PROJECT REPORT

IBM-Project-53180-1661316851

TITLE	:	Skill / Job Recommender Application
TEAM LEADER	:	Soundharya N
TEAM MEMBER	:	Sowndharya R Sureka M Subappriya K Dhivya dharshini M
TEAM ID	:	PNT2022TMID07494

TABLE OF CONTENTS	PAGE NO
1. INTRODUCTION 1.1 Project Overview 1.2 Purpose	4-6

2. LITERATURE SURVEY 2.1 Existing problem 2.2 References 2.3 Problem Statement Definition	7-11
3. IDEATION & PROPOSED SOLUTION 3.1 Empathy Map Canvas 3.2 Ideation & Brainstorming 3.3 Proposed Solution 3.4 Problem Solution fit	11-15
4. REQUIREMENT ANALYSIS 4.1 Functional requirement 4.2 Non-Functional requirements	16
5. PROJECT DESIGN 5.1 Data Flow Diagrams 5.2 Solution & Technical Architecture 5.3 User Stories	17-19
6. PROJECT PLANNING & SCHEDULING 6.1 Sprint Delivery Schedule 6.2 Reports from JIRA	20

7. CODING & SOLUTION 7.1 Feature 1 7.2 Feature 2	21-22
8. TESTING 8.1 Test Cases 8.2 User Acceptance Testing	22-23
9. RESULTS 9.1 Performance Metrics	23-24
10.ADVANTAGES & DISADVANTAGES	

	24-25
11.CONCLUSION	25-26
12.FUTURE SCOPE	26-27
13.APPENDIX	27-32

1. INTRODUCTION

1.1 Project Overview

Recruitment is an important task for the modern recruitment industry. A good job recommender system will not only allow you to recommend higher-paying jobs that best match your current job skills, but also help you acquire some additional skills needed to land a new position.

This is where a recommender system can be useful for selecting a suitable track according to a student's understanding of various job domains. This paper proposes course recommendation system that suggests courses by finding students similar to target students and then searching for a pattern in their understanding of the domains and courses that they took. Finally, the system recommends courses whose association with the target student's job profile is high.

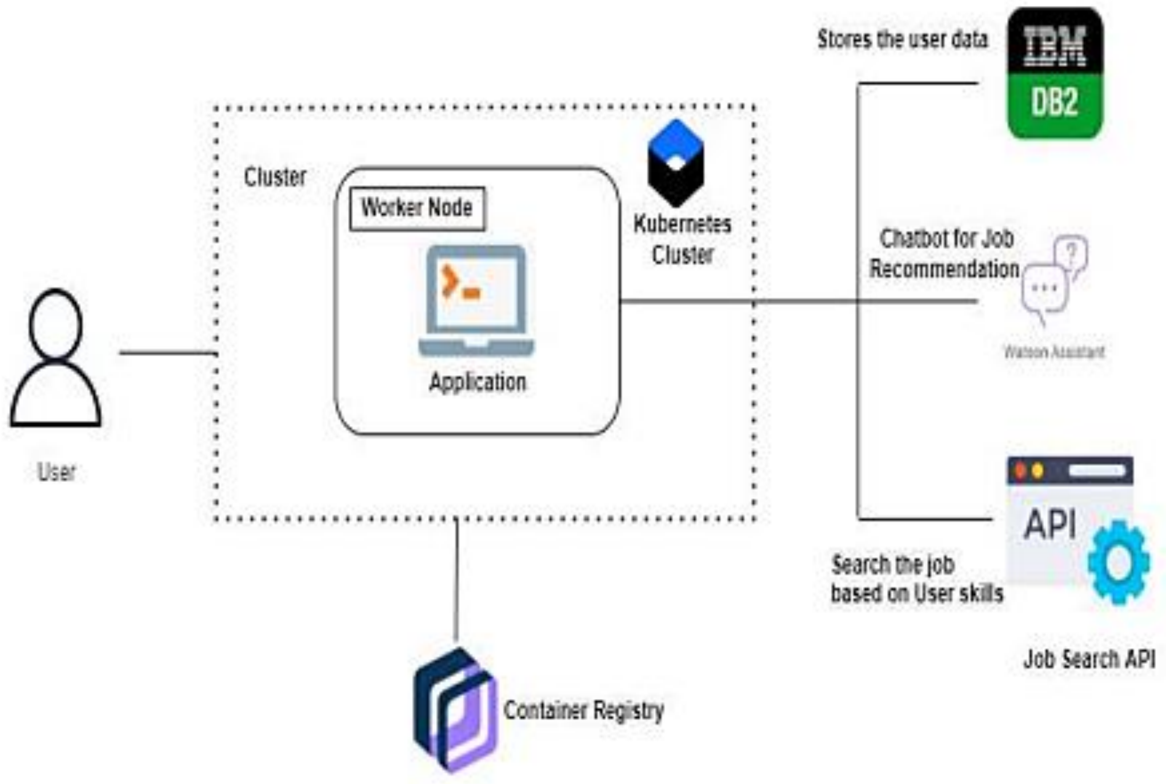
In this work, we constructed three types of information networks from historical job data:
(i) job transition networks,
(ii) job skill networks, and
(iii) skill co-occurrence networks.

Using information from all three networks, we provide a representation learning model that can collectively learn job and skill representations in a shared k-dimensional latent space. Our experiments show that by learning occupation and skill representations together, our model provides better recommendations for both occupations and skills. In addition, we also present some case studies that confirm our claims. Developing an end-to-end web applicationthat can display current job listings based on user's skills.

Users and their information are stored in a database. Based on the user's skill, a notification will be sent when there is a vacancy. Users can interact with the chatbot and get recommendations based on their skills. We may use the job search API to retrieve the latest job listings in the market. This will pull the data directly from the website.

1.2 Purpose

A good job recommender system will not only allow you to recommend high-paying jobs that best match your current job's skills, but also acquire only a few additional skills required to get a new position. I suggest creating software that filters jobs based on the skills of candidates seeking work. This filtered job is then recommended to these candidates based on their skills.



2.LITERATURE SURVEY

In the Paper, Amer Al-Badarenah et al. have proposed a system which recommends student courses based on the courses taken by their peers, having similarinterests. The similar interest students are then grouped based on their grades. Clusters are created by using the nearest neighbor algorithm. Their system also predicts grades that the student will achieve if they take up that course. The author has used Manhattan distance to find similarity between two students by calculating the distance between their grades in common subjects and to form clusters. After the clusters are formed, thesimilarity of the student in interests and the cluster is found from the group by using the
n-nearest neighbor algorithm. Course association rules are used in finding recommendations. It is found that in some scenarios the algorithm's performance is low.The prediction depends only on the grades which may not help in knowing the students entire potential.

Sh. Asadi et al talk about using both clustering and fuzzy logic to predict courses The author uses a clustering algorithm to group similar kinds of students and fuzzy logic to find association rules. PCA is used to decrease the number of components. The clustering is done with K means algorithm and the maximum number of clusters considered in this paper are 5. Student's scores are labeled as low, medium, high. If two
regions have equal membership that is if students mark in a subject belongs to two different regions with equal membership then both will be added in a set of rules. This is
a drawback.

Feng-Hsu Wanh et al talks about improving the recommendation system on websites and making the user's experience personalized. They have integrated clustering with the associative-mining method. The customers are clustered based on the time-framed navigation session assuming that their preference may change with time. For the selection of a good time frame they have used the Hierarchical Bisecting Medoids Algorithm. The nearest cluster is found and association rule is applied to that cluster. The maximum matching method preserves the sessions and finds maximum pages that match the rules and pages will be displayed whose confidence is beyond a threshold. This paper suggests that this method will be ideal in education system courses recommendation. However, the drawback of this system is that a customer won't be given recommendations if there are no pages match the association rule. Huiyi Tan et al. [4] propose a system that can generate recommendations for E-Learning. They collect the data of the user's previous interest and frequently visited pages are considered. Different recommendation models are stored in a separate dataset and the model is randomly selected. Once the model is selected the useful information from the user's history and recommendations are generated based on the algorithm that the model uses and sent back to the E-Learning websites.

2.1 Existing system:

The developed system consists of three modules: college campus recruitmentsystem, keyword-based search from online recruitment sites and Android application. In college campus recruitment system student's profiles and company's profiles are collected. Students profile generated by taking information from students through registration and login portal. Company's profile will be generated by the admin from the information and requirement provided by the company to admin. After that profile matching is perform on the students and company’s profiles. In second module i.e., keyword-based search module students have the provision to search for the companies from various online recruitment sites. Web crawling technique is used for searching through these sites. Students have to put the keyword e.g. C# and web crawler searches for those companies who have vacancies for C# developers through various online recruitment sites like Naukri.com.

2.2 References:

[1] The International Workshop on Artificial Intelligence and Smart City Applications (IWAISCA)August 9-12, 2020,Leuven, Belgium Job Recommendation based on Job Profile Clustering and Job Seeker Behavior.

[2] Wang, F.H. and Shao, H.M., 2004. Effective personalized recommendation based on time-framed navigation clustering and association mining. Expert systems with applications, 27(3), pp.365-377.

[3] Al-Badarenah, A. and Alsakran, J., 2016. An automated recommender system for course selection. International Journal of Advanced Computer Science and Applications, 7(3),pp.166-175.

[4] Asadi, S., Jafari, S. and Shokrollahi, Z., 2019. Developing a Course Recommender by Combining Clustering and Fuzzy Association Rules. Journal of AI and Data mining, 7(2), pp.249-262.

[5] Wang, F.H. and Shao, H.M., 2004. Effective personalized recommendation based on time-framed navigation clustering and association mining. Expert systems with applications, 27(3), pp.365-377.

[6] H. Tan, J. Guo and Y. Li, ”E-learning Recommendation System,” 2008 International Conference on Computer Science and Software Engineering, Hubei, 2008, pp. 430-433.

2.3 Problem Statement Definition

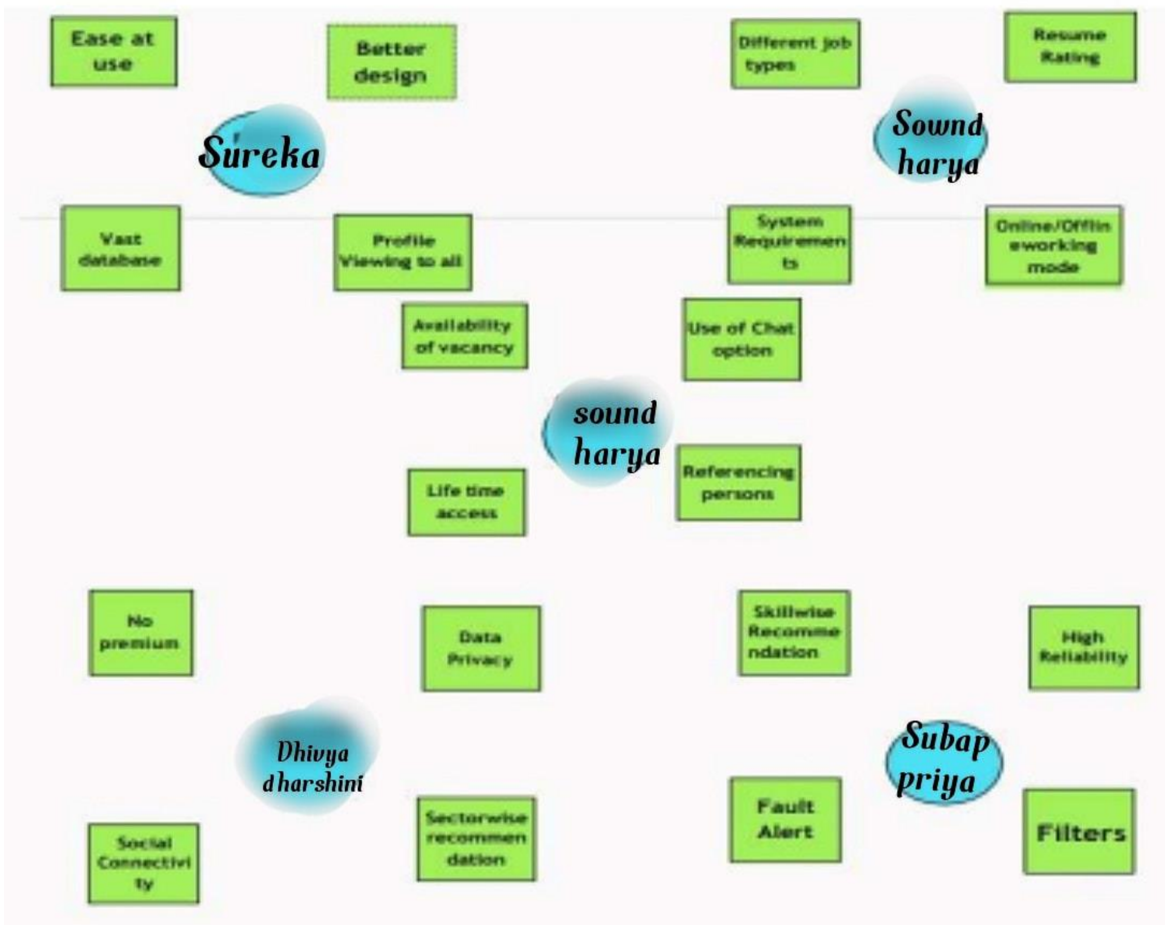
Having lots of skills but wondering which job will best suit you? Don’t need to worry! We have come up with a skill recommender solution through which the fresher or the skilled person can log in and find the jobs by using the search option or they can directly interact with the chatbot and get their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

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming



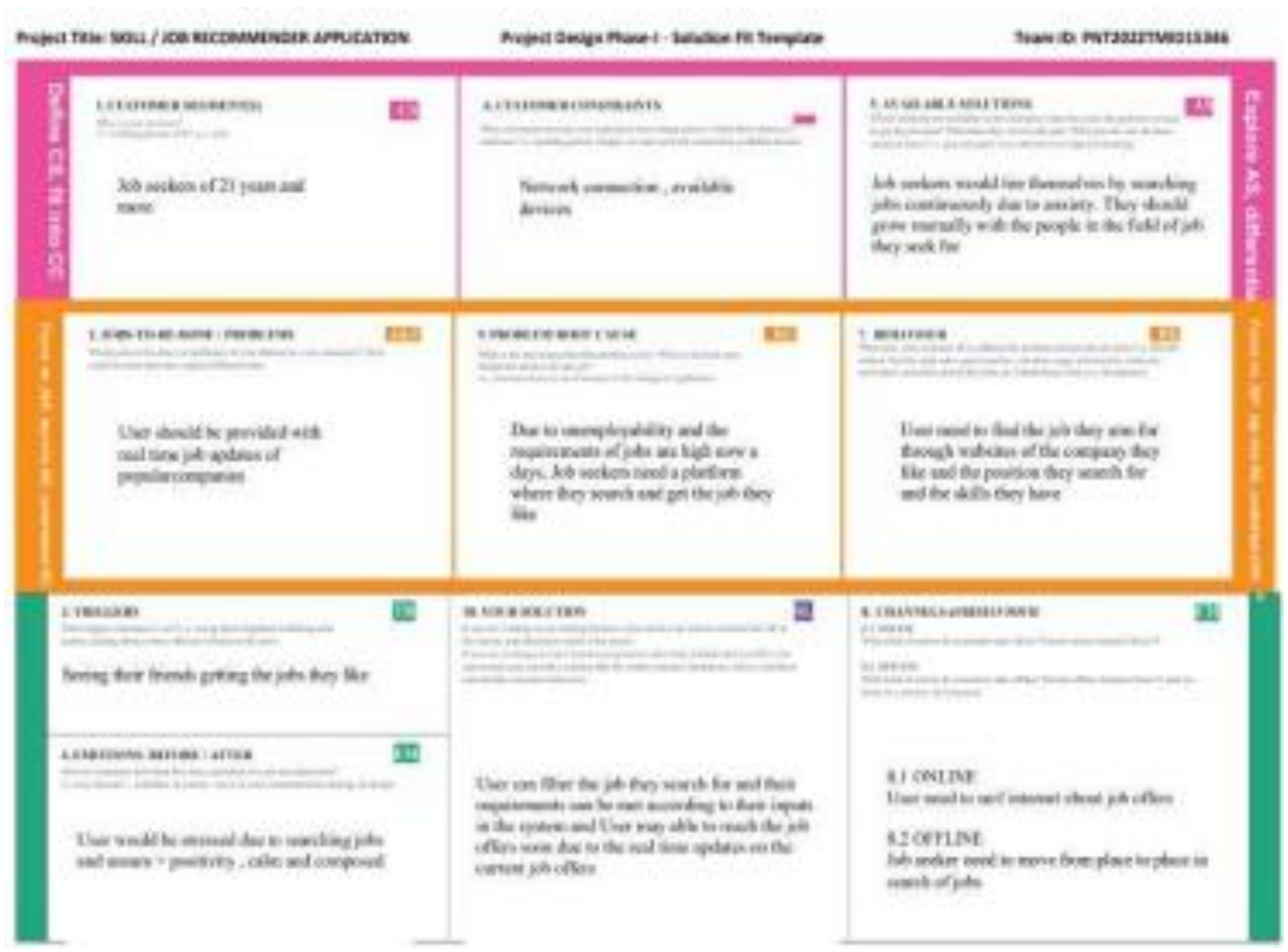
3.3 Proposed Solution

S.NO.	PARAMETER	DESCRIPTION
1.	Problem Statement (Problem to be solved)	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.
2.	Idea / Solution description	There are three contributions to this work. i) published a new data set consisting of a set of job seeker profiles and a set of job listings collected from various job search engine websites; ii) proposed a framework for submitted employment recommendations

		based on the professional skills of the job seeker iii)Conducting assessments to quantify. Empirically, the recommendation of two state-of-the-art methods considering different configurations within the proposed framework. We therefore present a general panorama of job recommendation tasks with the aim of facilitating real-world research and application design related to this important topic.
3.	Novelty / Uniqueness	We will propose the best position according to each person's skillset. This is based on known profile positions.

3.4 Problem Solution fit



Project Title: SKILL / JOB RECOMMENDER APPLICATION

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID15346

1. CUSTOMER SEGMENT(S)

Job seekers of 21 years and more

6. CUSTOMER CONSTRAINTS

Network connection , available devices

5. AVAILABLE SOLUTIONS

Job seekers would tire themselves by searching jobs continuously due to anxiety. They should grow mutually with the people in the field of job they seek for

2. JOBS-TO-BE-DONE / PROBLEMS

User should be provided with real time job updates of popularcompanies

9. PROBLEM ROOT CAUSE

Due to unemployability and the requirements of jobs are high now a days. Job seekers need a platform where they search and get the job they like

7. BEHAVIOUR

User need to find the job they aim for through websites of the company they like and the position they search for and the skills they have

3. TRIGGERS

Seeing their friends getting the jobs they like

4. EMOTIONS: BEFORE / AFTER

User would be stressed due to searching jobs and ensure > positivity , calm and composed

10. YOUR SOLUTION

User can filter the job they search for and their requirements can be met according to their inputs in the system and User may able to reach the job offers soon due to the real time updates on the current job offers

8. CHANNELS of BEHAVIOUR

8.1 ONLINE

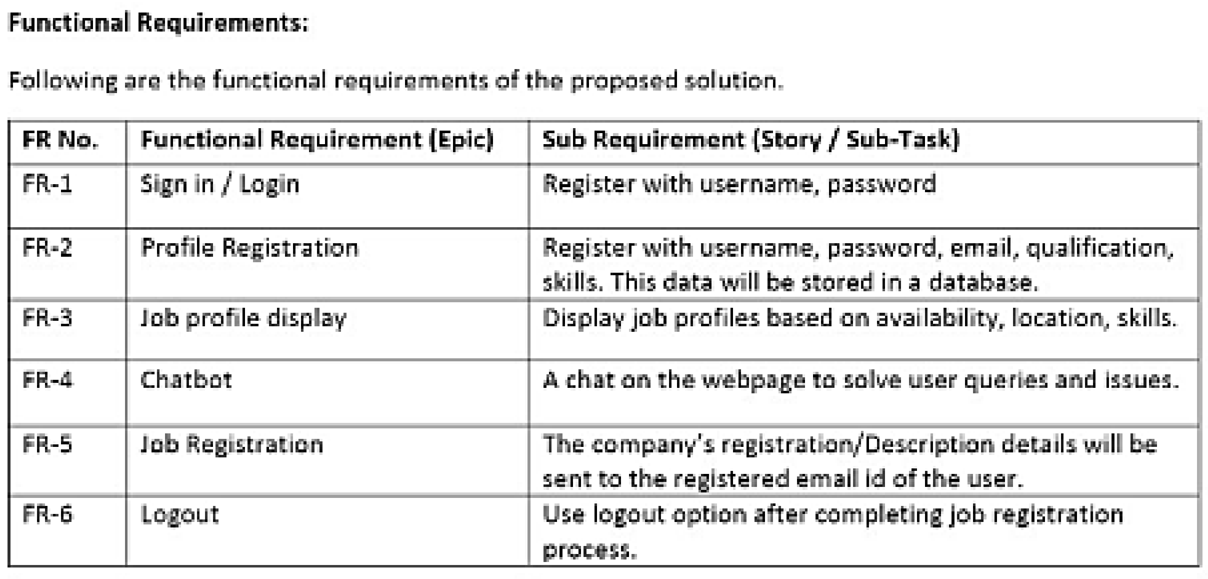
User need to surf internet about job offers

8.2 OFFLINE

Job seeker need to move from place to place in search of jobs

4.REQUIREMENT ANALYSIS

4.1 Funtional requirement

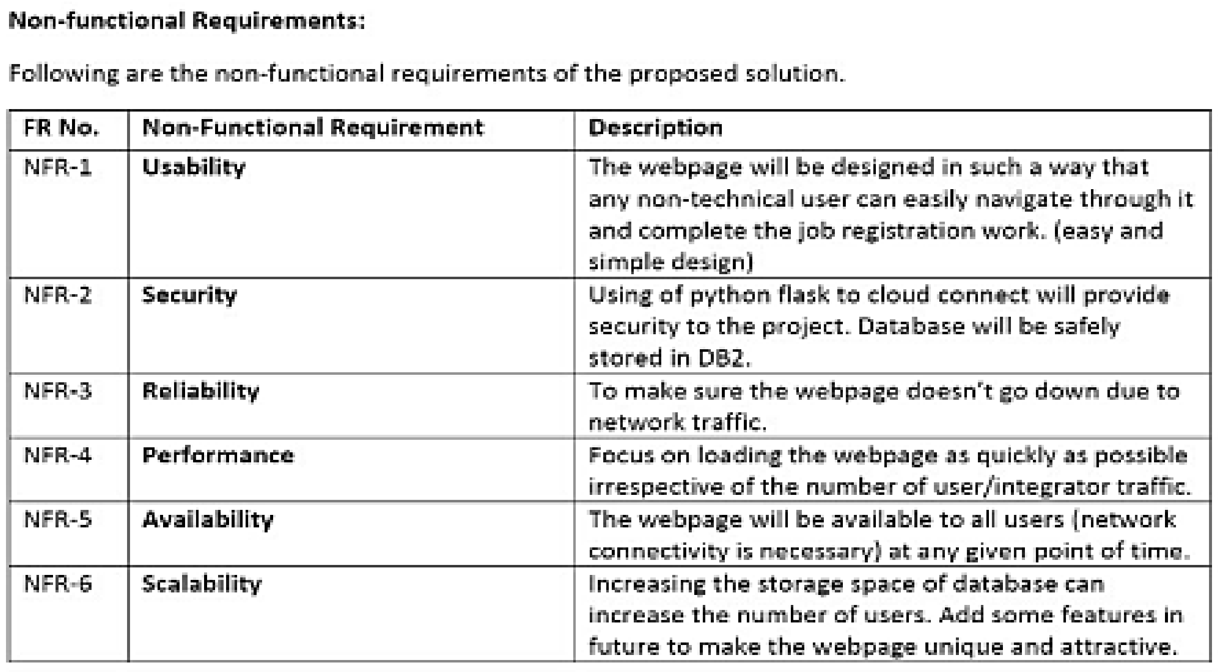


Functional Requirements:

Following are the functional requirements of the proposed solution.

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

4.2 Non-Functional requirements



Non-functional Requirements:

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

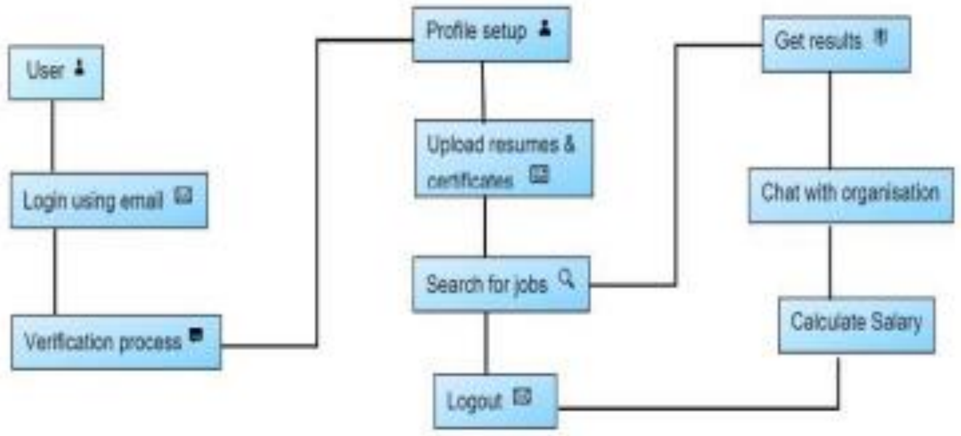
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The webpage will be designed in such a way that any non-technical user can easily navigate through it and complete the job registration work. (easy and simple design)
NFR-2	**Security**	Using of python flask to cloud connect will provide security to the project. Database will be safely stored in DB2.
NFR-3	**Reliability**	To make sure the webpage doesn't go down due to network traffic.
NFR-4	**Performance**	Focus on loading the webpage as quickly as possible irrespective of the number of user/integrator traffic.
NFR-5	**Availability**	The webpage will be available to all users (network connectivity is necessary) at any given point of time.
NFR-6	**Scalability**	Increasing the storage space of database can increase the number of users. Add some features in future to make the webpage unique and attractive.

5 PROJECT DESIGN

5.1 Data Flow Diagrams

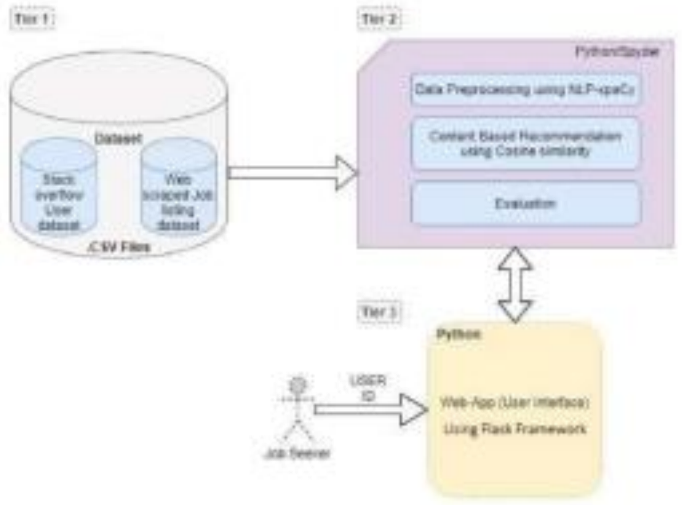
Data Flow Diagrams:

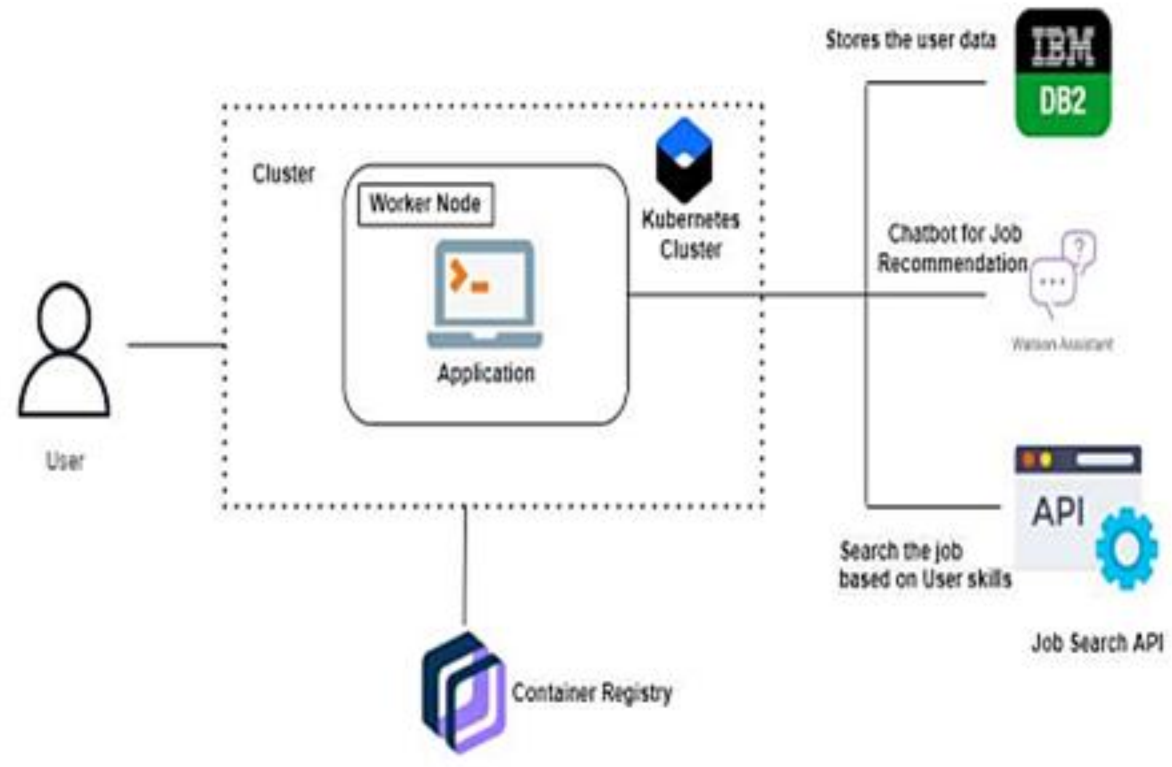
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



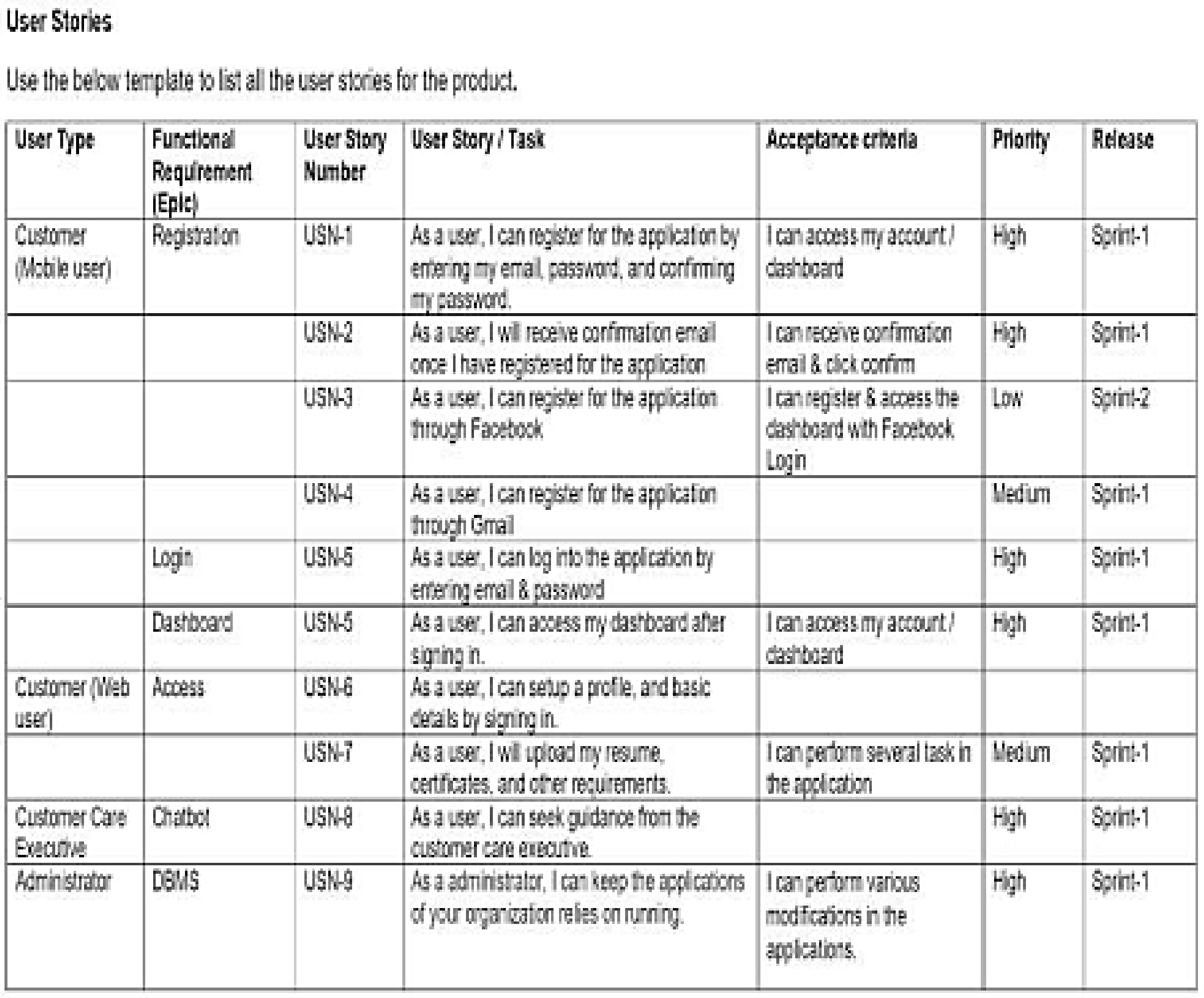
5.*2 Solution & Technical Architecture*

Example - Solution Architecture Diagram:


Tier 1
Tier 2
Tier 3
Dataset
Stack overflow User dataset
Web scraped Job listing dataset
.CSV Files
Python/Spyder
Data Preprocessing using NLP-spaCy
Content Based Recommendation using Cosine similarity
Evaluation
Python
USER ID
Web-App (User interface)
Using Flask Framework
Job Seeker



5.3 User Storie



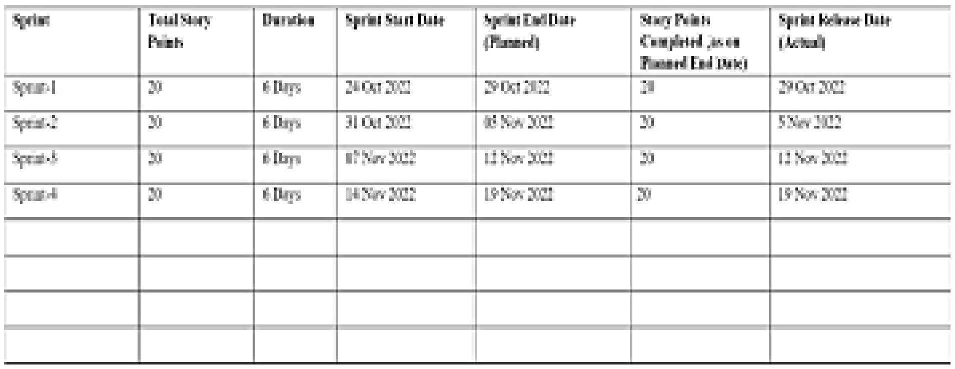
User Stories

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.			
		USN-7	As a user, I will upload my resume, certificates, and other requirements.	I can perform several task in the application	Medium	Sprint-1
Customer Care Executive	Chatbot	USN-8	As a user, I can seek guidance from the customer care executive.		High	Sprint-1
Administrator	DBMS	USN-9	As a administrator, I can keep the applications of your organization relies on running.	I can perform various modifications in the applications.	High	Sprint-1

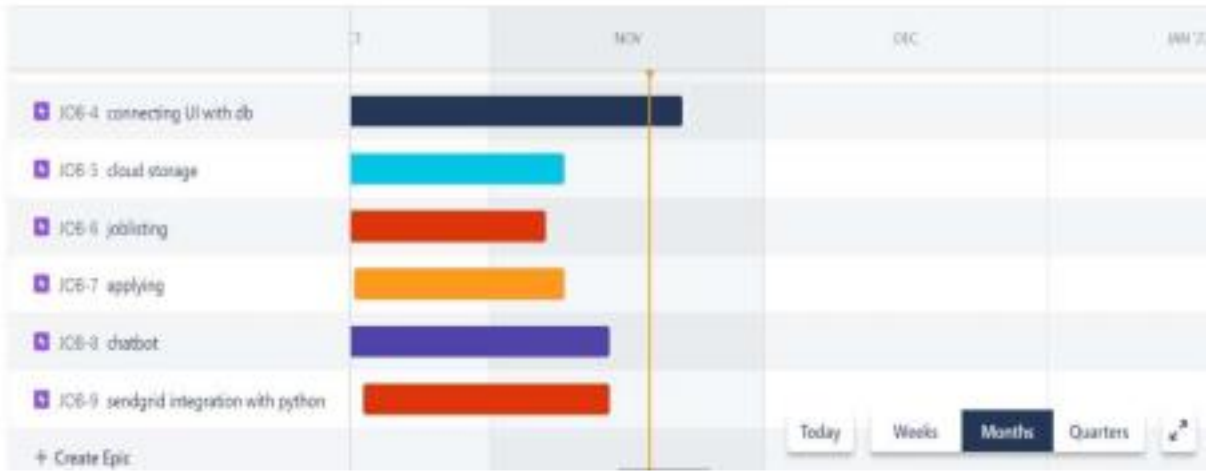
6.PROJECT SCHEDULING

6.1 SPRINT DELIVERY SCHEDULE



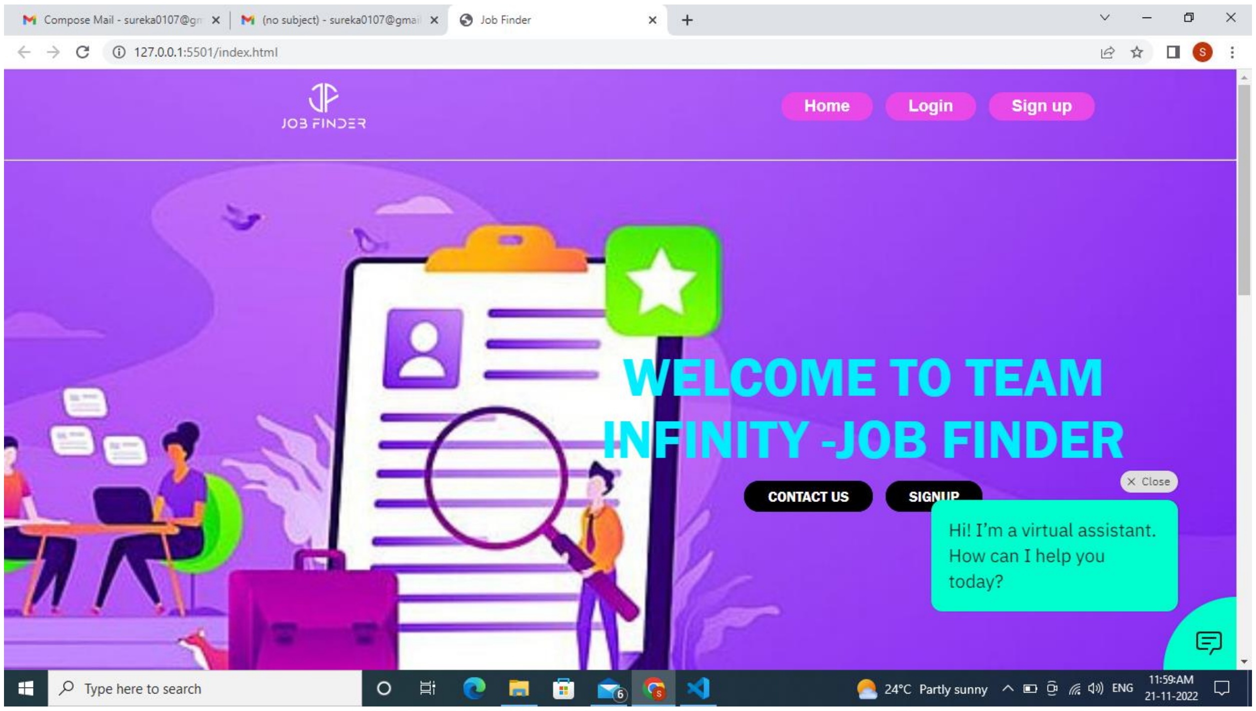
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	5 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.2 Reports from JIRA



7 .CODING & SOLUTION

7.1 Feature 1

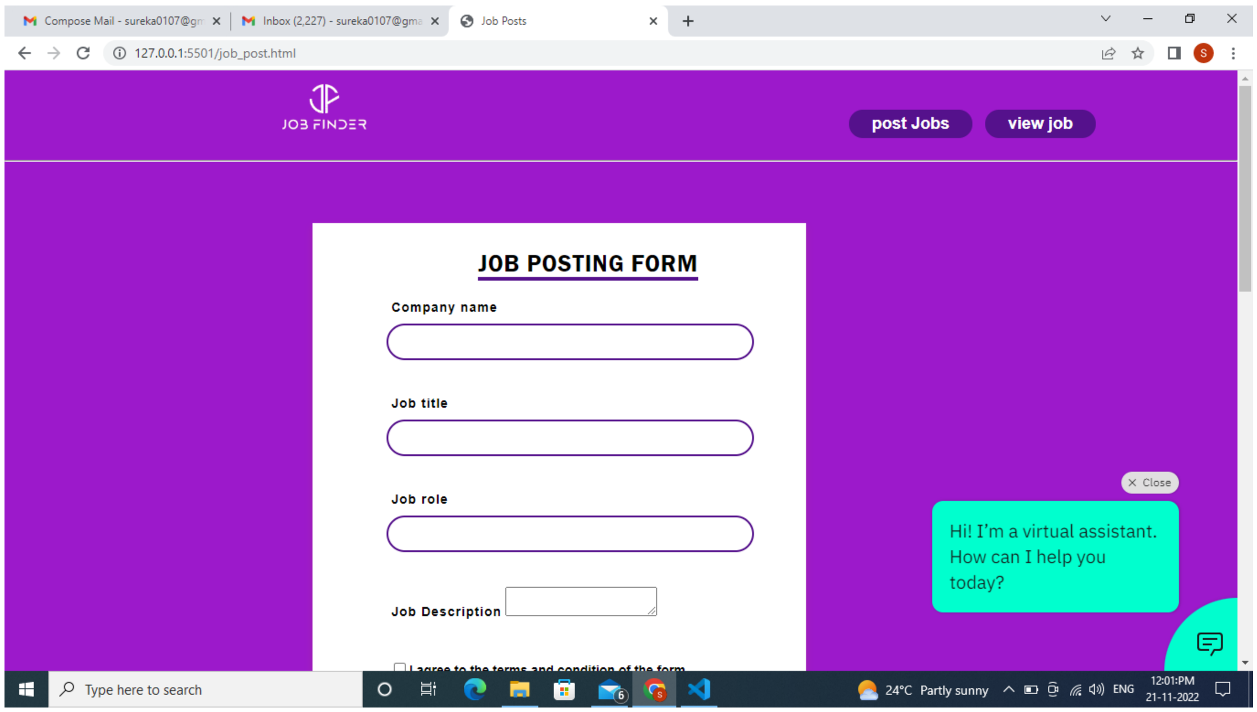

Compose Mail - sureka0107@gm
(no subject) - sureka0107@gmai
Job Finder
127.0.0.1:5501/index.html
JOB FINDER
Home
Login
Sign up
WELCOME TO TEAM INFINITY -JOB FINDER
CONTACT US
SIGNUP
Close
Hi! I'm a virtual assistant. How can I help you today?
Type here to search
24°C Partly sunny
ENG
11:59 AM
21-11-2022

The software has built-in features that can assist with ongoing inquiries, provide quick and effective resolution to user issues that may arise, and bring management attention if necessary. There is a "chat bot" for in the event of any complications, customer service is available 24/7 to assist with pending issues.

7.2 Feature 2

In this project, we created a dashboard page that displays available jobs and provides easy access to the website.

- *Communicate information quickly.*
- *Display information clearly and efficiently.*
- *Shows trends and changes in data over time.*
- *Easy to customize.*
- *The most important widgets and data components are displayed effectively in limited spaces.*



JOB POSTING FORM

Company name

Job title

Job role

Job Description

I agree to the terms and condition of the form

Hi! I'm a virtual assistant. How can I help you today?

TESTING

8.1 Test Cases

Software testing is the process of evaluating and verifying that a software product or application performs its intended functions. Benefits of testing include avoiding bugs, reducing development costs, and improving performance. This software has been successfully tested and evaluated.

8.2 User Acceptance Testing

Purpose of Document

The purpose of this document is to briefly describe the test coverage and open issues for the Inventory Management System project at the time of User Acceptance Testing (UAT) release.

User acceptance testing is performed in a separate test environment. Changes, updates, or new features are requested and developed. Unit and integration tests are run. Everything seems fine. However, serious problems arose after publication. Reworking and retesting is not the most expensive outcome in this case.

9 .RESULTS

9.1 Performance Metrics

We analyze the performance of all the above techniques based on the above two types of user recommendations. The resulting job recommended for each new user is then compared to the job the user originally belonged to according to the test record. If the original user job is recommended in the model result, the model adds 1 if yes, 0 otherwise.

This resulting array of 0s and 1s is checked for accuracy by calculating the number of 1s from the total user predictions. Among all models built with different similarity measures, the job recommendation system model based on cosine similarity outperformed all remaining models. The metrics used to analyze model performance are accuracy, precision, recall, and F1 score. This is because cosine takes into account the presence of duplicate terms when calculating similarity. Also, because only *nonzero dimensions are considered, cosine has low computational complexity and is easy to work with surrogate data vectors.*

Analysis of the results table shows that job recommendations based on top 5 and highest scores and high. The error rate will be 6-10%.

10. ADVANTAGES

* *The model doesn't need any data about other users, since the recommendations are specific to this user.*

* *This makes it easier to scale to a large number of users.*

* *The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.*

DISADVANTAGES

* *Since the feature representation of the items are hand-engineered to some extent,this technique requires a lot of domain knowledge. Therefore, the model can only be good as the hand-engineered features.*

 The model can only make recommendations based on existing interests of the user.

 In other words, the model has limited ability to expand on the users' existing interests .

11. CONCLUSION

In this project, we compared Content-Based Filtering and Recommendations' Collaborative Filtering. Additionally, the Aggregation Plus Recommender system was developed. Content-based filtering recommends results based on a user's personal preferences and matching to a particular document, while collaborative filtering recommends results based on other users' preferences. In evaluating these two methods, it was concluded that both hybrid systems overcome both limitations and improve the efficiency of ranking. Fixed issues with cold start, sparse database, scalability, and missing trend recommendations. The suggestion is to design a Job Recommender system that prioritizes quality over quantity. While job websites and portals already exist that recommend jobs based on job seeker profiles, this study of aggregated quality recommendations was achieved by overcoming limitations *through selective crawling. it was done. A full-featured user interface is designed to tie everything together and give users a seamless experience.*

<u>12. FUTURE SCOPE</u>

A future task in the case of a personalized job recommendation system is to use the user's preferred location to obtain job recommendations based on the jobs of organizations located in nearby regions. This can be done by extracting the latitude and longitude of the user's preferred location and calculating the Euclidean distance between the latitude and longitude.

of the organization's location. This excludes other jobs far from the user's preferred location, resulting in more accurate job recommendations.

As part of future work, we plan to use similar candidate and job functions for sequencing information. So far, recommendations for similar candidates and jobs are part of non machine learning-based recommendations, and initial results look promising. Finally, it would be interesting to extend our methodology to other recommendation systems.

<u>13. APPENDIX</u>

Source Code

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

app = Flask(__name__)
app.secret_key='vy@ur434'
#def connection():
try:
 conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=3883e7e4-
18f5-4afe be8c
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=
31498;SECUR
ITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=kwb02404
;PWD=1fso aAozUL4nkgW3",'','')
 print(conn)
 print("connection successful...")
 #return conn
except:

 print("Not Connected to Database")

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

@app.route('/contacts')
def contacts():
 return render_template('contacts.html')
```

```
@app.route('/forgot')
def forgot():
 return render_template('forgotten-password.html')
# return render_template('login.html' ,msg="success")

@app.route('/posts')
def posts():

 return render_template('job_post.html')

@app.route('/addrec',methods=['POST','GET'])
def addrec():

 arr = []
 sql = "SELECT * FROM LIST"
 stmt = ibm_db.exec_immediate(conn,sql)
 dictionary = ibm_db.fetch_both(stmt)
 while dictionary != False: @app.route('/signup', methods=['POST','GET'])
def signup():
 if request.method == 'POST':
 # conn = connection()
 try:
 sql = "INSERT INTO PRAC
VALUES('{}','{}','{}','{}')".format(request.form["name"],request.form["email"],r
equest.f orm["phone"],request.form["password"])
 ibm_db.exec_immediate(conn,sql)
 #flash("successfully Registered !")
 return render_template('login.html')
 except:
 #flash("Account already exists! ")
 return render_template('signup.html')
 else:
```

```
return render_template('signup.html')
# name = request.form['name']

#email = request.form['email']
#phone = request.form['phone']
#password = request.form['password']

#sql ="INSERT INTO users VALUES (?,?,?,?)"
#stmt = ibm_db.prepare(conn,sql)
#ibm_db.bind_param(stmt, 1, name)
#ibm_db.bind_param(stmt, 2, email)
#ibm_db.bind_param(stmt, 3, phone)
#ibm_db.bind_param(stmt, 4, password)
#ibm_db.execute(stmt)
# return render_template('signup.html')

@app.route('/login', methods=['POST','GET'])
def login():
if request.method == 'POST':
# conn =connection()
email = request.form["email"]
password = request.form["password"]
sql = "SELECT COUNT(*) FROM PRAC WHERE EMAIL=? AND
PASSWORD=?"
stmt = ibm_db.prepare(conn,sql)
ibm_db.bind_param(stmt, 1, email)
ibm_db.bind_param(stmt, 2, password)
ibm_db.execute(stmt)
res = ibm_db.fetch_assoc(stmt)
if res['1'] == 1:
session['loggedin'] = True
session['email'] = email
```

```
 return render_template('job_post.html')
 else:

#flash("email/ Password isincorrect! ")
 return render_template('login.html')
 else:
 return render_template('login.html')



 #email = request.form['email']
 #password = request.form['password']

 #sql = "SELECT * FROM users WHERE email=%s AND
password=%s"   #stmt = ibm_db.prepare(conn, sql)
 #ibm_db.bind_param(stmt,1,email)
 #ibm_db.bind_param(stmt,2,password)
 # user = ibm_db.execute(stmt).fetchone()


 inst={}
 inst['DNAME']=dictionary['DNAME']
 inst['DTITLE']=dictionary['DTITLE']
 inst['DROLE']=dictionary['DROLE']

inst['DESCRIPTION']=dictionary['DESCRIPTI
ON']   arr.append(inst)
 dictionary = ibm_db.fetch_both(stmt)
 return render_template('list.html')

@app.route('/list')
def list():
 return render_template('list.html')
```

```
    if __name__=='__main__':
app.run(debug=True)
```

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-53180-1661316851

Project Demo Link: *https://youtu.be/UFe6-Pkpvkw*