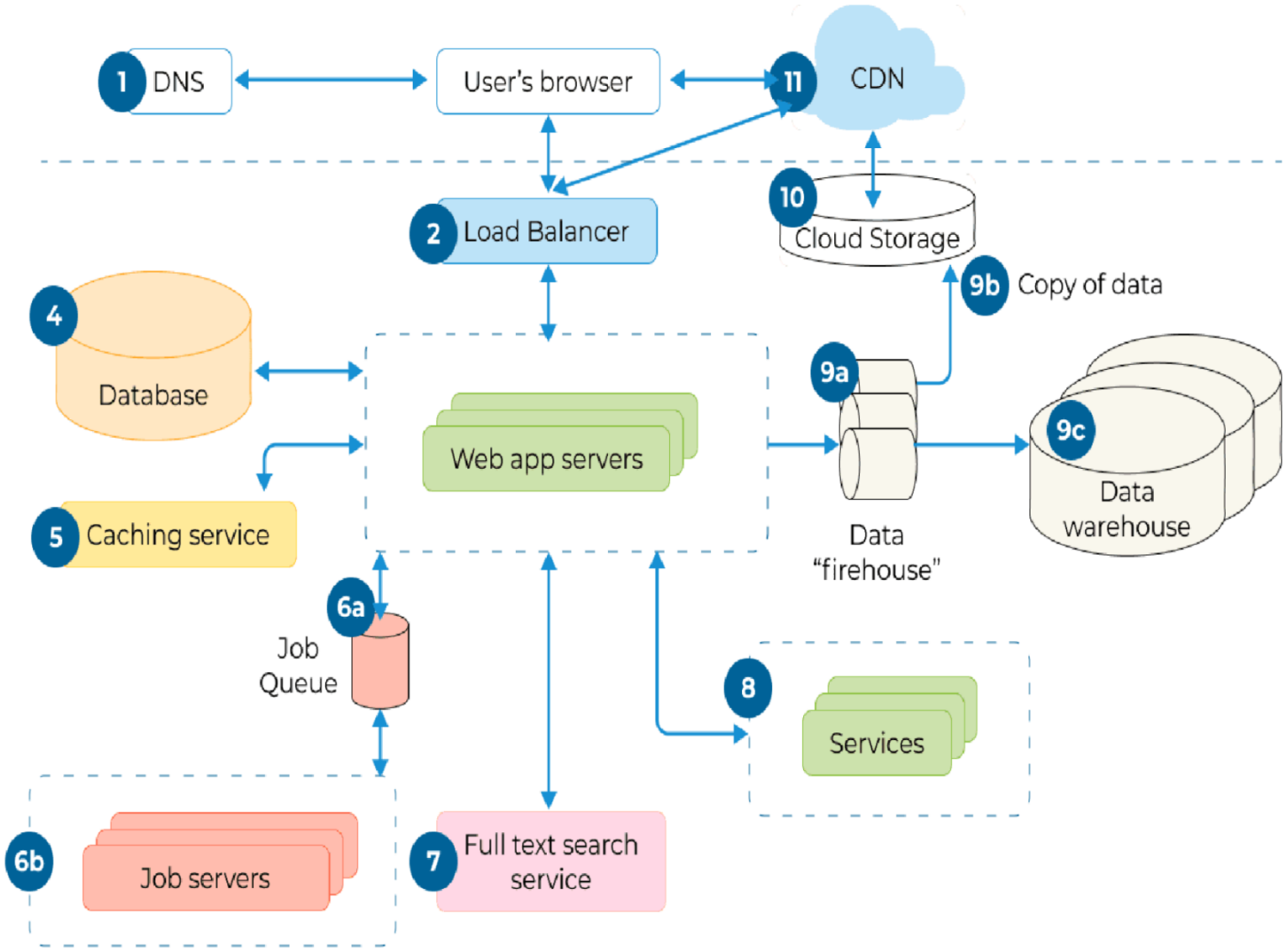
5. PROJECT DESIGN

5.1 Data Flow Diagrams

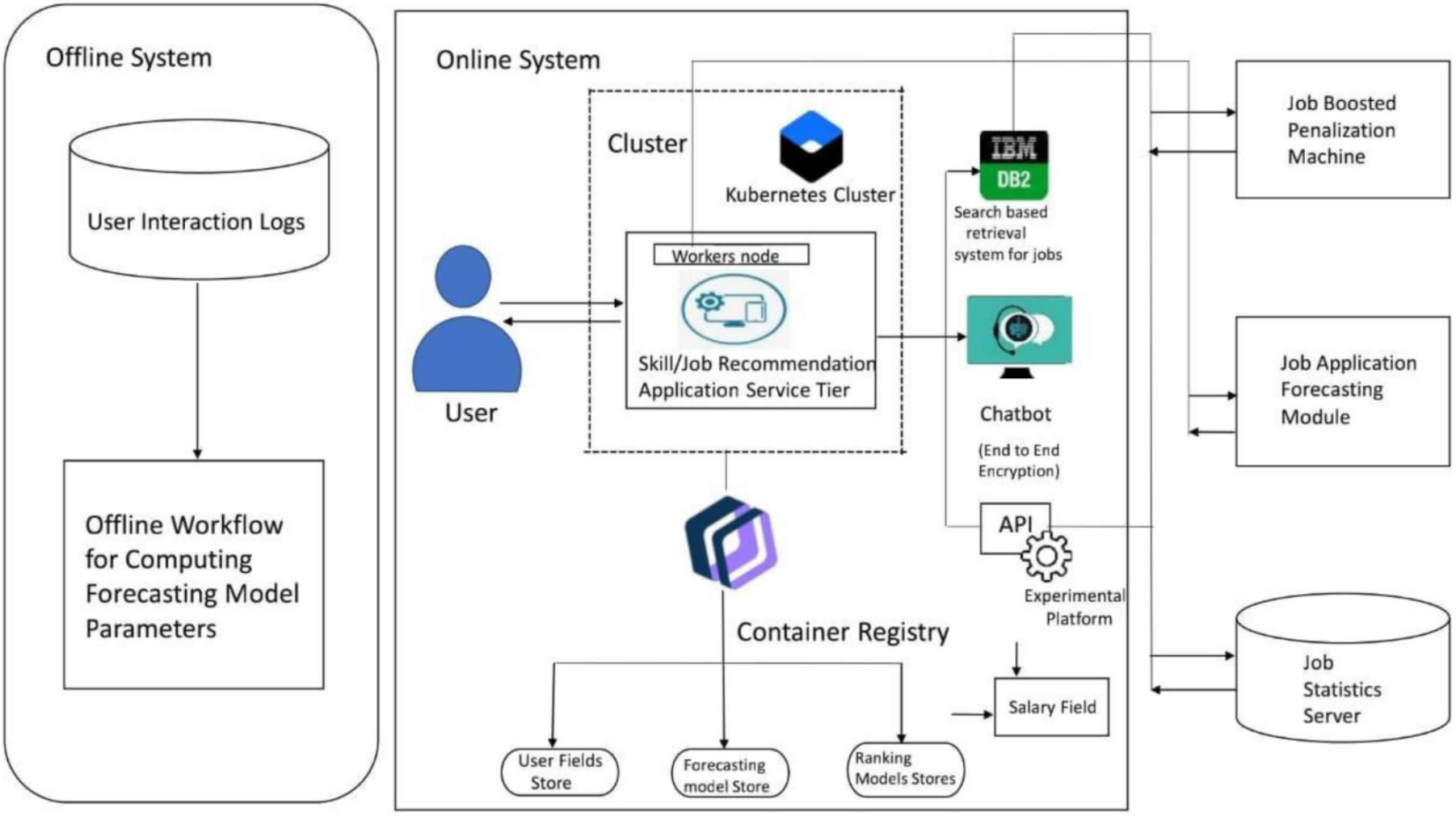


5.2 Solution & Technical Architecture

SOLUTION ARCHITECTURE



TECHNICAL ARCHITECTURE



5.3 User stories:

User Stories :

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will 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 & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can receive confirmation email & click confirm	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my account / dashboard	High	Sprint-1
	Dashboard	USN-6	Create a model set that contains those models, then assign it to a role.	Assign that group to the appropriate roles on the Roles page	High	Sprint-1
Customer (Web user)	Identity-Aware	USN-7	Open, public access, User-authenticated access, Employee-restricted access.	Company public website. App running on the company intranet. App with access to customer private information.	High	Sprint-1
Customer Care Executive	Communication	USN-8	A customer care executive is a professional responsible for communicating the how's and why's regarding service expectations within a company.	For how to tackle customer queries.	Medium	Sprint-1
Administrator	Device management	USN-9	You can Delete/Disable/Enable devices in Azure Active Directory but you cannot Add/Remove Users in the directory.	Ease of use.	Medium	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	FunctionalRequirement(Epic)	UserStoryNumber	UserStory/Task	Priority	Acceptancecriteria	TeamMembers
Sprint-1	UIDesign	USN-1	Asauser,Icanseeandexperienceanawesomeuser interfaceinthewebsite	Medium	BetterImpressionaboutawebsite	Damodarram K
Sprint-1	Registration	USN-2	As a user, I can register for theapplicationbyenteringmyemail,password, and confirming mypassword.	High	Icanaccessmyaccount / dashboard	Damodarram K
Sprint-1		USN-3	Asauser, Iwillreceiveconfirmationemail once I have registered for theapplication	High	Icanreceiveconfirmationemail& clickconfirm	Damodarram K
Sprint-1		USN-4	Asauser,Icanregisterfortheapplicationthrough Facebook	Low	I can register & access thedashboardwithFacebookLogin	Damodarram K
Sprint-1		USN-5	Asauser,Icanregisterfortheapplication throughGmail	Medium	Icanreceiveconfirmationemail& clickconfirm	Damodarram K
Sprint-1	Login	USN-6	Asauser,Icanlog intotheapplicationbyenteringemail& password	High	Icanaccessmy account/ dashboard	Damodarram K

Sprint-!	Flask	USN-7	Asauser, Ican accessthe websiteinasecond	High	Icanaccessmyaccount / dashboard	Damodarram K

Sprint	**FunctionalRequirement(Epic)**	**User Story Number**	**UserStory/Task**	**Priority**	**Acceptancecriteria**	**TeamMembers**
Sprint-1	Dashboard	USN-8	Asauser,IfILogged incorrectly,Icanview my dashboard and I can navigateto any pages which are already listedthere.	High	Icanaccessallthepages/dashboard	Damodarram K
			SubmissionOfSprint-1			
Sprint-2	UserProfile	USN-9	Asauser,Icanviewand updatemydetails	Medium	Icanmodifymydetails/data	SusilaraniK
Sprint-2	Database	USN-10	Asauser, Icanstoremydetailsanddata inthewebsitew	Medium	Icanstore mydata	SusilaraniK
Sprint-2	CloudStorage	USN-11	As a user, I can upload my photo,resumeandmuchmoreinthewebsite.	Medium	IcanUploadmydocumentsanddetails	SusilaraniK
Sprint-2	Chatbot	USN-12	Asauser,IcanasktheChatbot aboutlatest job openings, which will help meand show the recent job openingsbased onmyprofile	High	Icanknow therecentjobopenings	SusilaraniK

Sprint-2	Identity-Aware	USN-13	As a User, I can access my account byentering by correct login credentials. Myusercredentialsisonlydisplayedtome.	High	Icanhavemyaccountsafely	SusilaraniK
			SubmissionofSprint-2			

Sprint	**FunctionalRequirement(Epic)**	**UserStoryNumber**	**UserStory/Task**	**Priority**	**Acceptancecriteria**	**TeamMembers**
Sprint-3	Sendgridservice	USN-14	Asauser, Ican getanotificationormailabout a job opening with the help ofsendgridservice.	Medium	Icangetanotificationinasecond.	Priyadharshini R
Sprint-3	LearningResource	USN-15	As a user, I can learn the course and Iwillattaintheskillswhichwillbe usefulfordevelopingmytechnical skills.	High	Icangaintheknowledgeandskills	Priyadharshini R
Sprint-3	Docker	USN-16	Asauser, Icanaccessthewebsiteinanydevice	High	Icanaccessmyaccountin anydevice	Priyadharshini R
Sprint-3	Kubernates	USN-17	Asauser,Icanaccessthewebsiteinany device	High	Icanaccessmyaccountin anydevice	Priyadharshini R
Sprint-3	Deploymentincloud	USN-18	Asauser,Icanaccessthewebsiteinany device	High	Icanaccessmyaccountin anydevice	Priyadharshini R
Sprint-3	Technicalsupport	USN-19	Asauser,Icanget acustomercaresupport from the website which willsolvemy queries.	Medium	Icantacklemyproblem & queries.	Priyadharshini R

			SubmissionofSprint-3			
Sprint-4	Unit Testing	USN-15	Asauser,Icanaccessthewebsitewith out any interruption	High	Icanaccessthewebsitewithoutan yinterruption	Abinaya TKanakavalli M
Sprint-4	Integrationtesting	USN-16	Asauser,Icanaccessthewebsitewith out any interruption	High	Icanaccessthewebsitewithoutan yinterruption	Abinaya TKanakavalli M

Sprint	**FunctionalRequire ment(Epic)**	**User Story Number**	**UserStory/Task**	**Priority**	**Acceptancecriteria**	**TeamMembers**
Sprint-4	System testing	USN-17	Asauser,Icanaccessthewebsitewith out any interruption	High	Icanaccessthewebsitewithoutan yinterruption	Abinaya TKanakavalli M
Sprint-4	Correction	USN-18	Asauser,Icanaccessthewebsitewith out any interruption	High	Icanaccessthewebsitewithoutan yinterruption	Abinaya TKanakavalli M
Sprint-4	Acceptancetesting	USN-19	Asauser,Icanaccessthewebsitewith out any interruption	High	Icanaccessthewebsitewithoutan yinterruption	Abinaya TKanakavalli M
			SubmissionofSprint-4			

6.2 SprintDeliveryplanning:

ProjectTracker,Velocity&BurndownChart:

Sprint	Total StoryPoints	Duration	SprintStartDate	SprintEnd Date(Planned)	Story PointsCompleted (as onPlannedEndDate)
Sprint-1	20	6Days	24Oct2022	29Oct2022	20
Sprint-2	20	6Days	31Oct2022	05Nov2022	20
Sprint-3	20	6Days	07Nov2022	12Nov2022	20
Sprint-4	20	6Days	14Nov2022	19Nov2022	20

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 averagevelocity(AV) per iteration unit (storypointsper day)

6.3 Report from JIRA

$$AV_{sprintduration} \quad 202_{velocity} \quad 10$$

7. CODING & SOLUTIONING

7.1 Feature 1

Registration page

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style> body{ font-family: Calibri,
Helvetica, sans-serif; background-color:
pink;
}
.container {
    padding: 50px;
  background-color: lightblue;
}

input[type=text], input[type=password], textarea {
  width: 100%; padding: 15px; margin: 5px 0 22px
  0; display: inline-block; border: none;
  background: #f1f1f1;
}
input[type=text]:focus, input[type=password]:focus
  { background-color: orange; outline: none;
} div
 {
            padding: 10px 0;
         } hr { border: 1px
solid #f1f1f1; margin-
bottom: 25px;
}
.registerbtn { background-
  color: #4CAF50; color:
  white; padding: 16px 20px;
  margin: 8px 0; border:
  none; cursor: pointer;
  width: 100%; opacity: 0.9;
```

```
}
.registerbtn:hover {
  opacity: 1;
}
</style>
</head>
<body>
<form action="file:///D:/Skill%20Job%20Recommender/login.html?username=admin&password=PSW">
  <div class="container">
  <center> <h1> Student Registeration Form</h1> </center>
  <hr>
  <label> Firstname </label>
<input type="text" name="firstname" placeholder= "Firstname" size="15" required />
<label> Middlename: </label>
<input type="text" name="middlename" placeholder="Middlename" size="15" required />
<label> Lastname: </label>
<input type="text" name="lastname" placeholder="Lastname" size="15"required />
<div>
<label>
Course :
</label>

<select>
<option value="Course">Course</option>
<option value="BCA">BCA</option>
<option value="BBA">BBA</option>
<option value="B.Tech\B.E">B.Tech/B.E</option>
<option value="MBA">MBA</option>
<option value="MCA">MCA</option>
<option value="M.Tech">M.Tech</option>
</select>
</div>
<div>
<label>
Gender :
</label><br>
<input type="radio" value="Male" name="gender" checked > Male
<input type="radio" value="Female" name="gender"> Female
<input type="radio" value="Other" name="gender"> Other

</div>
<label>
Phone :
</label>
<input type="text" name="country code" placeholder="Country Code" value="+91" size="2"/>
<input type="text" name="phone" placeholder="phone no." size="10"/ required> Current
Address :
<textarea cols="80" rows="5" placeholder="Current Address" value="address" required>
</textarea>
 <label for="email"><b>Email</b></label>
 <input type="text" placeholder="Enter Email" name="email" required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw" required>

    <label for="psw-repeat"><b>Re-type Password</b></label>
    <input type="password" placeholder="Retype Password" name="psw-repeat" required>
    <button type="submit" class="registerbtn">Register</button>
</form>
```

```
</body>
</html>
```

Login.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<title> Login Page </title>
<style>
Body { font-family: Calibri, Helvetica, sans-serif;
  background-color: pink;
} button { background-color: #4CAF50; width:
100%;
        color: orange; padding: 15px;
        margin: 10px 0px; border: none;
        cursor: pointer;
         } form { border: 3px solid #f1f1f1;
    } input[type=text], input[type=password] {
        width: 100%; margin: 8px 0; padding:
        12px 20px; display: inline-block;
        border: 2px solid green; box-sizing:
        border-box;
    } button:hover { opacity:
 0.7;
    }
  .cancelbtn { width: auto; padding:
    10px 18px; margin: 10px 5px; }


 .container { padding: 25px;
        background-color: lightblue; }
</style>
</head> <body>
    <center> <h1> Student Login Form </h1> </center> <form>
        <div class="container">
            <label>Username : </label>
            <input type="text" placeholder="Enter Username" name="username" required>
            <label>Password : </label>
            <input type="password" placeholder="Enter Password" name="password" required>
            <button type="submit">Login</button>
            <input type="checkbox" checked="checked"> Remember me
            <button type="button" class="cancelbtn"> Cancel</button> Forgot <a
            href="#"> password? </a>
        </div>
```

```
    </form>
</body>
</html>
```

7.2 Feature 2

```
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="btnbtn-ghost normal-case text-xl"
        to={user ? "/dashboard" : "/"}
>
        F-ing Jobs
</Link>
</div>
     {user && (
<div className="flex-none gap-2">
<div className="dropdown dropdown-end">
<label tabIndex={0} className="btnbtn-ghost btn-circle avatar ">
<div className="w-10 rounded-full ring ring-opacity-50 ring-purple-700">
<imgsrc="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>
 );
};

export default Navbar;
```

CHATBOT :

Chatbot has been implemented to provide assistance .

```
window.watsonAssistantChatOptions = { integrationID: "d73273d3-3f44-430484ee-
8fd243016d1d", // The ID of this integration.
 region: "jp-tok",
 // The region your integration is hosted in.
                serviceInstanceID: "81229104-ee6b-46daac1c-67ede110663a", // The
                ID of your service instance.
                        onLoad: function(instance) {
```

```
        instance.render(); }
    };
      setTimeout(function(){
 const t=document.createElement('script');
t.src="https://webchat.global.assistant.watson.appdo
    main. cloud/versions/" +
    (window.watsonAssistantChatOptions.clientVersio
    n || 'latest') +
    "/WatsonAssistantChatEntry.js";
document.head.appendChild(t);
  });
```

7.3 Database Schema(if Applicable) :

```
# using SendGrid's Python Library
# https://github.com/sendgrid/sendgrid-python
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

# from_address we pass to our Mail object, edit with your name
FROM_EMAIL = 'Your_Name@SendGridTest.com'



def SendEmail(to_email):
    """ Send an email to the provided email addresses


    :param to_email = email to be sent to
:returns API response code
    :raises Exception e: raises an exception """
    message = Mail(
from_email=FROM_EMAIL,
to_emails=to_email,
        subject='A Test from SendGrid!',
```

```
html_content='<strong>Hello there from SendGrid your URL is: ' +
      '<a href=''https://github.com/cyberjive''>right here!</a></strong>')
   try:
      sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
      response = sg.send(message)
      code, body, headers = response.status_code, response.body,
response.headers
print(f"Response Code: {code} ")
print(f"Response Body: {body} ")
print(f"Response Headers: {headers} ")
print("Message Sent!")
   except Exception as e:
print("Error: {0}".format(e))
   return str(response.status_code)



if __name__ == "__main__":
SendEmail(to_email=input("Email address to send to? "))
```

8. TESTING

8.2 User Acceptance Testing

➢ Purpose of Document

This document's goal is to succinctly explain the test coverage and unresolved problems of the Skills/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.

Section	Total Cases	Not Tested
Print Engine	7	0
Client Application	5	0
Security	3	0
Outsource Shipping	7	0

Resolution	Severity 1	Severity 2	Severity 3	Severity 4
By Design	3	2	1	1
Duplicate	1	0	2	0
External	2	0	0	1
Fixed	5	2	5	7
Not Reproduced	0	0	1	0
Skipped	0	0	0	1
Won't Fix	0	5	1	1
Totals	11	9	10	11

1. **Test Case Analysis**

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

Exception Reporting	6	0	0	6
Final Report Output	3	0	0	3
Version Control	2	0	0	2

9. RESULTS

As anticipated, the job was successfully finished. We made sure the database was created and link were obtained.

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Based on their skill set, a person looking for work may quickly locate a suited position.
- Attending an eligibility test allows a person to confirm their eligibility.
- The majority of recruiters choose the right candidate based on the results they received on the eligibility tests.

DISADVANTAGES

- Person Job May get technical difficulty while taking theeligibility
- Job seeker may have trouble to contact recruitersdirectly.

11. CONCLUSION

The programme was created to make the job hunting process simpler. We created an application that is easy to use. Based on their skill set, a user can quickly obtain employment. The use of this application benefits the jobseeker without a doubt. Additionally, Chatbot was put into use with IBM Whatson's assistance. When organizations and job seekers run into problems, the chatbot offers assistance.

12. FUTURE SCOPE

The popular application's linked in feature helps users find work and maintain professional connections. Employers and job seekers both use LinkedIn to locate employment. There are many opportunities to improve our project in the future that are comparable to those of LinkedIn.

13. APPENDIX

SOURCE CODE

__init__.py

```
from dotenv import dotenv_values
from flask import Flask
from flask_cors import CORS
import ibm_db

# Get the environment variables
config = dotenv_values("backend/.env")

# Connect to db
try:
   # conn = 'dd'
   conn = ibm_db.pconnect(
      f"DATABASE={config['DB2_DATABASE']};HOSTNAME={config['DB2_HOSTNAME']};
PORT={config['DB2_PORT']};SECURITY=SSL; SSLServerCertificate=backend/
DigiCertGlobalRootCA.crt;UID={config['DB2_USERNAME']};
PWD={config['DB2_PASSWORD']}", '', '')
print("Connected to IBM_DB2 successfully!!")
   print(conn)
except:
print("Failed to connect to Database!")


def create_app():
   # Tell flask to use the build directory of react to serve static content
   app = Flask(__name__, static_folder='../build', static_url_path='/')

CORS(app)

   # Set the secret key for flask
app.config['SECRET_KEY'] = config['APP_SECRET']
```

```
    # Import and register auth_router
from .auth_router import auth
app.register_blueprint(auth, url_prefix='/api/auth')

from .files_router import files
app.register_blueprint(files, url_prefix='/api/files')

    from .user_router import user
app.register_blueprint(user, url_prefix='/api/user')

    # In production serve the index.html page at root

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

    return app
```

auth_middleware.py

```
from functools import wraps
import jwt
from flask import request
from backend import conn, config
import ibm_db

# Middleware function that checks for JWT token in header
# All routes that have the @token_required decorator will be protected


def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = None
        if "Authorization" in request.headers:
            token = request.headers["Authorization"].split(" ")[1]
        if not token:
            return {
                "error": "Unauthorized"
            }, 401
```

```
        try:
            # Get the user's email from the decoded token
            data = jwt.decode(
                token, config["APP_SECRET"], algorithms=["HS256"])

            # Retreive user's info from the database
sql = f"select * from users where email='{data['email']}'"
stmt = ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)
current_user = ibm_db.fetch_assoc(stmt)

            # If user does not exist throw error.
            if current_user is None:
                return {
                    "error": "Unauthorized"
                }, 401
        except Exception as e:
            return {
                "error": str(e)
            }, 500

        # Pass the authorized user in function args.
        return f(current_user, *args, **kwargs)

    return decorated
```

auth_router.py

```
from flask import Blueprint, jsonify, request
from backend import conn, config
import bcrypt
import jwt
import ibm_db

auth = Blueprint("auth", __name__)

LOGIN_FEILDS = ('email', 'password')
SIGNUP_FEILDS = ('name', 'email', 'phone_number', 'password')
```

```
@auth.route("/login", methods=['POST'])
def login_user():
   # Check if all the required feild are present
   for feild in LOGIN_FEILDS:
      if not (feild in request.json):
         return jsonify({"error": f"Allfeilds are required!"}), 409
   email = request.json['email']
   password = request.json['password']
sql = f"select * from users where email='{email}'"
stmt = ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)
   user = ibm_db.fetch_assoc(stmt)
   if not user:
      return jsonify({"error": "Invalid credentials!"}), 401
   if bcrypt.checkpw(password.encode('utf-8'),
                  user["PASSWORD"].encode('utf-8')):
      token = jwt.encode(
         {"email": email},
         config["APP_SECRET"],
         algorithm="HS256"
      )
      return jsonify({"name": user["NAME"], "email": email, "phone_number":
user["PHONE_NUMBER"], "token": token}), 200
   else:
      return jsonify({"error": "Invalid credentials!"}), 401


@auth.route("/signup", methods=['POST'])
def register_user():
   # Check if all the required feild are present
   for feild in SIGNUP_FEILDS:
      if not (feild in request.json):
         return jsonify({"error": f"Allfeilds are required!"}), 409

   email = request.json['email']
phone_number = request.json['phone_number']
   name = request.json['name']
   password = request.json['password']

   # Sqlstmt to check if email/number is already in use
```

```
sql = f"select * from users where email='{email}' or phone_number='{phone_number}'"
stmt = ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)
   user = ibm_db.fetch_assoc(stmt)
   if user:
      return jsonify({"error": f"Email/Phone number is alread in use!"}), 409

   # If user does not exist, then create account
hashed_password = bcrypt.hashpw(
password.encode('utf-8'), bcrypt.gensalt())
sql = f"insert into users(name,email,phone_number,password) values('{name}','{email}','{phone_number}',?)"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, hashed_password)
ibm_db.execute(stmt)
   token = jwt.encode(
      {"email": email},
      config["APP_SECRET"],
      algorithm="HS256"
   )
   return jsonify({"name": name, "email": email, "phone_number": phone_number, "token": token}), 200
```

files_router.py

```
from flask import Blueprint
from backend.auth_middleware import token_required
import ibm_boto3
from ibm_botocore.client import Config, ClientError
from backend import config

cos = ibm_boto3.resource("s3",
ibm_api_key_id=config["COS_API_KEY_ID"],
ibm_service_instance_id=config["COS_INSTANCE_CRN"],
                  config=Config(signature_version="oauth"),
endpoint_url=config["COS_ENDPOINT"]
                  )

files = Blueprint("files", __name__)
```

```
def multi_part_upload(bucket_name, item_name, file_path):
   try:
print("Starting file transfer for {0} to bucket: {1}\n".format(
item_name, bucket_name))
      # set 5 MB chunks
part_size = 1024 * 1024 * 5

      # setthreadhold to 15 MB
file_threshold = 1024 * 1024 * 15

      # set the transfer threshold and chunk size
transfer_config = ibm_boto3.s3.transfer.TransferConfig(
multipart_threshold=file_threshold,
multipart_chunksize=part_size
      )

      # theupload_fileobj method will automatically execute a multi-part upload
      # in 5 MB chunks for all files over 15 MB
      with open(file_path, "rb") as file_data:
cos.Object(bucket_name, item_name).upload_fileobj(
Fileobj=file_data,
            Config=transfer_config
         )

print("Transfer for {0} Complete!\n".format(item_name))
   except ClientError as be:
print("CLIENT ERROR: {0}\n".format(be))
   except Exception as e:
print("Unable to complete multi-part upload: {0}".format(e))


@files.route('/avatar', methods=["POST"])
@token_required
def upload_profile_photo(current_user):
   return "hello"
```

user_router.py

```
from flask import Blueprint, jsonify, request
from backend import conn
```

```
from backend.auth_middleware import token_required
import ibm_db

user = Blueprint("user", __name__)


@user.route("/skills", methods=["GET", "POST", "DELETE"])
@token_required
def manage_skills(current_user):
   # Get user_id of current user
user_id = current_user['USER_ID']

   # Handle GET request
   if request.method == 'GET':
      skills = []

sql = f"select name from skills where user_id={user_id}"
stmt = ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)
dict = ibm_db.fetch_assoc(stmt)

      # Iterate over all the results and append skills to the array
      while dict != False:
skills.append(dict['NAME'])
dict = ibm_db.fetch_assoc(stmt)

      return jsonify({"skills": skills}), 200

   # Get the skills from the request
   if not ('skills' in request.json):
      return jsonify({"error": f"Allfeilds are required!"}), 409

   skills = request.json['skills']

   # If no skills are provided then return empty array
   if skills == []:
      return jsonify({"skills": []}), 200

   # Handle POST request
   if request.method == "POST":
      # Prepare the SQL statement to insert multiple rows
```

```
        values = ''
        for i in range(len(skills)):
            if i == 0:
                values += 'values'
            values += f"('{skills[i]}',{user_id})"
            if i != len(skills)-1:
                values += ','
sql = f"insert into skills(name,user_id) {values}"
stmt = ibm_db.prepare(conn, sql)
        status = ibm_db.execute(stmt)

        if status:
            return jsonify({"message": "Updated skills successfully!"}), 200
        else:
jsonify({"error": "Something went wrong!!"}), 409

    # Handle DELETE request
    if request.method == 'DELETE':
        values = ""
        for i in range(len(skills)):
            values += f"'{skills[i]}'"
            if i != len(skills)-1:
                values += ','
sql = f"delete from skills where name in ({values})"
stmt = ibm_db.prepare(conn, sql)
        status = ibm_db.execute(stmt)
        if status:
            return jsonify({"message": "Deleted skills successfully!"}), 200
        else:
jsonify({"error": "Something went wrong!!"}), 409
```

avatar.svg

```
<svg width="480" height="480" fill="none"
xmlns="http://www.w3.org/2000/svg"><rect opacity=".1" width="480"
height="480" rx="32" fill="#fff"/><path d="M374.308 240c0 71.691-58.117
129.808-129.808 129.808S114.692 311.691 114.692 240 172.809 110.192
244.5 110.192 374.308 168.309 374.308 240z" fill="#F6F6F6" stroke="#fff"
stroke-width="10.385"/><path fill-rule="evenodd" clip-rule="evenodd"
d="M244.5 256.2c-21.627 0-64.8 10.854-64.8 32.4v16.2h129.6v-16.2c0-
21.546-43.173-32.4-64.8-32.4m0-16.2c17.901 0 32.4-14.499 32.4-32.4 0-
```

17.901-14.499-32.4-32.4-32.4-17.901 0-32.4 14.499-32.432.4 0 17.901 14.499 32.4 32.4 32.4" fill="#35374A" opacity=".3"/></svg>

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-semiboldhover: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 React, { useContext, useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { AppContext } from "../context/AppContext";
import { loginUser } from "../proxies/backend_api";
import { emailRegex } from "../utils/helper";

const Login = () => {
  const { setShowAlert, 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) {
setShowAlert({ type: "error", message: data.error, duration: 3000 });
        return;
      }
setUser(data);
setShowAlert({
        type: "success",
        message: `Welcome back ${data.name}`,
        duration: 3000,
      });
localStorage.setItem("user", JSON.stringify(data));
      navigate("/dashboard");
    }
  };

  return (
<div className="flex flex-col justify-center items-center gap-10 mt-5">
<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>
<imgsrc={`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="btnbtn-smbtn-primary mb-4"
>
            Login
</button>
<p>
            Don't have an account?{" "}
<Link className="text-blue-400" to="/signup">
              Sign up
</Link>
</p>
</div>
</form>
</div>
</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="btnbtn-ghost normal-case text-xl"
        to={user ? "/dashboard" : "/"}
>
        F-ing Jobs
</Link>
</div>
     {user && (
<div className="flex-none gap-2">
<div className="dropdown dropdown-end">
<label tabIndex={0} className="btnbtn-ghost btn-circle avatar ">
<div className="w-10 rounded-full ring ring-opacity-50 ring-purple-700">
<imgsrc="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>
  );
};

export default Navbar;
```

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 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 { 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("/profile");
  }
 };

 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>
<imgsrc={`github-dark.png`} alt="github" width="14%" />
</button>
<div className="divider max-w-xs">or</div>
<div 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"
classNameName="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="text"
            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="btnbtn-smbtn-primary mb-4"
>
            Sign Up
</button>
</div>
</div>
</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;
```

AppContext.jsx

```
import { createContext, useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";

export const AppContext = createContext();

export const AppProvider = ({ children }) => {
  const navigate = useNavigate();

  const [skills, setSkills] = useState([]);

  const [user, setUser] = useState(null);

useEffect(() => {
    let temp_user = JSON.parse(localStorage.getItem("user"));
    if (!temp_user) {
navigate("/");
    } else {
setUser(temp_user);
    }
  }, []);

  return (
<AppContext.Provider value={{ user, setUser, skills, setSkills }}>
      {children}
</AppContext.Provider>
  );
};
```

backend_api.js

```
import { BASE_URL } from "../utils/helper";

export const loginUser = async (inputs) => {
  try {
    const response = await fetch(`${BASE_URL}/auth/login`, {
      method: "POST",
      body: JSON.stringify(inputs),
      headers: {
        "Content-Type": "application/json",
      },
    });
    const data = await response.json();
    return data;
  } catch (error) {
console.error(error);
  }
};

export const registerUser = async (inputs) => {
  try {
    const response = await fetch(`${BASE_URL}/auth/signup`, {
      method: "POST",
      body: JSON.stringify(inputs),
      headers: {
        "Content-Type": "application/json",
      },
    });
    const data = await response.json();
```

```
    return data;
  } catch (error) {
console.error(error);
  }
};
```

Profile.jsx

```
import {
  Progress,
SkeletonCircle,
SkeletonText,
  Spinner,
useToast,
} from "@chakra-ui/react";
import React, { useContext, useEffect, useState } from "react";
import { AiOutlineClose } from "react-icons/ai";
import { BsLinkedin } from "react-icons/bs";
import { GoMarkGithub } from "react-icons/go";
import { MdDeleteForever } from "react-icons/md";
import { RiEdit2Fill } from "react-icons/ri";
import { TfiTwitterAlt } from "react-icons/tfi";
import { VscAdd } from "react-icons/vsc";
import { AppContext } from "../context/AppContext";
import {
getUserSkills,
removeUserSkills,
saveUserSkills,
updateUserDetails,
} from "../proxies/backend_api";
```

```
const Profile = () => {
  const toast = useToast();

  const { user, setUser, skills, setSkills } = useContext(AppContext);

  const [addSkill, setAddSkill] = useState("");

  const [newSkills, setNewSkills] = useState([]);

  const [removedSkills, setRemovedSkills] = useState([]);

  const [isEditingEnabled, setIsEditingEnabled] = useState(false);

  const [loading, setLoading] = useState(false);

  const [userInfo, setUserInfo] = useState({
    name: "",
phone_number: "",
  });

  const handleUserInfoChange = ({ target: { name, value } }) => {
setUserInfo((prev) =>({ ...prev, [name]: value }));
  };

  const changeSkills = () => {
    if (
addSkill !== "" &&
!skills.find((item) =>item.toLowerCase() === addSkill.toLowerCase())
```

```
    ) {
setNewSkills((prev) => [...prev, addSkill.trim()]);
setSkills((prev) => [...prev, addSkill.trim()]);
    }
setAddSkill("");
  };

  const removeSkills = (skill_name) => {
setRemovedSkills((prev) => [...prev, skill_name]);

setSkills((prev) =>prev.filter((item) => item !== skill_name));

setNewSkills((prev) =>prev.filter((item) => item !== skill_name));
  };

  const updateSkills = async () => {
setLoading(true);

    let skillsAdded = false,
skillsRemoved = false;

    if (newSkills.length !== 0) {
skillsAdded = await saveUserSkills(newSkills, user.token);
    }

    if (removeSkills.length !== 0) {
skillsRemoved = await removeUserSkills(removedSkills, user.token);
    }
```

```
   if (skillsAdded || skillsRemoved) {
toast({
       title: "Profile Updated!",
       status: "info",
       duration: 3000,
isClosable: true,
       variant: "left-accent",
       position: "top",
     });
   }

setNewSkills([]);

setRemovedSkills([]);

setLoading(false);
  };

  const updateUserInfo = async () => {
setLoading(true);

   const data = await updateUserDetails(userInfo, user.token);

   if (data) {
setUser((prev) => {
prev = { ...prev, name: data.name, phone_number: data.phone_number };

localStorage.setItem("user", JSON.stringify(prev));
```

```
        return prev;
      });

toast({
        title: "Profile Updated!",
        status: "info",
        duration: 3000,
isClosable: true,
        variant: "left-accent",
        position: "top",
      });
    }

setLoading(false);

setIsEditingEnabled(false);
  };

useEffect(() => {
    if (user) {
      (async () => {
setLoading(true);

        let data = await getUserSkills(user?.token);

        if (data) setSkills(data);

setLoading(false);
      })();
```

```
setUserInfo({
       name: user.name,
phone_number: user.phone_number,
     });
   }
  }, [user]);

  return (
<>
    {loading &&<Progress size="xs" isIndeterminatecolorScheme={"purple"} />}
<div className="my-5 mx-10">
<div className="border-2 border-blue-100 w-full h-fit rounded-xl p-5 flex flex-col gap-3">
<div className="flex justify-between w-full min-h-[25vh]">
<div className="flex flex-col justify-between">
<h1 className="md:text-2xl  text-xl font-medium flex items-center gap-4">
            Your Profile{" "}
<button>
              {isEditingEnabled ? (
<AiOutlineClose
                 color="#ff8977"
onClick={() =>setIsEditingEnabled(!isEditingEnabled)}
                />
              ) : (
<RiEdit2Fill
                 color="#4506cb"
onClick={() =>setIsEditingEnabled(!isEditingEnabled)}
                />
```

```
            )}
</button>
</h1>
<div className="flex flex-col gap-3">
            {isEditingEnabled ? (
<>
<input
                name="name"
                value={userInfo.name}
className="input input-bordered w-full input-xs p-3 text-lg input-primary"
                type="text"
                placeholder="name"
onChange={handleUserInfoChange}
              />
<input
                disabled
                value={user?.email}
className="input input-bordered w-full input-xs p-3 text-lg input-primary"
                type="text"
                placeholder="name"
              />
<input
                name="phone_number"
                value={userInfo.phone_number}
className="input input-bordered w-full input-xs p-3 text-lg input-primary"
                type="number"
                placeholder="phone number"
onChange={handleUserInfoChange}
              />
```

```
<button
className="btnbtn-xsbtn-outline btn-primary"
onClick={updateUserInfo}
>
                Update
</button>
</>
            ) : (
<>
<h2 className="md:text-2xl xl:text-2xl sm:text-xl">
                {user?.name}
</h2>
<p className="md:text-xlsm:text-md text-gray-700">
                {user?.email}
</p>
<span className="text-gray-700">{user?.phone_number}</span>
</>
            )}
</div>
</div>
<div className="flex flex-col justify-end w-fit gap-4">
<img
src="avatar.webp"
            alt="profile"
className="md:w-36 w-20 rounded-md object-contain"
           />
</div>
</div>
<div className="divider my-2"></div>
```

```
<div className="flex flex-col">
<div className="flex justify-between gap-2 flex-col">
<h4 className="text-xl">Skills</h4>
<form
className="flex gap-5 items-center"
onSubmit={(e) =>e.preventDefault()}
>
<input
autoComplete="off"
            value={addSkill}
            type="text"
            name="addSkill"
            placeholder="Add skills"
onChange={(e) =>setAddSkill(e.target.value)}
className="input input-bordered w-full input-primary max-w-xl input-sm"
          />

<button
className="hover:rotate-90 transition-all"
onClick={changeSkills}
>
<VscAdd size={20} />
</button>
</form>
        {loading ? (
<Spinner
            thickness="3px"
            speed="0.65s"
emptyColor="gray.200"
```

```
            color="blue.500"
            size="md"
className="m-3"
          />
        ) : (
<ulclassName="flex gap-2 flex-wrap">
            {skills?.map((addSkill, ind) => (
<li
className="bg-indigo-100 rounded p-2 flex gap-2 items-center"
              key={ind}
>
              {addSkill}
<MdDeleteForever
               color="#ff8977"
onClick={() =>removeSkills(addSkill)}
               size={20}
              />
</li>
          ))}
</ul>
        )}

<button
className="btnbtn-sm w-fit btn-primary"
          type="button"
onClick={updateSkills}
>
          Save
</button>
```

```
</div>
<div className="divider my-2"></div>
<div className="flex justify-between gap-2 flex-col">
<h4 className="text-xl">Resume/Portfolio</h4>
<div className="flex gap-5">
<input
className="input input-bordered w-full input-primary max-w-xl input-sm"
              type="text"
              placeholder="paste the link"
             />
<button className="btnbtn-primary btn-sm">update</button>
</div>
</div>
<div className="divider my-2"></div>
<div className="flex gap-2 flex-col">
<h3 className="text-xl">Socials</h3>
<div className="flex flex-col gap-2">
<div className="flex gap-5 items-center">
<GoMarkGithub size={20} />
<input
               type="text"
               placeholder="paste the link"
className="border-2 border-gray-300 rounded-md px-3 my-1 max-w-md"
             />
</div>
<div className="flex gap-5 items-center">
<BsLinkedin size={20} />
<input
               type="text"
```

```
                placeholder="paste the link"
className="border-2 border-gray-300 rounded-md px-3 my-1 max-w-md"
              />
</div>
<div className="flex gap-5 items-center">
<TfiTwitterAlt size={20} />
<input
                type="text"
                placeholder="paste the link"
className="border-2 border-gray-300 rounded-md px-3 my-1 max-w-md"
              />
</div>
<button className="btnbtn-primary btn-sm max-w-fit">
              save
</button>
</div>
</div>
</div>
</div>
</div>
</>
  );
};

export default Profile;
```

Dashboard.jsx

```
import {
```

```
  Progress,
SkeletonCircle,
SkeletonText,
  Spinner,
} from "@chakra-ui/react";
import axios from "axios";
import React, { useContext, useEffect, useState } from "react";
import JobCard from "../components/JobCard";
import SearchBar from "../components/SearchBar";
import Skill from "../components/Skill";
import { AppContext } from "../context/AppContext";
import { getUserSkills } from "../proxies/backend_api";

const Dashboard = () => {
  const { user, skills, setSkills } = useContext(AppContext);

  const [selectedSkills, setSelectedSkills] = useState([]);

  const [skillsLoading, setSkillsLoading] = useState(false);

  const [jobsLoading, setJobsLoading] = useState(false);

  const [query, setquery] = useState("");

  const [posts, setPosts] = useState(null);

  const id = import.meta.env.VITE_ADZUNA_API_ID;

  const key = import.meta.env.VITE_ADZUNA_API_KEY;
```

```
  const baseURL_with_skills =
`http://api.adzuna.com/v1/api/jobs/in/search/1?app_id=${id}&app_key=${key
}&results_per_page=15&what=${query}&what_and=${selectedSkills.join(
    " "
)}&&content-type=application/json`;


  const baseURL =
`http://api.adzuna.com/v1/api/jobs/in/search/1?app_id=${id}&app_key=${key
}&results_per_page=15&what=${query}&content-type=application/json`;


  const searchJobsFromQuery = async () => {
setJobsLoading(true);


    if (query !== "" || !posts) {
      const { data } = await axios.get(baseURL);
setPosts(data.results);
    }


setJobsLoading(false);
  };


  const searchWithSkills = async () => {
setJobsLoading(true);


    const { data } = await axios.get(baseURL_with_skills);


setPosts(data.results);


setJobsLoading(false);
```

```
  };

useEffect(() => {
    if (user) {
      (async () => {
setSkillsLoading(true);
setSkills(await getUserSkills(user.token));
setSkillsLoading(false);
      })();
    }
  }, [user]);

useEffect(() => {
searchWithSkills();
  }, [selectedSkills]);

useEffect(() => {
searchJobsFromQuery();
  }, []);

  return (
<>
     {(jobsLoading || skillsLoading) && (
<Progress size="xs" isIndeterminatecolorScheme={"purple"} />
     )}
<div className="flex gap-10 m-10">
<div className="hidden lg:flex bg-purple-600 w-1/5 p-5 h-3/6 rounded-lg text-center flex-col gap-4">
<div className="text-2xl text-white capitalize font-extrabold">
```

```
          Your skills
</div>
        {skillsLoading ? (
<Spinner
className="self-center my-5"
            thickness="3px"
            speed="0.65s"
emptyColor="gray.200"
            color="black.100"
            size="lg"
          />
        ) : (
<ulclassName="list-none text-gray-200 flex flex-col gap-2">
            {skills?.length === 0 ? (
<p className="text-gray-300">
              Skills you add in the profile section will appear here!!
</p>
            ) : (
skills.map((skill, ind) => (
<Skill
                skill={skill}
                key={ind}
setSelectedSkills={setSelectedSkills}
                disabled={skillsLoading}
              />
            ))
            )}
</ul>
        )}
```

```
<p className="text-white text-sm">
          (Include your skills in the search result)
</p>
</div>

<div className="mx-auto min-w-[80%] ">
<SearchBarsetquery={setquery} onClick={searchJobsFromQuery} />
        {query === "" ? (
<h2 className="text-2xl mt-5">Recommended Jobs</h2>
        ) : (
<h2 className="text-2xl mt-5">
            Search for keywords {query}
            {filterUsingSkills&&`,${skills.join(",")}`}
</h2>
        )}

<div className="mt-5 grid grid-cols-1 lg:grid-cols-3 md:grid-cols-2 gap-5">
          {jobsLoading
            ? [...new Array(10)].map((_, ind) => (
<div key={ind}>
<SkeletonCircle size="8" className="mb-5" />
<SkeletonText
                  mt="4"
noOfLines={8}
                  spacing="4"
                  color={"red"}
                />
</div>
              ))
```

```
          : posts?.map((post, ind) => (
<JobCard
                key={ind}
                title={post.title}
                company={post.company.display_name}
                description={post.description}
                link={post.redirect_url}
               />
             ))}
</div>
</div>
</div>
</>
  );
};

export default Dashboard;
```

Auth.jsx

```
import { Tab, TabList, TabPanel, TabPanels, Tabs } from "@chakra-ui/react";
import React, { useContext, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import Login from "../components/Login";
import SignUp from "../components/Signup";
import { AppContext } from "../context/AppContext";

const Auth = () => {
  const navigate = useNavigate();
```

```
  const { user } = useContext(AppContext);

useEffect(() => {
    if (user) navigate("dashboard");
  }, []);

  return (
<div className="flex flex-col justify-center items-center gap-10 mt-5">
<Tabs isFitted variant="line" colorScheme={"purple"}>
<TabList mb="1em">
<Tab>Login</Tab>
<Tab>SignUp</Tab>
</TabList>
<TabPanels>
<TabPanel>
<Login />
</TabPanel>
<TabPanel>
<SignUp />
</TabPanel>
</TabPanels>
</Tabs>
</div>
  );
};

export default Auth;
```

helper.js

```
export const emailRegex = /^[\w-.]+@([\w-]+\.)+[\w-]{2,4}$/;


export const urlRegex =
  /((([A-Za-z]{3,9}:(?:\/\/)?)(?:[-;:&=\+\$,\w]+@)?[A-Za-z0-9.-]+(:[0-
9]+)?|(?:www.|[-;:&=\+\$,\w]+@)[A-Za-z0-9.-]+)((?:\/[\+~%\/.\w-_]*)?\??(?:[-
\+=&;%@.\w_]*)#?(?:[\w]*))?)/;


export const BASE_URL = import.meta.env.VITE_BACKEND_ENDPOINT;
```

App.jsx

```
import { useEffect } from "react";
import { HashRouter, Route, Routes } from "react-router-dom";
import Navbar from "./components/Navbar";
import { AppProvider } from "./context/AppContext";
import Auth from "./screens/Auth";
import Dashboard from "./screens/Dashboard";
import Profile from "./screens/Profile";


function App() {
useEffect(() => {
window.watsonAssistantChatOptions = {
integrationID: import.meta.env.VITE_WATSON_INTEGRATION_ID, // The ID of
this integration.
    region: import.meta.env.VITE_WATSON_REGION, // The region your
integration is hosted in.
serviceInstanceID: import.meta.env.VITE_WATSON_SERVICE_INSTANCE_ID, //
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);
    });
  }, []);
  return (
<HashRouter>
<AppProvider>
<Navbar />
<Routes>
<Route path="/" element={<Auth />} />
<Route path="/dashboard" element={<Dashboard />} />
<Route path="/profile" element={<Profile />} />
</Routes>
</AppProvider>
</HashRouter>
  );
}

export default App;
```

main.jsx

```
import { ChakraProvider } from "@chakra-ui/react";
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import "./index.css";

ReactDOM.createRoot(document.getElementById("root")).render(
<React.StrictMode>
<ChakraProvider>
<App />
</ChakraProvider>
</React.StrictMode>
);
```

Index.css

```
@import
url("https://fonts.googleapis.com/css2?family=Ubuntu&display=swap");

@tailwind base;
@tailwind components;
@tailwind utilities;

:root {
  font-family: Inter, Avenir, Helvetica, Arial, sans-serif;
  font-size: 16px;
  line-height: 24px;
  font-weight: 400;
```

```
  color-scheme: light;
  /* color: rgba(255, 255, 255, 0.87);
  background-color: #242424; */

  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  -webkit-text-size-adjust: 100%;
}

* {
  margin: 0;
  padding: 0;
  font-family: "Ubuntu", sans-serif;
}

body::-webkit-scrollbar {
  width: 5px;
  background-color: none;
  border-radius: 20px;
}

body::-webkit-scrollbar-thumb {
  background-color: #adadad;
  border-radius: 20px;
}

body {
```

```
  max-height: 100vh;
}
```

Deployment.yaml

```
## Enter your <docker_username> before use

apiVersion: v1
kind: Service
metadata:
  name: test
  labels:
    app: test
spec:
  type: NodePort
  ports:
    - port: 5000
      name: http
nodePort: 30080
  selector:
    app: app
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: test
spec:
  replicas: 1
  template:
```

```
  metadata:
    labels:
      app: app
  spec:
    containers:
      - name: ibm_project
image:https://github.com/IBM-EPBL/IBM-Project-3989-1658678612
        ports:
          - containerPort: 5000
imagePullSecrets:
      - name: regcred
```

main.py

```
from backend import create_app

app = create_app()

if __name__ == '__main__':
  from waitress import serve
serve(app, port=5000)
```

package.json

```
{
 "name": "react-flask-app",
 "private": true,
 "version": "0.0.0",
```

```
 "type": "module",
 "scripts": {
  "start": "vite",
  "build": "vite build",
  "preview": "vite preview",
  "server": "cd backend && flask --debug run"
 },
 "dependencies": {
  "axios": "^1.1.3",
  "daisyui": "^2.33.0",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-icons": "^4.6.0",
  "react-router-dom": "^6.4.2"
 },
 "devDependencies": {
  "@types/react": "^18.0.17",
  "@types/react-dom": "^18.0.6",
  "@vitejs/plugin-react": "^2.1.0",
  "autoprefixer": "^10.4.12",
  "postcss": "^8.4.18",
  "tailwindcss": "^3.1.8",
  "vite": "^3.1.0"
 } }
```

postcss.config.cjs

```
module.exports = {
 plugins: {
tailwindcss: {},
autoprefixer: {},
```

```
  },
}
```

tailwind.config.cjs

```
/** @type {import('tailwindcss').Config} */
module.exports = {
darkMode: "class",
  content: ["./index.html", "./src/**/*.{js,ts,jsx,tsx}"],
  theme: {
    extend: {},
  },
  plugins: [require("daisyui")],
daisyui: {
    themes: ["light"],
  },
};
```

vite.config.js

```
import react from "@vitejs/plugin-react";
import { defineConfig } from "vite";

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  server: {
    port: 3000,
cors: false,
  },
});
```

Dockerfile

```
# Build step #1: build the React front end
FROM node:16-alpine as react-builder
WORKDIR /app
ENV PATH /app/node_modules/.bin:$PATH
COPY package.json ./
COPY ./src ./src
COPY ./public ./public
COPY ./index.html ./vite.config.js ./postcss.config.cjs./tailwind.config.cjs ./.env
./
RUN npm install
RUN npm run build

# Build step #2: build the API with the client as static files
FROM python:3.10
WORKDIR /app
COPY --from=react-builder /app/dist ./dist
COPY main.py ./main.py

RUN mkdir ./backend
COPY backend/ ./backend/
RUN pip install -r ./backend/requirements.txt

EXPOSE 5000
ENTRYPOINT ["python","main.py"]
```

GITHUB & PROJECT DEMO LINK:

All the tasks of developing the application were uploaded

on the github.

The github has been uploaded below.

https://github.com/IBM-EPBL/IBM-Project-34631-1660240516

Project DemoLink:

https://drive.google.com/file/d/1PeY_AnXuEbwrNCDw8L038jI_LLRqBm1Q/view?usp=drivesdk