SKILL / JOB RECOMMENDER APPLICATION

A PROJECT REPORT

Submitted by

TEAM ID **PNT2022TMID32090**

Team Lead	**Manojkumar. E**	**(731619205031)**
Team Member 1	**Barathkumar. S**	**(731619205004)**
Team Member 2	**Mythreyan. S**	**(731619205032)**
Team Member 3	**Sanjay. M**	**(731619205044)**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KSR INSTITUTE FOR ENGINEERING AND TECHNOLOGY,

TIRUCHENGODE

ANNA UNIVERSITY : CHENNAI 600 025

TABLE OF CONTENTS

TITLE

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Having lots of skills but wondering which job will best suit you? Don’t need to worry! We have come up with a skill recommender solution through which the fresher or the skilled person can log in and find the jobs by using the search option or they can directly interact with the chatbot and get their dream job.

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.

This domain is the hiring process, where a job seeker applies to a job by creating a profile on a job portal by providing all his/her work preferences. These work preferences of each user can be collected from each user and provide job recommendations based on their preference. Data acquired for our study has no previous interaction between the user data and job listing data.

The Internet-based recruiting platforms become a primary recruitment channel in most companies. While such platforms decrease the recruitment time and advertisement cost, they suffer from an inappropriateness of traditional information retrieval techniques like the Boolean search methods. Consequently, a vast amount of candidates missed the opportunity of recruiting. In order to improve the e-recruiting functionality, many recommender system approaches have been proposed. This article will present a survey of e-recruiting process and existing recommendation approaches for building personalized recommender systems for candidates/job matching.

Based on the person-job fit premise, we propose a framework for job recommendation based on professional skills of job seekers. We automatically extracted the skills from the job seeker profiles using a variety of text processing techniques.

1.2 PURPOSE

Industries try finding ways to increase their revenue. In a classic business model, the upselling is the term used when a sales advisor tries to sell an item to a customer based on things that he is planning to purchase. As business started to integrate the technology to increase the user interaction with business, customers gave out their preferences by interacting or purchasing the product the business is selling to them. The business has utilized the collected big data to make a better-personalized recommendation to its customer base. Netflix is a streaming service that generates its revenue from customer subscriptions to its content. According to reports from business insider Australia, Netflix estimates 20% of their video watches are derived from the search. Whereas, 80% comes from its recommendation system.

This domain is the hiring process, where a job seeker applies to a job by creating a profile on a job portal by providing all his/her work preferences. These work preferences of each user can be collected from each user and provide job recommendations based on their preference. Data acquired for our study has no previous interaction between the user data and job listing data.

The Internet-based recruiting platforms become a primary recruitment channel in most companies. While such platforms decrease the recruitment time and advertisement cost, they suffer from an inappropriateness of traditional information retrieval techniques like the Boolean search methods. Consequently, a vast amount of candidates missed the opportunity of recruiting. In order to improve the e-recruiting functionality, many recommender system approaches have been proposed. This article will present a survey of e-recruiting process and existing recommendation approaches for building personalized recommender systems for candidates/job matching.

To develop an end-to-end web application capable of displaying the current job openings based on the user skillset. The user and their information are stored in the Database. An alert is sent when there is an opening based on the user skillset. Users will interact with the chatbot and can get the recommendations based on their skills. We can use a job search API to get the current job openings in the market which will fetch the data directly from the webpage.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

2.1.1 Job Recommendation based on Job Seeker Skills: An Empirical Study

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.

This article was published in March 2018.

Thus, the contributions of this work are,

a. 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.

b. Put forward the proposal of a framework for job recommendation based on professional skills of job seekers.[4]

2.1.2 Job Recommendation System Using Machine Learning and Natural Language Processing

The rise of digital communication and the spread of the internet has made an enormous impact in every industry. One such domain is the Hiring process, where a job seeker applies to a job by creating a profile on a job portal by providing all his/her work preferences. These work preferences of each user can be collected from each user and provide job recommendations based on their preference.

This domain is the Hiring process, where a job seeker applies to a job by creating a profile on a job portal by providing all his/her work preferences. These work preferences of each user can be collected from each user and provide job recommendations based on their preference. Data acquired for our study has no previous interaction between the user data and Job listing data. With such a dataset, we have addressed the issue of cold start from both User and Job perspective. This article was published in May 2020.[6]

2.1.3 A Survey of Job Recommender System

The internet-based recruiting platforms become a primary recruitment channel in most companies. While such platforms decrease the recruitment time and advertisement cost, they suffer from an inappropriateness of traditional information retrieval techniques like the Boolean search methods. Consequently, a vast amount of candidates missed the opportunity of recruiting. In order to improve the e-recruiting functionality, many recommender system approaches have been proposed. This article will present a survey of e-recruiting process and existing recommendation approaches for building personalized recommender systems for candidates/job matching. This article was published in July 2012.

2.2 REFERENCES

[1] W. Hong, S. Zheng, H. Wang, "Dynamic User Profile-Based Job Recommender System", 8th International Conference on Computer Science & Education (ICCSE), Colombo, pp. 1499- 1503, 26-28 Apr. 2013.

[2] W. Shalaby et al., "Help Me Find a Job: A Graph-based Approach for Job Recommendation at Scale," Dec. 2017.

[3] M. Hutterer, "Enhancing a Job Recommender with Implicit User Feedback," vol. 2011, 2011.

[4] Dr. Jorge Valverde-Rebaza, MsC. Ricardo Puma-Alvarez and MBA Nathalia C. Silva "Job Recommendation based on Seeker Skills: An Empirical Study", March 2018.

[5] S. T. Al-Otaibi and M. Ykhlef, "A survey of job recommender systems," Int. J. Phys. Sci., vol. 7, no. 29, pp. 5127–5142, Jul. 2012.

[6] Jeevankrishna, "Job Recommender System Using Machine Learning and Natural Language Processing", May 2020.

2.3 PROBLEM STATEMENT DEFINITION

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 recommend job vacancies that fit properly to the job seekers profiles. Thus, the contributions of this work are threefold, 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, put forward the proposal of a framework for job recommendation based on professional skills of job seekers, and carried out an evaluation to quantify empirically the recommendation abilities of two state-of-the-art methods, considering different configurations, within the proposed framework.

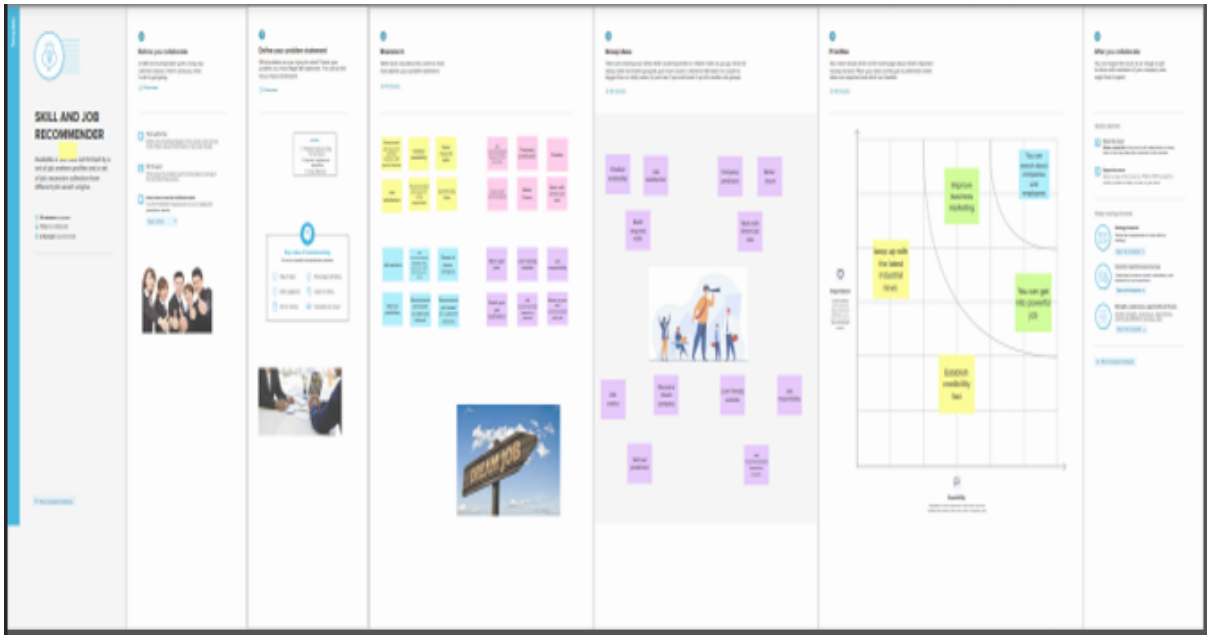
Thus present a general panorama of job recommendation task aiming to facilitate research and real-world application design regarding this important issue. Job matching, job seeking, job search, job recommender systems. Proposed a framework for job recommendation task. This framework facilitates the understanding of job recommendation process as well as it allows the use of a variety of text processing and recommendation methods according to the preferences of the job recommender system designer. Moreover, we also contribute making publicly available a new dataset containing job seekers profiles and job vacancies. Future directions of our work will focus on performing a more exhaustive evaluation considering a greater amount of methods and data as well as a comprehensive evaluation of the impact of each professional skill of a job seeker on the received job recommendation.

CHAPTER 3

IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

3.2 IDEATION AND BRAINSTORMING



3.3 PROPOSED SOLUTION

3.3.1 Problem statement (problem to be solved)

To develop an end-to-end web application capable of displaying the current job openings based on the user skillset.

3.3.2 Idea /solution description

In this application, we proposed a framework for job recommendation task. This framework facilities the understanding of job recommendation process. Moreover, we also contribute making publicly available a new dataset containing job seekers profiles and job vacancies.

3.3.3 Novelty/Uniqueness

Novelty is one of the important metrics of customer satisfaction. with the development of information technology and application of the internet, people gradually entered the time of information overload from information scarcity.

3.3.4 Social Impact / Customer Satisfaction

The result show that accuracy and diversity positively affect customer satisfaction when applying a deep learning-based recommender system. By contrast, only accuracy positively affects customer satisfaction when applying traditional recommender systems.

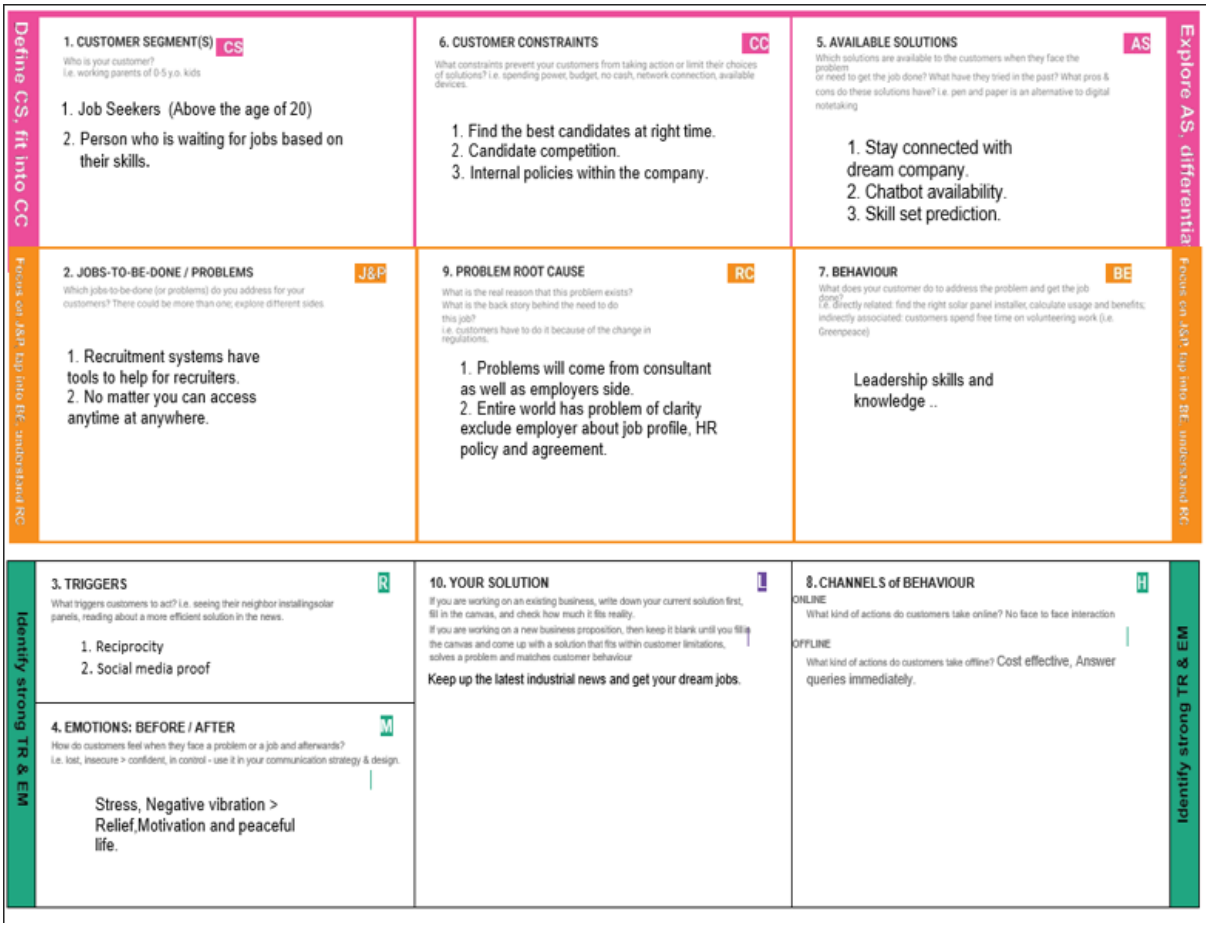
3.3.5 Business Model (Revenue Model)

Licensing, advertising, Affiliate company, know your skills, Filter your skills.

3.3.6 Scalability of the Solution

Recommendation system is a technique, which provides users with information, which he/she may be interested in or accessed in past. Traditional recommender techniques such as content and collaborative filtering used in various applications such as education, social media, marketing, entertainment, e-governance and many more.

3.4 PROBLEM SOLUTION FIT



CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
FR-1	User Registration	Register via email and password
FR-2	User Confirmation	Confirmation via EmailConfirmation via OTP verification code
FR-3	User Browse	Browse for learning and developing skills
FR-4	User post	Post for who are searching for job opportunities
FR-5	User learn	User can gain learning
FR-6	User recommendation	Recommendation as per user search

4.2 NON - FUNCTIONAL REQUIREMENT

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The job seeker can view the job role, job vacancy, salary expectations and also improve their skill.
NFR-2	Security	The job seeker and the recruiter can log in/register by using e-mail onlyand the verification code is sent to user.
NFR-3	Reliability	Reliability for the project is high.
NFR-4	Performance	The project has met its goal and objective clearly.
NFR-5	Availability	The resources are available.
NFR-6	Scalability	Scalability is high.

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

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

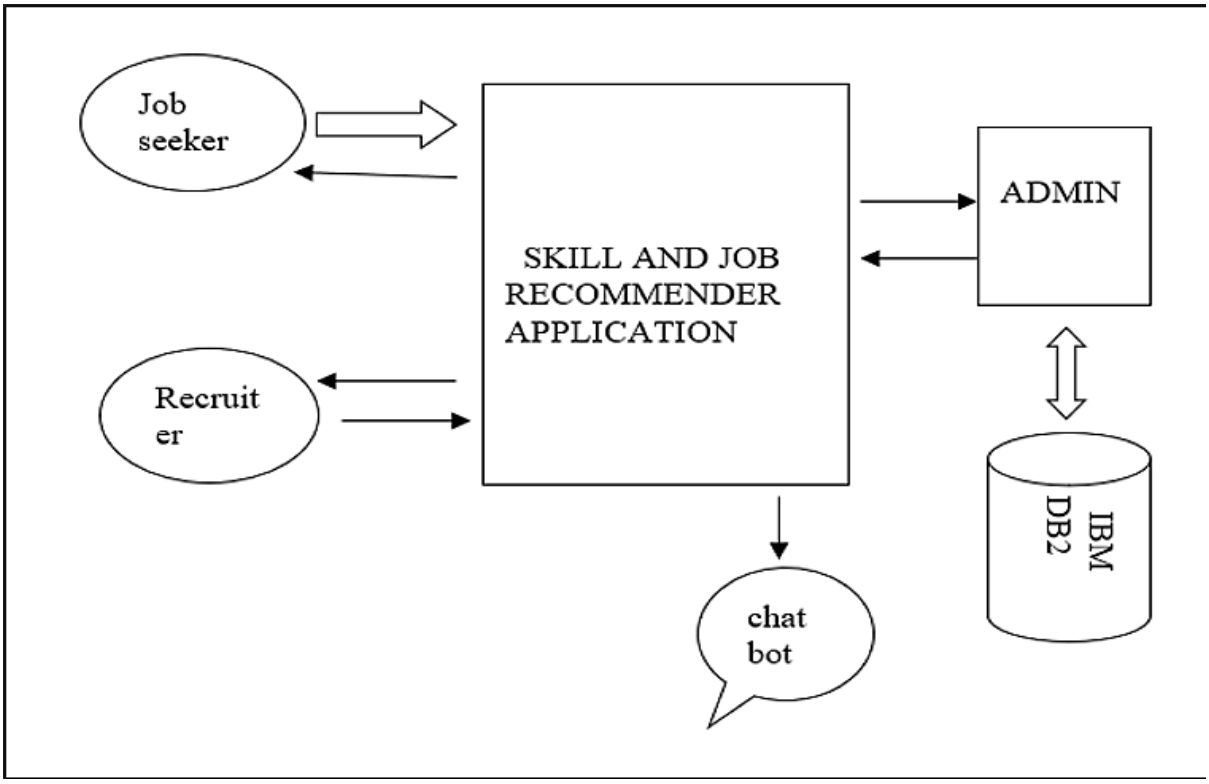


Fig 5.1.1 Data Flow Diagram Level 0 (Industry Standard)

5.2 SOLUTION AND TECHNICAL ARCHITECTURE

5.2.1 Technical Architecture

An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.

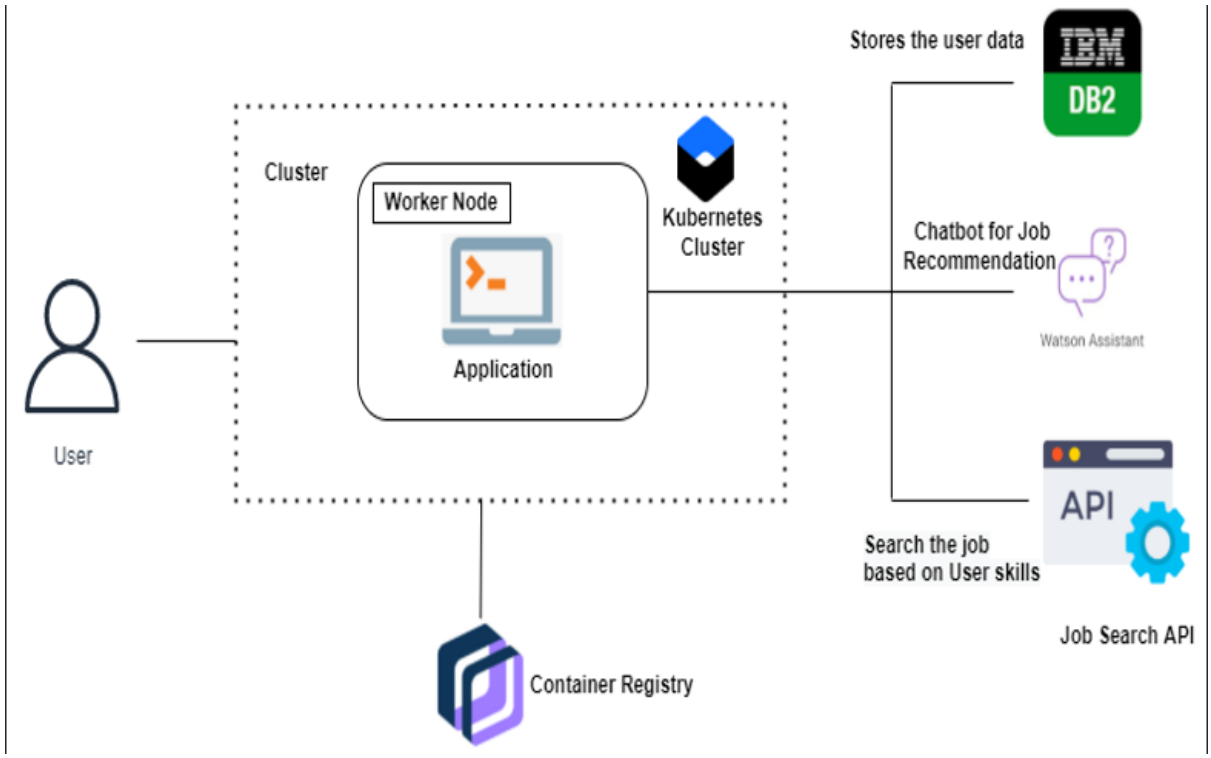


Fig 5.2.1 Technical Architecture Diagram

5.2.2 Solution Architecture

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.

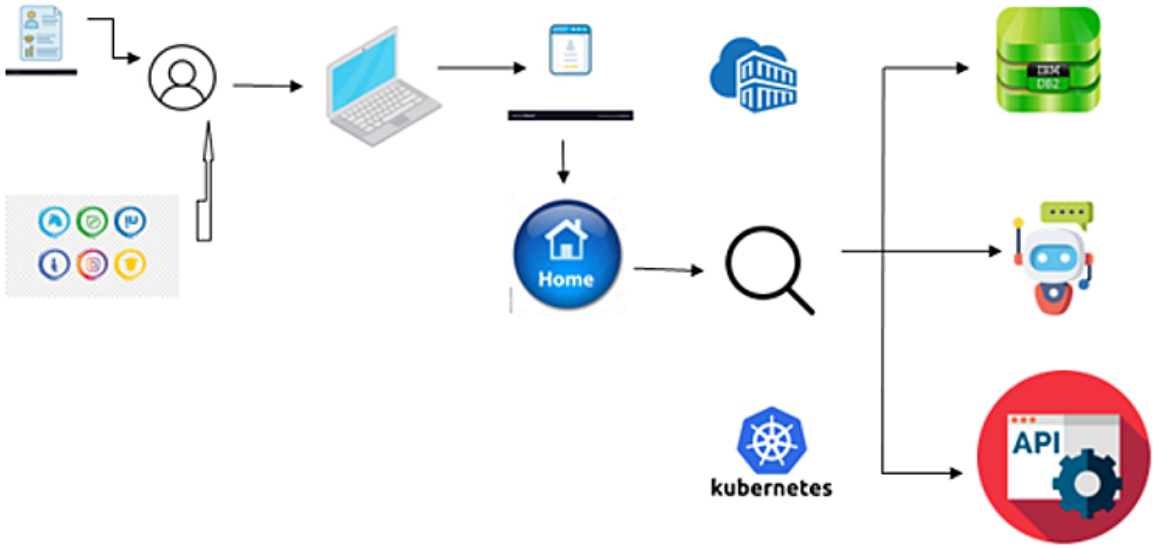


Fig 5.2.2 Solution Architecture Diagram

5.3 USER STORIES

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	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.		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password.		High	Sprint-1
	Dashboard	USN-5	As a user, I can access my dashboard after signing in.	I can access my account / dashboard	High	Sprint-1
Customer (Web user)	Access	USN-6	As a user, I can setup a profile, and basic details by signing in.			

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User interface	USN-1	As a user I can experience a good feel of the presentation	7	High	Manojkumar. E
Sprint-1	Home	USN-2	As a user, it will be the first page of the website	3	Medium	Mythreyan. S
Sprint-1	Database	USN-3	As a user my data will be stored in database for further use	4	High	Bharathkumar. S
Sprint-1	Registration	USN-4	As a user, I can register for the application as a Job seeker or Recruiter.	2	low	Sanjay. M
Sprint-1	Login	USN-5	As a user, I can log into the application as Jobseeker or Recruiter by entering registered username & correct password	2	low	Manojkumar. E
Sprint-2	Profile Setup	USN-6	As a fresh user I need to setup the profile	5	High	Manojkumar. E
Sprint-2		USN-7	As a fresh recruiter I need to setup profile for my company by filling required details	5	High	Mythreyan. S
Sprint-2	Cloud Storage	USN-8	As a user I can upload my Image, Resume and much more in the website	8	Medium	Sanjay. M
Sprint-2	Posting the jobs	USN-9	As a Recruiter I can post various job openings	2	High	Bharathkumar. S
Sprint-3	Job Listing	USN-10	As a user I can access jobs posted by recruiters and Google Job Search API	5	High	Sanjay. M, Bharathkumar. S
Sprint-3	Applying	USN-11	As a Job Seeker I can view all Job openings in the home page and also, I can search for specific jobs and apply for the same	2	High	Manojkumar. E

Sprint-3	Chatbot	USN-13	As a User I can access chatbot to avail any kind of guidance in the website	8	High	Mythreyan. S
Sprint-3	Notification (SendGrid)	USN-14	As a User, I can get notification on new Job openings via email using SendGrid service	5	Medium	Mythreyan. S
Sprint-4	System testing	USN-15	As a user I can access my website without any fault or malfunction	5	Medium	Sanjay. M
Sprint-4	Docker	USN-16	As a user I can access my containerized application in any device	5	High	Bharathkumar. S
Sprint-4	Kubernetes	USN-17	As a user I can access my containerized application in any device with greater security	5	High	Manojkumar. E
Sprint-4	Deployment in the Cloud	USN- 18	As a user I can access the website from anywhere in the world	5	High	Mythreyan. S

6.2 SPRINT DELIVERY SCHEDULE

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	18	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

CHAPTER 7

CODING AND SOLUTIONING

7.1 FEATURE 1

SendGrid is a cloud-based SMTP provider that allows you to send email without having to maintain email servers. SendGrid manages all of the technical details, from scaling the infrastructure to ISP outreach and reputation monitoring to whitelist services and real time analytics.

Verify your email sender in the SendGrid app and create an API key. Connect to SendGrid's SMTP server at smtp.sendgrid.net and use port 587. Authenticate against the SMTP server using "apikey" as the username and the actual API key as the password.

Code for sendgrid :

```
# using SendGrid's Python Library
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

message = Mail(
    from_email='from_email@example.com',
    to_emails='to@example.com',
    subject='Sending with Twilio SendGrid is Fun',
    html_content='<strong>and easy to do anywhere, even with Python</strong>')
try:
    sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sg.send(message)
    print(response.status_code)
    print(response.body)
    print(response.headers)
except Exception as e:
    print(e.message)
```

Fig 7.1 Using sendgrid’s python Library

7.2 FEATURE 2

Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime.

Docker is an open source platform that enables developers to build, deploy, run, update and manage containers—standardized, executable components that combine application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

```
# 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"]
```

7.3 FEATURE 3

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

```
<!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">
<title>Document</title>
</head>
<body>
<h1>My Chatbot</h1>
<blockquote>Click the bottom right corner to chat</blockquote>
<script>
window.watsonAssistantChatOptions = {
integrationID: "01ca5fe5-3f42-4a97-8965-332afedd97be", // The ID of this integration.
region:
"au-syd", // The region your integration is hosted in.
serviceInstanceID: "5683f375-e95c-4fa1-8471-5b76177675c2", // 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>
```

CHAPTER 8

TESTING

8.1 TEST CASES

A test case is a set of actions performed on a system to determine if it satisfies software requirements and functions correctly. The purpose of a test case is to determine if different features within a system are performing as expected and to confirm that the system satisfies all related standards, guidelines and customer requirements. The process of writing a test case can also help reveal errors or defects within the system.

Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
Verify that after registration users are navigated to login page	Mail id, Username, Password,Phone number, Pin	1. Open the website and go to register page. 2.Enter details and press register 3.Verify that users are navigated to registration page	https://drive.google.com/drive/folders/1OIftiud9BWX7m0v-iEyDMp41Nmv1KFs5	Users should be navigated to registration page	Working as expected	Pass	Excellent	`N		Manojkumar. E
Verify the UI elements in Login/Signup popup	Username & Password	1. Open the website 2.Enter details and press login 3.Verify that users are notified of login process	https://drive.google.com/drive/folders/1OkpV13F1nZ1ObSw2z9AIk8ELGsV9HRok	Users should be notified of login process	Not working	Fail	Trying To Recover	N	BUG-12	Mythreyan. S
Verify user is able to log into application with Valid credentials		1. Open the website 2.Enter details and press login 3.Verify that users are logged into website properly	Username: gr03122001@gmail.com password: 123	User should be logged into website properly	Working as expected	Pass	Good	N		Barathkumar. S
Verify that categories of skills and jobs are shown in homepage		1. Open the website 2.Enter details and press login 3.Verify that categories of are showing Jobs shown in homepage		Categories of skills and jobs should be shown in homepage	Working as expected	Partially pass	Will be solve	N	BUG-14	Sanjay. M
Verify that jobs are displayed in homepage		1. Open the website 2.Enter details and press login 3.Verify that jobs are displayed in homepage		jobs should be displayed in homepage	Working as expected	Pass	Good	N		Mythreyan. S
Verify that when clicked on jobs it is redirected to correct page		1. Open the website 2.Enter details and press login 3.Verify that when clicked on jobs it is redirected to correct page		When clicked on job link it should be redirected to correct page	Working as expected	Pass	Excellent	N		Manojkumar. E

Fig 8.1 Screenshot about the Testcases

8.2 USER ACCEPTANCE TESTING

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

8.2.1 DEFECT ANALYSIS

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	7

8.2.2 TEST CASE ANALYSIS

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

CHAPTER 9

RESULTS

9.1 PERFORMANCE METRICES

Performance metrics are defined as figures and data representative of an organization's actions, abilities, and overall quality. There are many different forms of performance metrics, including sales, profit, return on investment, customer happiness, customer reviews, personal reviews, overall quality, and reputation in a marketplace. Performance metrics can vary considerably when viewed through different industries.

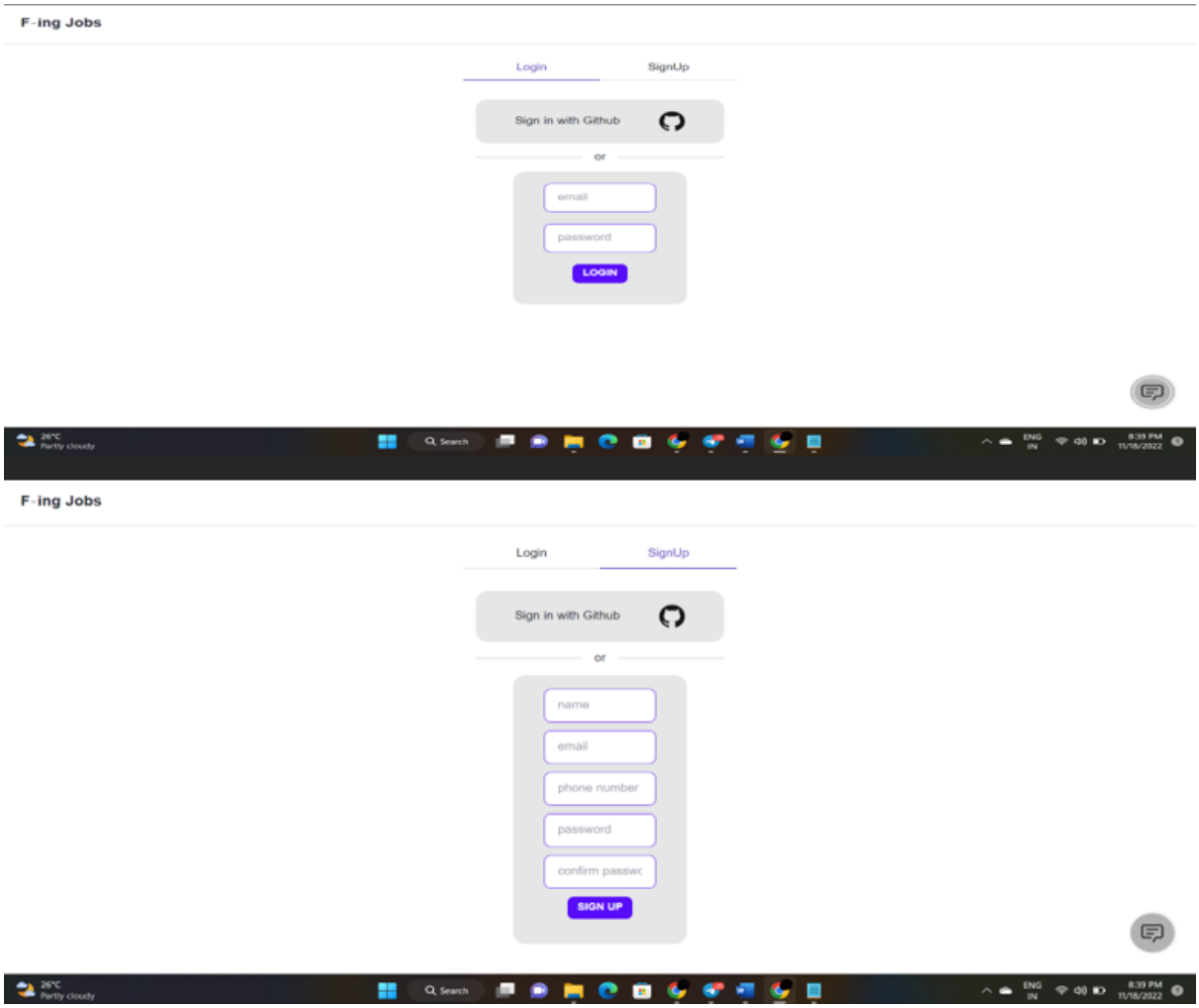


Fig 9.1 Login and Signup page of the application

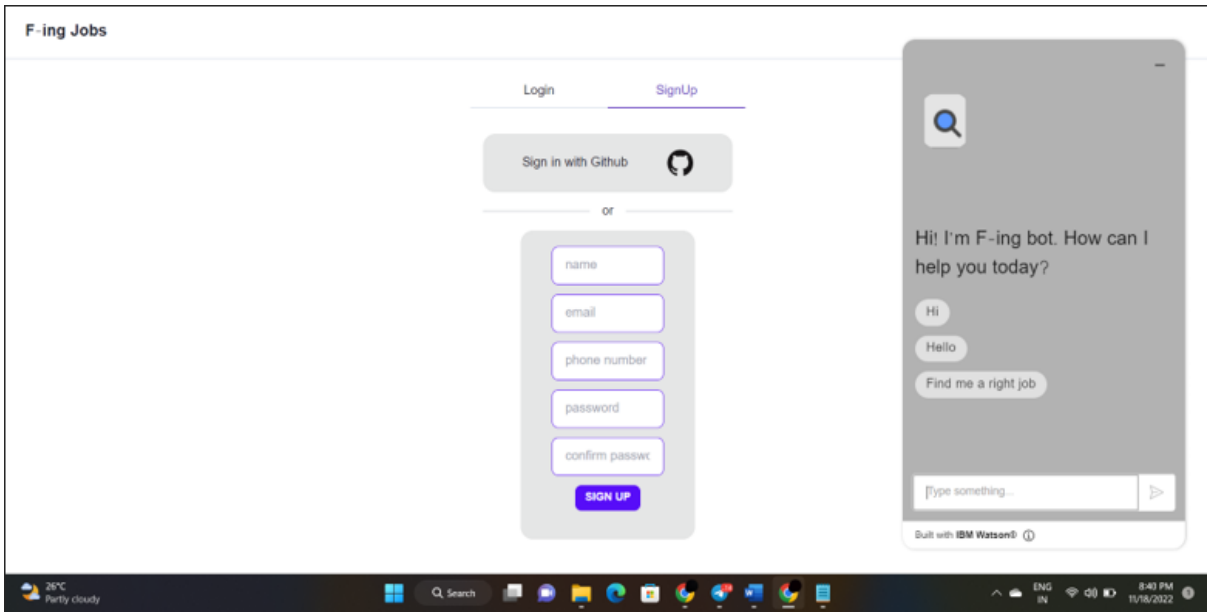


Fig 9.2 Using Chatbot for user

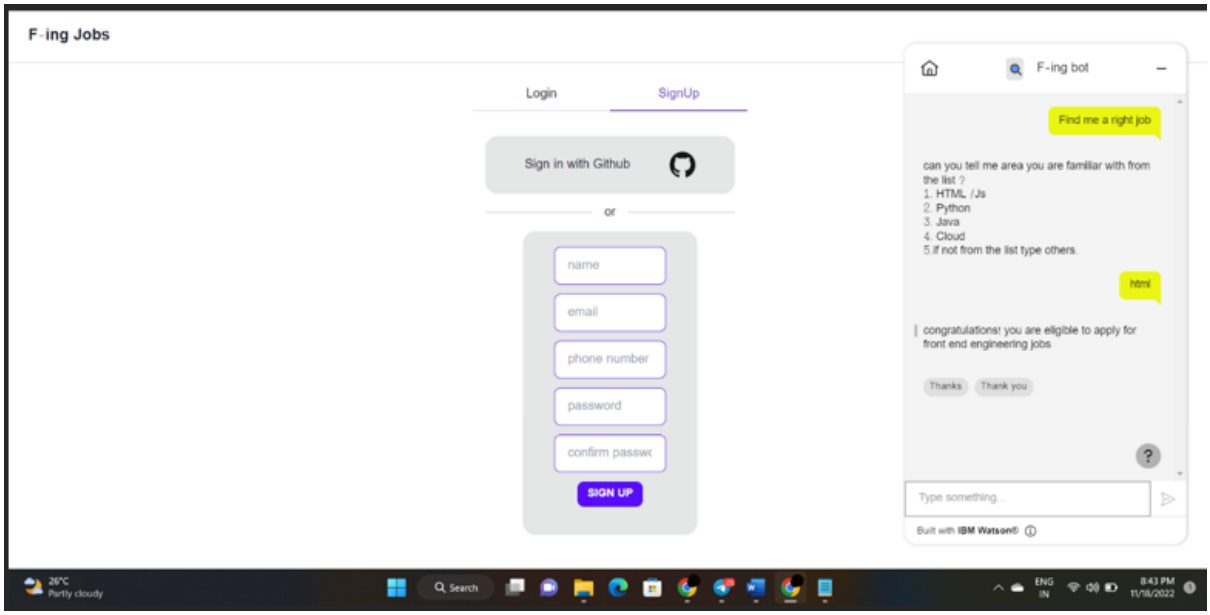


Fig 9.3 Interact with Chatbot

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

10.1 ADVANTAGES

- **Probabilistic hybrid approach :** Bidirectional recommendation and Relational aspects are included.
- **Proactive job recommender system :** Adaptive system & Use many attributes.
- **Semantic matchmaking for job recruitment :** Qualitative and quantity representation & Use two levels in skills matching (constrains and preferences).
- **System for screening candidates :** Various information retrieval techniques are used.

10.2 DISADVANTAGES

- **Probabilistic hybrid approach :** Less attributes used & No perfect measures.
- **Proactive job recommender system :** Key words search method & One way recommendation
- **Semantic matchmaking for job recruitment :** Tools and technologies skills excluded.
- **System for screening candidates :** Inefficient measures & One way recommendation.

CHAPTER 11

CONCLUSION

In this article, we used a literature analysis of many journals and proceedings related to the recruiting process and the job recommendation researches. We have seen from our literature review and from the challenges that faced the holistic E-recruiting platforms, an increased need for enhancing the quality of candidates/job matching. The recommender system technologies accomplished significant success in a broad range of applications and potentially a powerful searching and recommending techniques. Consequently, there is a great opportunity for applying these technologies in recruitment environment to improve the matching quality. This survey shows that several approaches for job recommendation have been proposed, and many techniques combined in order to produce the best fit between jobs and candidates. We presented state of the art of job recommendation as well as, a comparative study for its approaches that proposed by literatures. Additionally, we reviewed typical recommender system techniques and the recruiting process related issues. We conclude that the field of job recommendations is still unripe and require further improvements. As part of our ongoing research, we aim to build a new recommendation approach and test with real data for employee and staffing data from large companies. In addition to, we plan to enhance the similarity measures that suitable for this problem.

CHAPTER 12

FUTURE SCOPE

12.1 FUTURE SCOPE

In this field of job recommendation system there is a large scope for future work such as:

- By using different similarity measure we can see which gives the most accurate answer when compared with the other similarity measures.
- If we take into consideration the recommendation of the recommender system in contrast with the real life preferences, we can compare their mean absolute error.
- We can consider large number of parameters by giving them associated weights for more accurate Content Based recommendation.
- We can perform natural language processing to extract information from jobseekers resume and then recommending him the jobs

CHAPTER 13

APPENDIX

SOURCE CODE

```
from backend import create_app


app = create_app()


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


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_USERNAM
E']};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
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"All feilds 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"All feilds are required!"}), 409

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

    # Sql stmt 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
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"All feilds 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


@user.route('/profile', methods=["POST"])
@token_required
def update_user_info(current_user):
    user_id = current_user['USER_ID']
    update_fields = ['name', 'phone_number']
    for feild in update_fields:
        if not (feild in request.json):
            return jsonify({"error": f"All feilds are required!"}), 409
    name = request.json['name']
    phone_number = request.json['phone_number']
          sql = f"update users set name='{name}',phone_number='{phone_number}' where
user_id={user_id}"
    stmt = ibm_db.prepare(conn, sql)
    status = ibm_db.execute(stmt)
```

```
    if status:
        return jsonify({"name": name, "phone_number": phone_number}), 200
    else:
        jsonify({"error": "Something went wrong!!"}), 409
```

GITHUB & DEMO LINK

GITHUB : IBM-EPBL/IBM-Project-16373-1659612340: Skill / Job Recommender Application

DEMO LINK :

https://drive.google.com/file/d/1t8laNJOPZ81hWMbua5aGqLwB_N3S4QqA/view?usp=share_link