PROJECT REPORT

IBM-Project-39021- 1660389280

TITLE:

Skill / Job Recommender Application

TEAM LEADER:

JASHIMA FATHIMA R

TEAM MEMBER:

DHANSREE R

JAISHREE K

GETHARA GOWRI R

NEHA S

TEAM ID: PNT2022TMID31521

TABLE OF CONTENTS	PAGE NO
1. **INTRODUCTION** 1.1 Project Overview 1.2 Purpose	**1-2**
2. **LITERATURE SURVEY** 2.1 Existing problem 2.2 References 2.3 Problem Statement Definition	**3-5**
3. **IDEATION & PROPOSED SOLUTION** 3.1 Empathy Map Canvas 3.2 Ideation & Brainstorming 3.3 Proposed Solution 3.4 Problem Solution fit	**6-9**
4. **REQUIREMENT ANALYSIS** 4.1 Functional requirement 4.2 Non-Functional requirements	**9-10**
5. PROJECT DESIGN 5.1 Data Flow Diagrams 5.2 Solution & Technical Architecture 5.3 User Stories	**10-13**
6. PROJECT PLANNING & SCHEDULING 6.1 Sprint Planning & Estimation 6.2 Sprint Delivery Schedule 6.3 Reports from JIRA	**14-15**
7. CODING & SOLUTIONING 7.1 Feature 1 7.2 Feature 2	**15-16**
8. TESTING 8.1 Test Cases 8.2 User Acceptance Testing	**17**

CHAPTER-1

1.INTRODUCTION

1.1 Project Overview

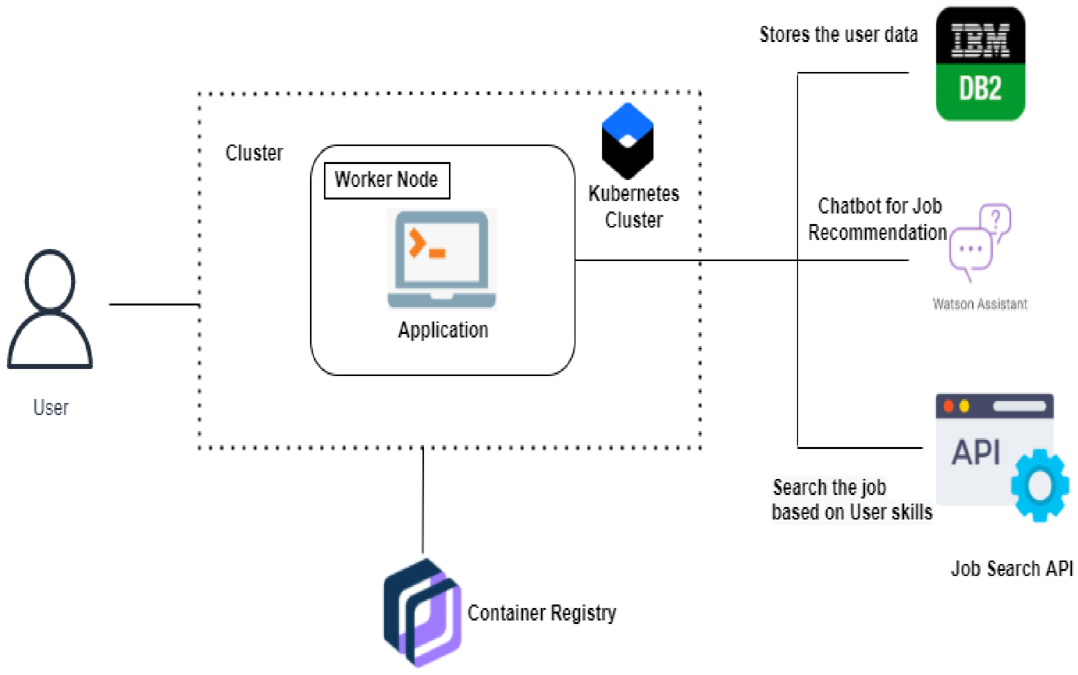
Nowadays, searching for a job on the Internet is a regular practise using job search engines like LinkedIn1, Indeed2, and others. A job seeker can typically use these sites in one of two ways: 2) building and/or updating a professional profile containing information related to his/her education, professional experience, professional skills, and other, and receiving personalised job recommendations based on this data. 1) conducting a search based on keywords related to the job vacancy that he/she is looking for. Sites that support the first scenario are more widely used and have a simpler layout, but their recommendations are less precise than those of the sites that use profile data.

Thoughts on the greatest employment for someone with a variety of skills you shouldn't be concerned. With our skill recommender solution, either a skilled or a fresher user may sign up, search for jobs using the search bar, or speak with the chatbot directly to land their ideal position.
To create a complete online application that can show available jobs based on user skill sets. The database contains both the user and their data. Based on the user's skill set, an alert is delivered when a position becomes available. The chatbot will be used by users, who can engage with it and receive recommendations depending on their skill set. To obtain the most recent job vacancies on the market, we can utilize a job search API, which will retrieve the information straight from the website.

1.2 Purpose:

These days, there are many opportunities available worldwide, and there are more people than ever. To survive in today's striving and competitive society, people have a great need for jobs. The current situation has a high rate of literacy, but the main issue is that educated people are underemployed. The problem of unemployment is also brought on by a failure to match the appropriate talents to the appropriate jobs.

Therefore, the primary goal of the skills/job recommender application is to assist individuals in selecting a position that is appropriate for their career. It offers job searchers a variety of skills and employment options.



CHAPTER-2

LITERATURE SURVEY

2.1 Existing problem

As a recommendation algorithm, the existing recommendation system employs Collaborative filtering (CF) and its modifications. There are two ways to perform collaborative filtering: user-based collaborative filtering and collaborative filtering using items, respectively. This recommendation engine is mostly built in two steps. To begin, determine how many users/items in the database are comparable to the given user/item. Second, assess other users/items to predict what grade you would give the user of his item based on the total weight of the users/items that are more similar to this one.

Correlations between vectors of users/items are used to calculate similarity.

2.2 References

[1] Amber Nigam, Aakash Roy, Arpan Saxena, Hartaran Singh Job Recommendation through Progression of Job Selection 2019, IEEE 6th International Conference on Cloud Computing and Intelligence System (CCIS)December 2019.

[2] Jorge Valverde Rebaz, Ricardo Puma, PaulBustios, Nathalia C.Silva.Job Recommendation based on Job Seeker Skills: An Empirical Study First Workshop on Narrative Extraction From Text(Text2Story 2018) colocated with40th European Conference on Information Retrieval(ECIR 2018)At:Grenoble,France March 2018.

[3] Yi-Chi Chou and Han-Yen Yu. Based on the application of AI technology in resume analysis and job recommendation. In 2020 IEEE International Conference on CComputational Electromagnetics(ICCEM). IEEE, 2020.

[4} Motebang Daniel Mpela and Tranos Zuva. A mobile proximity job employment recommender system.In 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Commu-nication Systems (icABCD), pages 1–6. IEEE, 2020.

[5] Jack Bandy. Problematic machine behavior: A systematic literature review of algorithm audits. Pro-ceedings of the ACM on Human-Computer Interaction, 5(CSCW1):1–34, 2021

[6 Jorge Valverde Rebaz,Ricardo Puma, PaulBustios,Nathalia C.Silva.Job Recommendation based on Job Seeker Skills: An Empirical Study FirstWorkshop on Narrative Extraction From Text(Text2Story 2018) colocated with40th European Conference on Information Retrieval(ECIR 2018)

[7] Himan Abdollahpouri, Gediminas Adomavicius, Robin Burke, Ido Guy, Dietmar Jannach, Toshihiro Kamishima, Jan Krasnodebski, and Luiz Pizzato 2020.

[8] Motebang Daniel Mpela and Tranos Zuva. A mobile proximity job employment recommender system.In 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Commu-nication Systems (icABCD), pages 1–6. IEEE, 2020.

[9] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. 2020. Addressing the multistakeholder impact of popularity bias in recommendation through calibration. (2020).

[10] Shuqing Bian, Xu Chen, Wayne Xin Zhao, Kun Zhou, Yupeng Hou, Yang Song, Tao Zhang, andJi-Rong Wen. Learning to match jobs with resumes from sparse interaction data using multi-viewco-teaching network. In Proceedings of the 29th ACM International Conference on Information &Knowledge Management, pages 65–74, 2020.

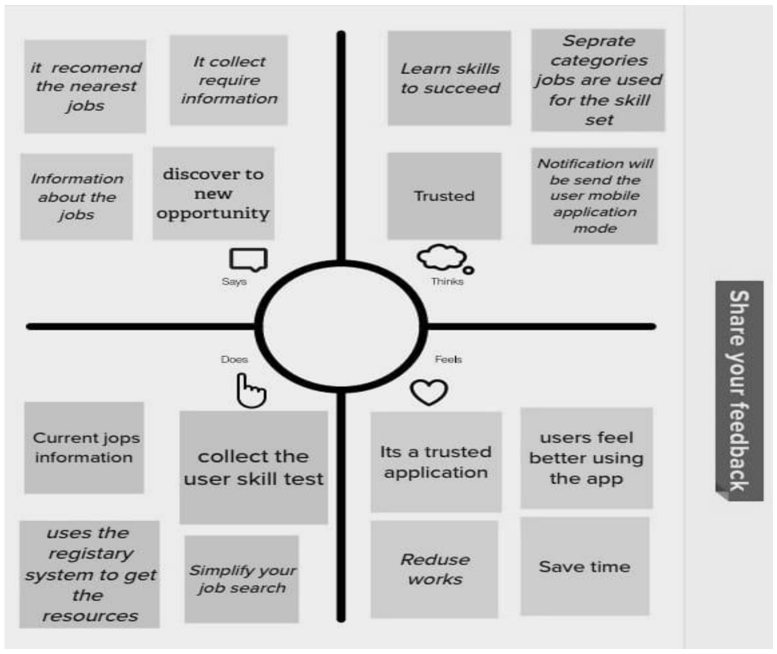
2.3 Problem Statement Definition

It is crucial to filter, prioritize, and effectively transmit essential information on the Internet, where the sheer volume of alternatives is overwhelming, in order to reduce information overload, which has potentially caused a problem for many Internet users.The most challenging and time-consuming task is gathering information and choosing the optimum user-job relationship mapping based on a user's skills and interests. Relevant employment advice would consequently facilitate the work of a wide range of job seekers. If this were the case, each user would spend less time online looking for and applying for potential jobs.

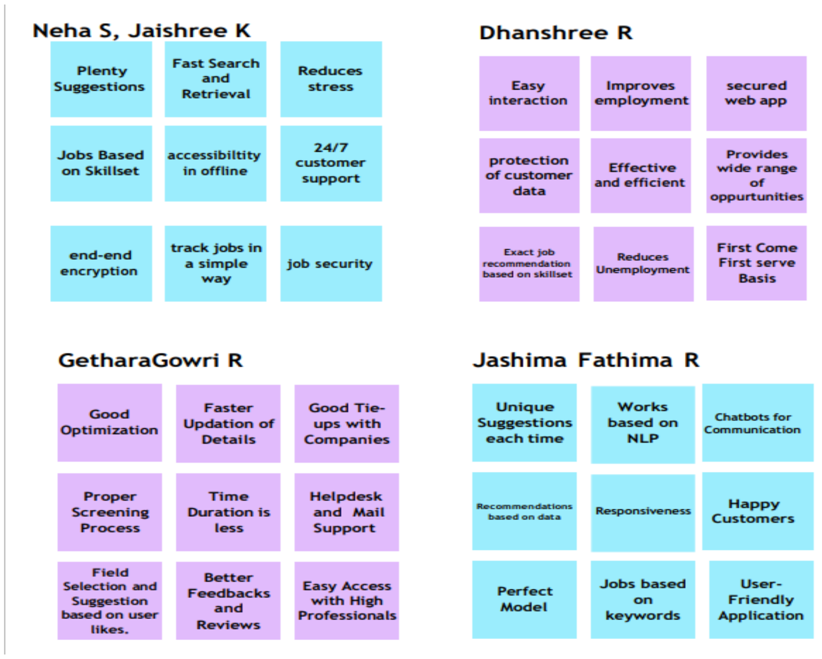
CHAPTER-3

3 IDEATION & PROPOSED SOLUTION

3.1Empathy Map Canvas


it recomend the nearest jobs
It collect require information
Learn skills to succeed
Seprate categories jobs are used for the skill set
Information about the jobs
discover to new opportunity
Trusted
Notification will be send the user mobile application mode
Says
Thinks
Does
Feels
Current jops information
collect the user skill test
Its a trusted application
users feel better using the app
uses the registary system to get the resources
Simplify your job search
Reduse works
Save time
Share your feedback

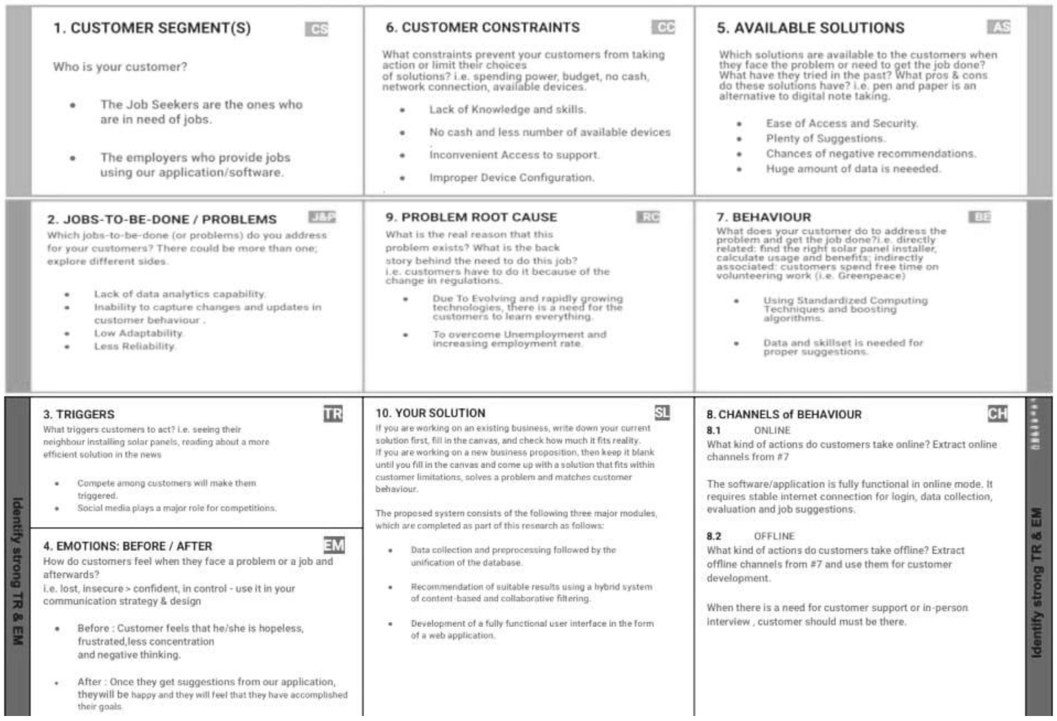
3.2 Ideation & Brainstorming


Neha S, Jaishree K
Plenty Suggestions
Fast Search and Retrieval
Reduces stress
Jobs Based on Skillset
accessibiltity in offline
24/7 customer support
end-end encryption
track jobs in a simple way
job security
Dhanshree R
Easy interaction
Improves employment
secured web app
protection of customer data
Effective and efficient
Provides wide range of oppurtunities
Exact job recommendation based on skillset
Reduces Unemployment
First Come First serve Basis
GetharaGowri R
Good Optimization
Faster Updation of Details
Good Tie-ups with Companies
Proper Screening Process
Time Duration is less
Helpdesk and Mail Support
Field Selection and Suggestion based on user likes.
Better Feedbacks and Reviews
Easy Access with High Professionals
Jashima Fathima R
Unique Suggestions each time
Works based on NLP
Chatbots for Communication
Recommendations based on data
Responsiveness
Happy Customers
Perfect Model
Jobs based on keywords
User-Friendly Application

3.3 Proposed Solution

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	To develop an end-to-end web application which is capable of displaying the job openings based on the student or job seeker skill set.
2.	Idea / Solution description	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 need to interact with the chatbot/webpage so that they get recommendations based on their skills. We can use a job search to get the current job openings in the market.
3.	Novelty / Uniqueness	As of now there is no application for automated skill /job recommender application. Compared to LinkedIn the individuals need to follow and check job opening tabs on all the respective companies to apply for the job.
4.	Social Impact / Customer Satisfaction	It will be helpful for many freshers or the ones who is unemployed

3.4 Problem Solution fit



1. CUSTOMER SEGMENT(S) CS

Who is your customer?

- The Job Seekers are the ones who are in need of jobs.
- The employers who provide jobs using our application/software.

6. CUSTOMER CONSTRAINTS CC

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

- Lack of Knowledge and skills.
- No cash and less number of available devices
- Inconvenient Access to support.
- Improper Device Configuration.

5. AVAILABLE SOLUTIONS AS

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

- Ease of Access and Security.
- Plenty of Suggestions.
- Chances of negative recommendations.
- Huge amount of data is neeeded.

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

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

- Lack of data analytics capability.
- Inability to capture changes and updates in customer behaviour .
- Low Adaptability.
- Less Reliability.

9. PROBLEM ROOT CAUSE RC

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

- Due To Evolving and rapidly growing technologies, there is a need for the customers to learn everything.
- To overcome Unemployment and increasing employment rate.

7. BEHAVIOUR BE

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

- Using Standardized Computing Techniques and boosting algorithms.
- Data and skillset is needed for proper suggestions.

Identify strong TR & EM

3. TRIGGERS TR

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

- Compete among customers will make them triggered.
- Social media plays a major role for competitions.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design

- Before : Customer feels that he/she is hopeless, frustrated,less concentration and negative thinking.
- After : Once they get suggestions from our application, theywill be happy and they will feel that they have accomplished their goals.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

The proposed system consists of the following three major modules, which are completed as part of this research as follows:

- Data collection and preprocessing followed by the unification of the database.
- Recommendation of suitable results using a hybrid system of content-based and collaborative filtering.
- Development of a fully functional user interface in the form of a web application.

8. CHANNELS of BEHAVIOUR CH

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

The software/application is fully functional in online mode. It requires stable internet connection for login, data collection, evaluation and job suggestions.

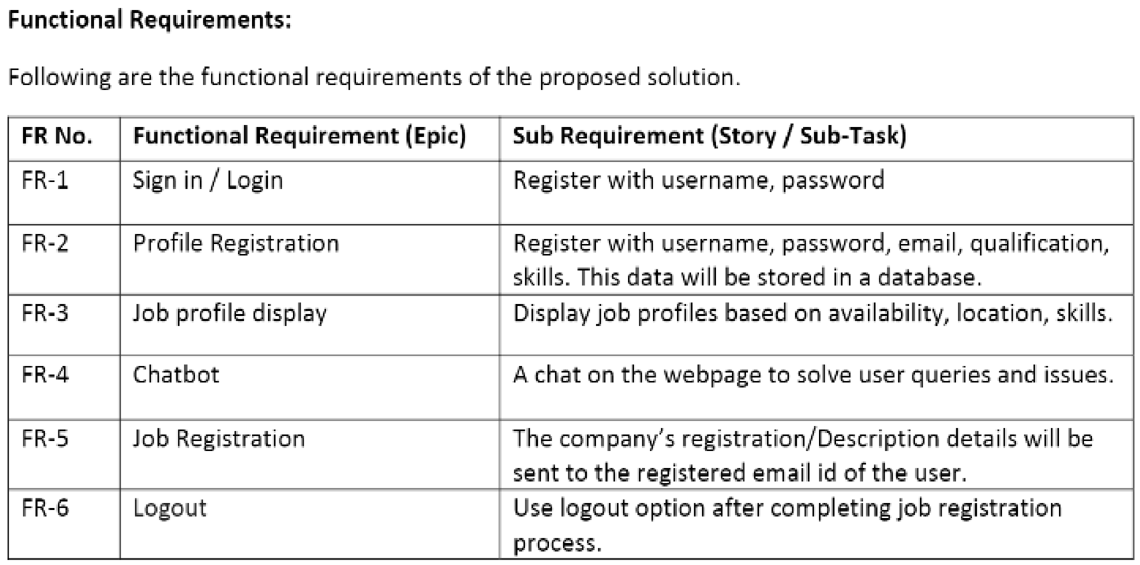
8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

When there is a need for customer support or in-person interview , customer should must be there.

Identify strong TR & EM

CHAPTER-4
4 REQUIREMENT ANALYSIS

4.1Functional 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

RESPONSE TIME:

This web application loads the first time in 1-2 seconds and the second time it is refreshed. When a user logs in, job recommendations are loaded immediately. The search query's quick results are a result of optimised queries. The api's response time is quick. The chatbot doesn't stop answering or freeze.

OPTIMIZATION:

All of the graphics in the web application were compressed to speed up loading. JavaScript optimization was carried out with great care.

RESPONSIVE DESIGN:

This web application is designed to work on a variety of devices and screen sizes. Every element uses Bootstrap classes to adapt to different screen sizes. Different elements are used in some areas for smaller and larger screen sizes.

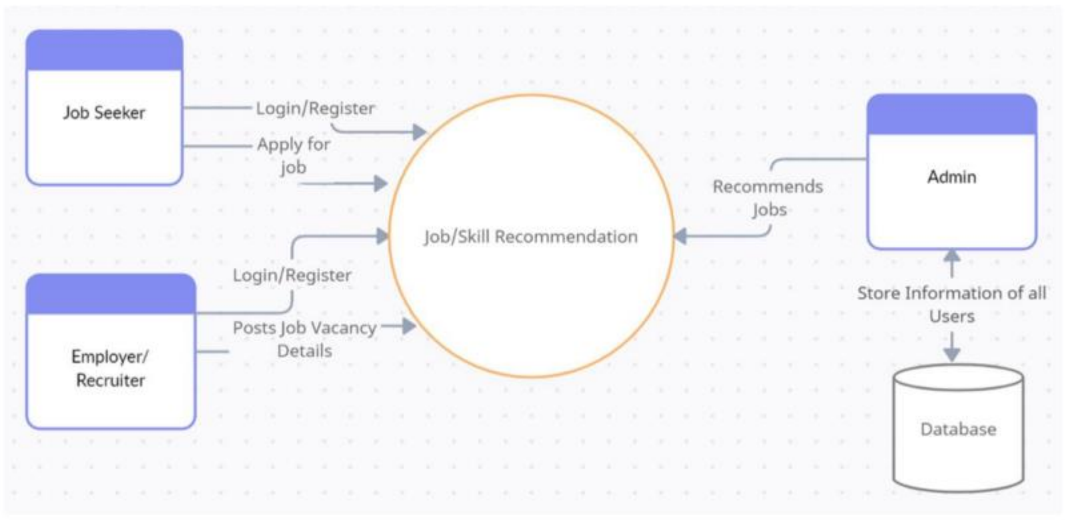
ACCESIBILITY:

To accommodate screen readers and people with impairments, the website's hue is modified, and descriptions of the objects, photographs, and other content are provided.

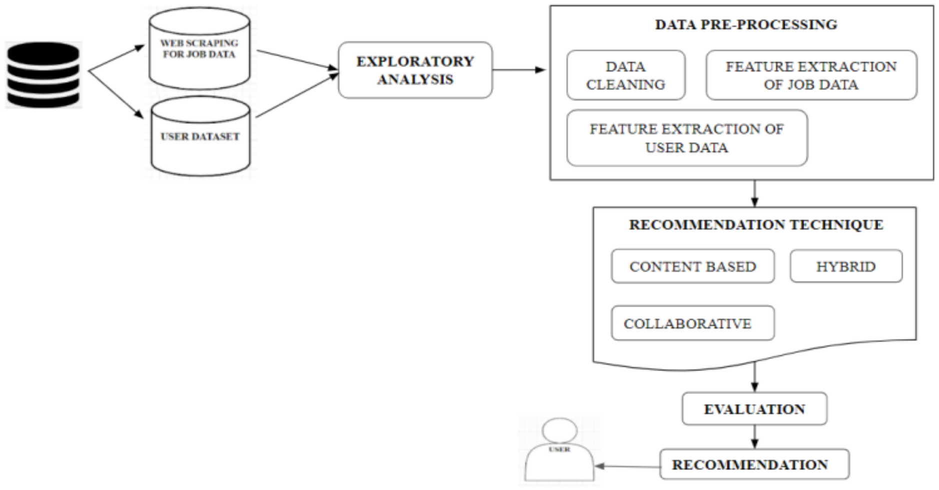
CHAPTER -5
PROJECT DESIGN

5.1 Data Flow Diagrams

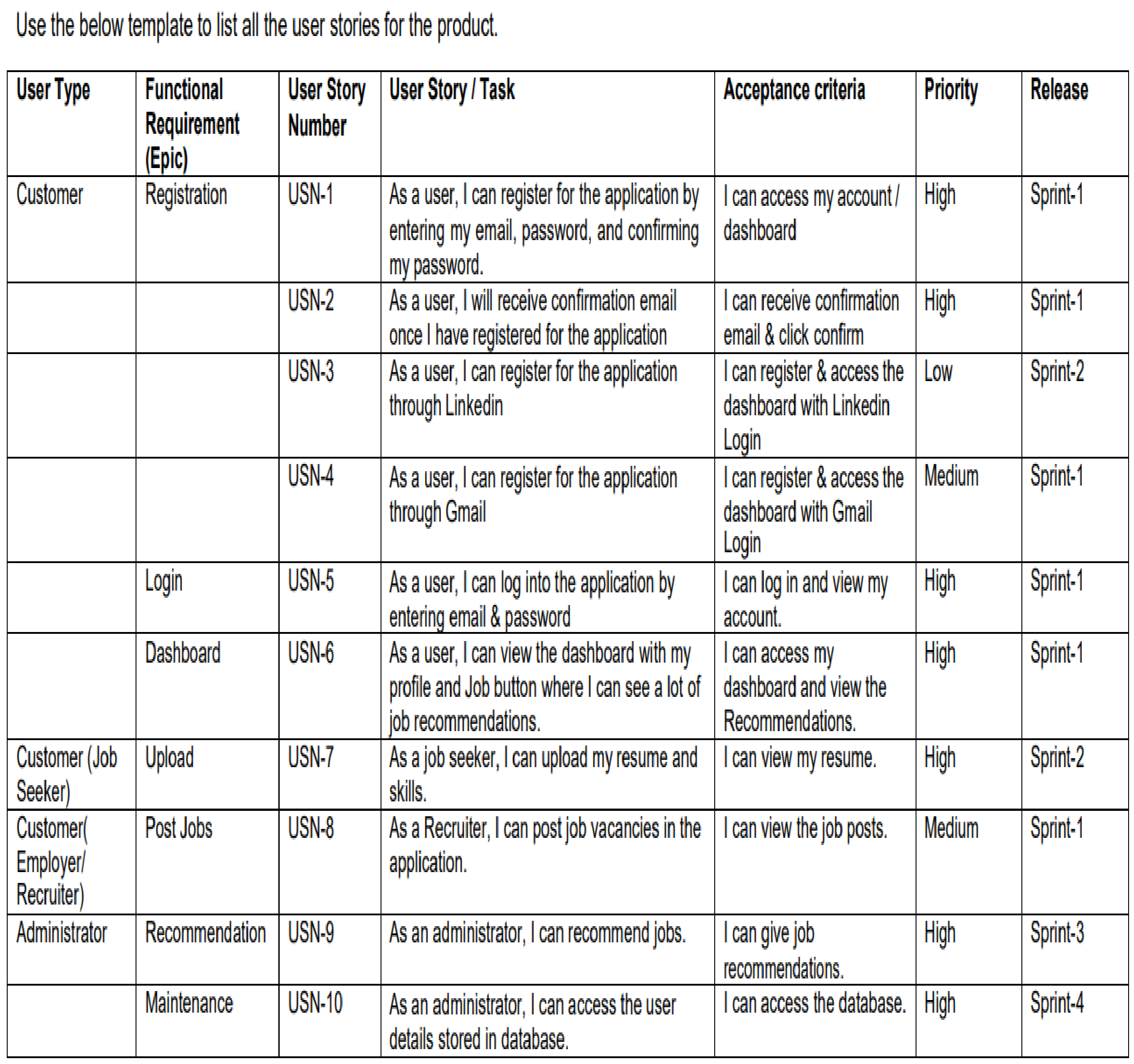
A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement.



5.2 Solution & Technical Architecture:



5.3 User Stories



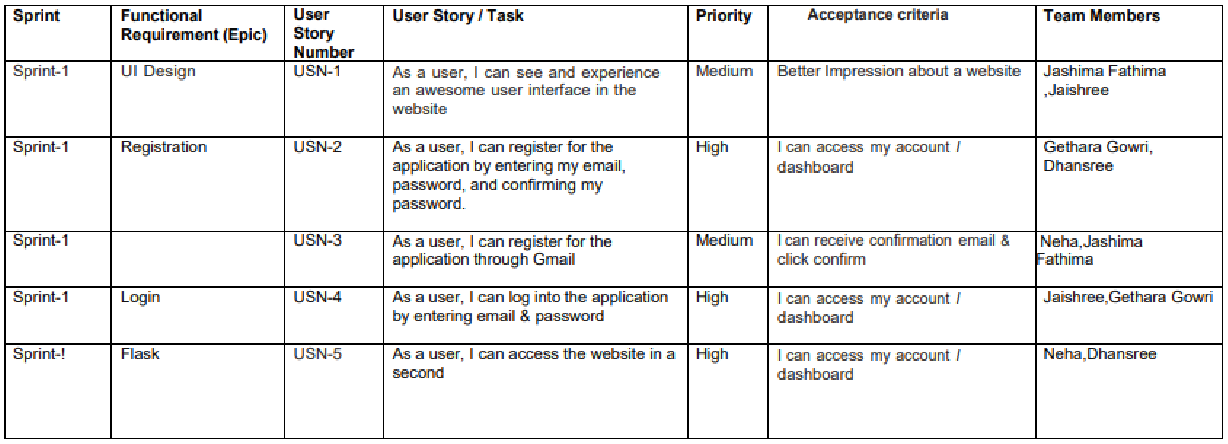
Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	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 Linkedin	I can register & access the dashboard with Linkedin Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can log in and view my account.	High	Sprint-1
	Dashboard	USN-6	As a user, I can view the dashboard with my profile and Job button where I can see a lot of job recommendations.	I can access my dashboard and view the Recommendations.	High	Sprint-1
Customer (Job Seeker)	Upload	USN-7	As a job seeker, I can upload my resume and skills.	I can view my resume.	High	Sprint-2
Customer(Employer/ Recruiter)	Post Jobs	USN-8	As a Recruiter, I can post job vacancies in the application.	I can view the job posts.	Medium	Sprint-1
Administrator	Recommendation	USN-9	As an administrator, I can recommend jobs.	I can give job recommendations.	High	Sprint-3
	Maintenance	USN-10	As an administrator, I can access the user details stored in database.	I can access the database.	High	Sprint-4

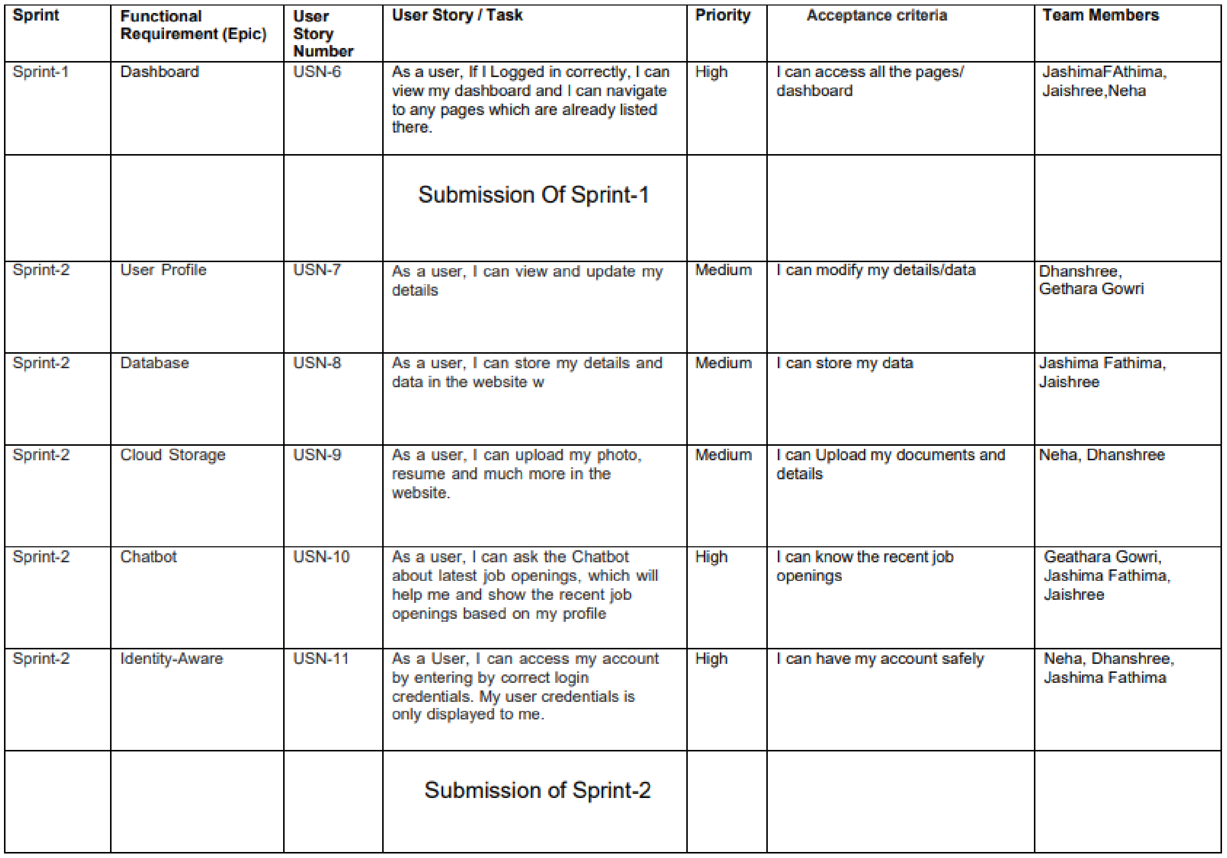
CHAPTER-6

6 PROJECT PLANNING & SCHEDULING

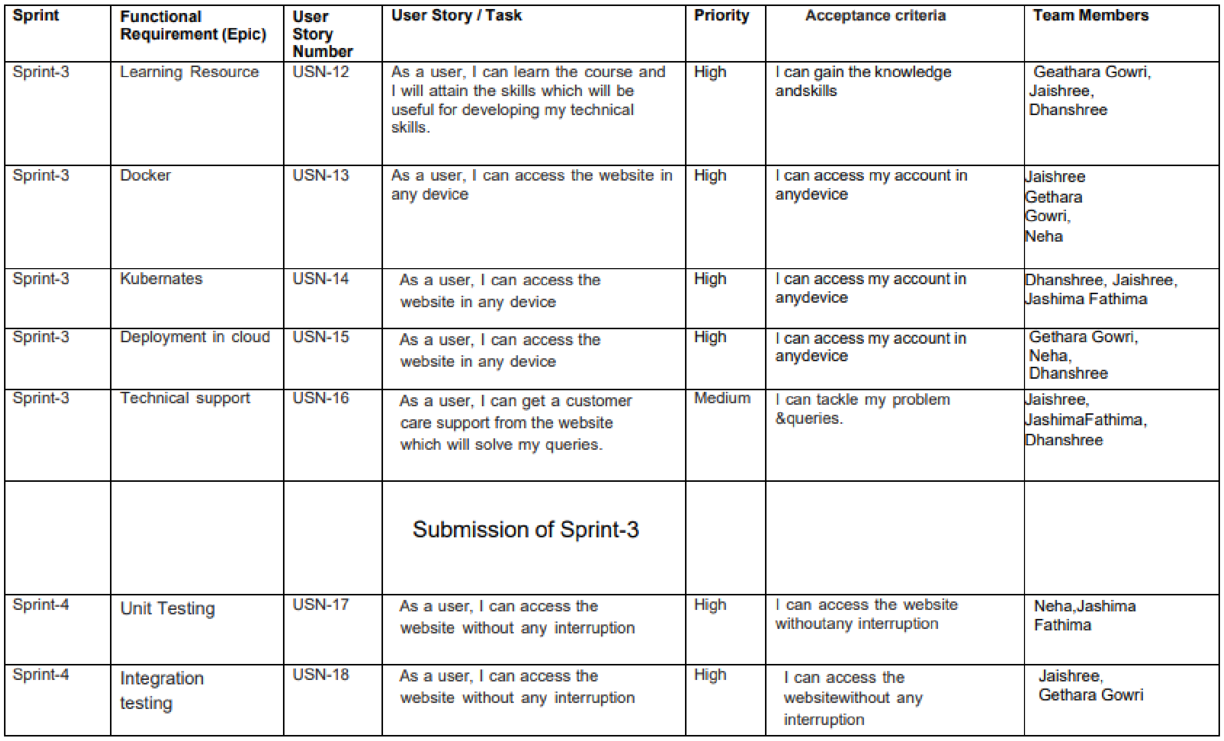
6.1 Sprint Planning & Estimation



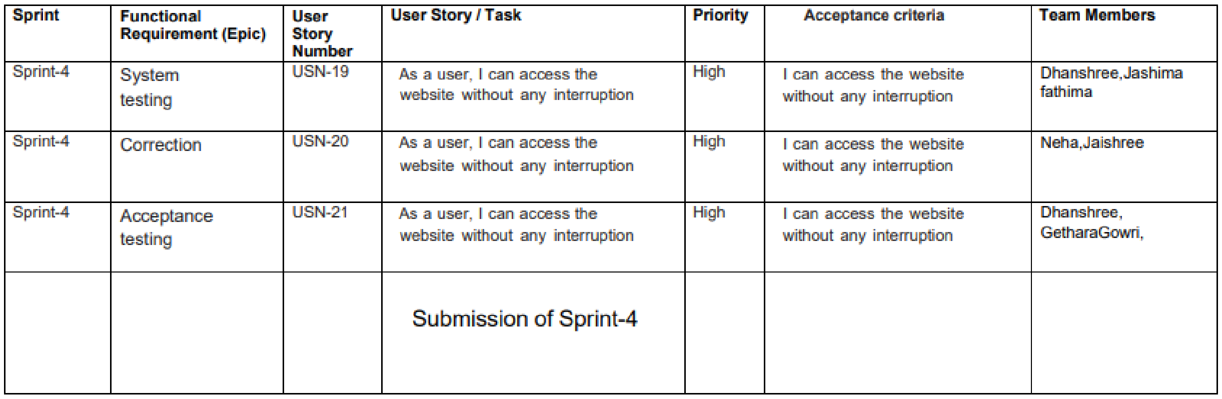
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Acceptance criteria	Team Members
Sprint-1	UI Design	USN-1	As a user, I can see and experience an awesome user interface in the website	Medium	Better Impression about a website	Jashima Fathima ,Jaishree
Sprint-1	Registration	USN-2	As a user, I can register for the application by entering my email, password, and confirming my password.	High	I can access my account / dashboard	Gethara Gowri, Dhansree
Sprint-1		USN-3	As a user, I can register for the application through Gmail	Medium	I can receive confirmation email & click confirm	Neha,Jashima Fathima
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	High	I can access my account / dashboard	Jaishree,Gethara Gowri
Sprint-!	Flask	USN-5	As a user, I can access the website in a second	High	I can access my account / dashboard	Neha,Dhansree



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Acceptance criteria	Team Members
Sprint-1	Dashboard	USN-6	As a user, If I Logged in correctly, I can view my dashboard and I can navigate to any pages which are already listed there.	High	I can access all the pages/ dashboard	JashimaFAthima, Jaishree,Neha
			Submission Of Sprint-1			
Sprint-2	User Profile	USN-7	As a user, I can view and update my details	Medium	I can modify my details/data	Dhanshree, Gethara Gowri
Sprint-2	Database	USN-8	As a user, I can store my details and data in the website w	Medium	I can store my data	Jashima Fathima, Jaishree
Sprint-2	Cloud Storage	USN-9	As a user, I can upload my photo, resume and much more in the website.	Medium	I can Upload my documents and details	Neha, Dhanshree
Sprint-2	Chatbot	USN-10	As a user, I can ask the Chatbot about latest job openings, which will help me and show the recent job openings based on my profile	High	I can know the recent job openings	Geathara Gowri, Jashima Fathima, Jaishree
Sprint-2	Identity-Aware	USN-11	As a User, I can access my account by entering by correct login credentials. My user credentials is only displayed to me.	High	I can have my account safely	Neha, Dhanshree, Jashima Fathima
			Submission of Sprint-2			



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Acceptance criteria	Team Members
Sprint-3	Learning Resource	USN-12	As a user, I can learn the course and I will attain the skills which will be useful for developing my technical skills.	High	I can gain the knowledge andskills	Geathara Gowri, Jaishree, Dhanshree
Sprint-3	Docker	USN-13	As a user, I can access the website in any device	High	I can access my account in anydevice	Jaishree Gethara Gowri, Neha
Sprint-3	Kubernates	USN-14	As a user, I can access the website in any device	High	I can access my account in anydevice	Dhanshree, Jaishree, Jashima Fathima
Sprint-3	Deployment in cloud	USN-15	As a user, I can access the website in any device	High	I can access my account in anydevice	Gethara Gowri, Neha, Dhanshree
Sprint-3	Technical support	USN-16	As a user, I can get a customer care support from the website which will solve my queries.	Medium	I can tackle my problem &queries.	Jaishree, JashimaFathima, Dhanshree
			Submission of Sprint-3			
Sprint-4	Unit Testing	USN-17	As a user, I can access the website without any interruption	High	I can access the website withoutany interruption	Neha,Jashima Fathima
Sprint-4	Integration testing	USN-18	As a user, I can access the website without any interruption	High	I can access the websitewithout any interruption	Jaishree, Gethara Gowri



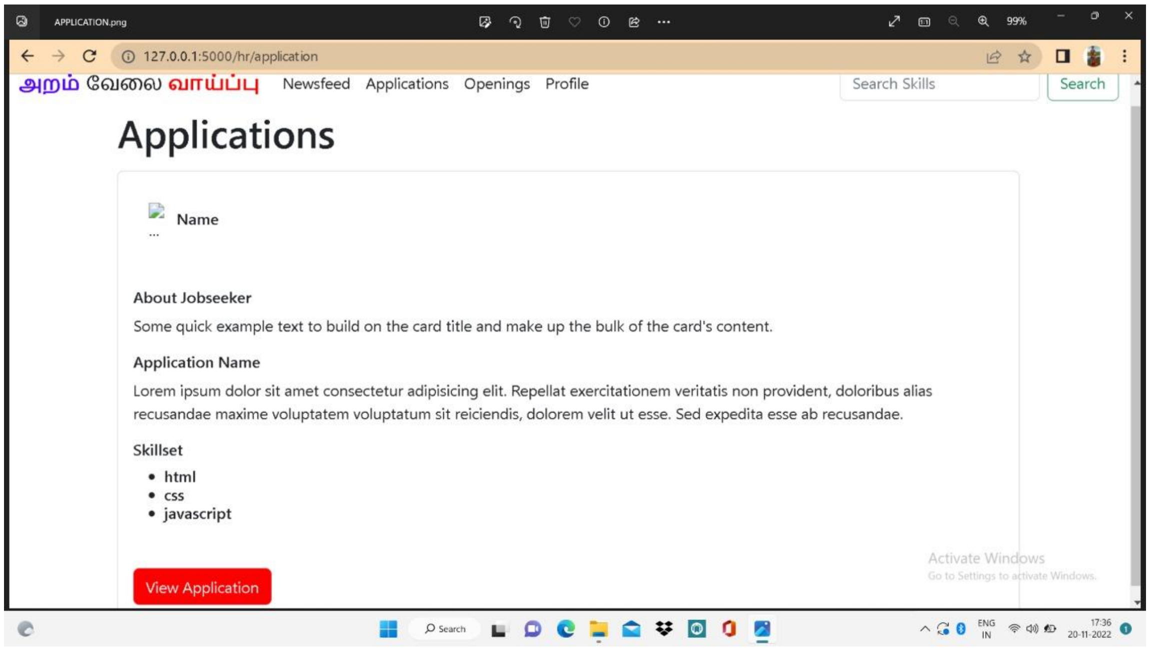
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Acceptance criteria	Team Members
Sprint-4	System testing	USN-19	As a user, I can access the website without any interruption	High	I can access the website without any interruption	Dhanshree,Jashima fathima
Sprint-4	Correction	USN-20	As a user, I can access the website without any interruption	High	I can access the website without any interruption	Neha,Jaishree
Sprint-4	Acceptance testing	USN-21	As a user, I can access the website without any interruption	High	I can access the website without any interruption	Dhanshree, GetharaGowri,
			Submission of Sprint-4			

6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	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

6.3 Reports from JIRA



APPLICATION.png

127.0.0.1:5000/hr/application

அறம் வேலை வாய்ப்பு Newsfeed Applications Openings Profile

Applications

Name

About Jobseeker

Some quick example text to build on the card title and make up the bulk of the card's content.

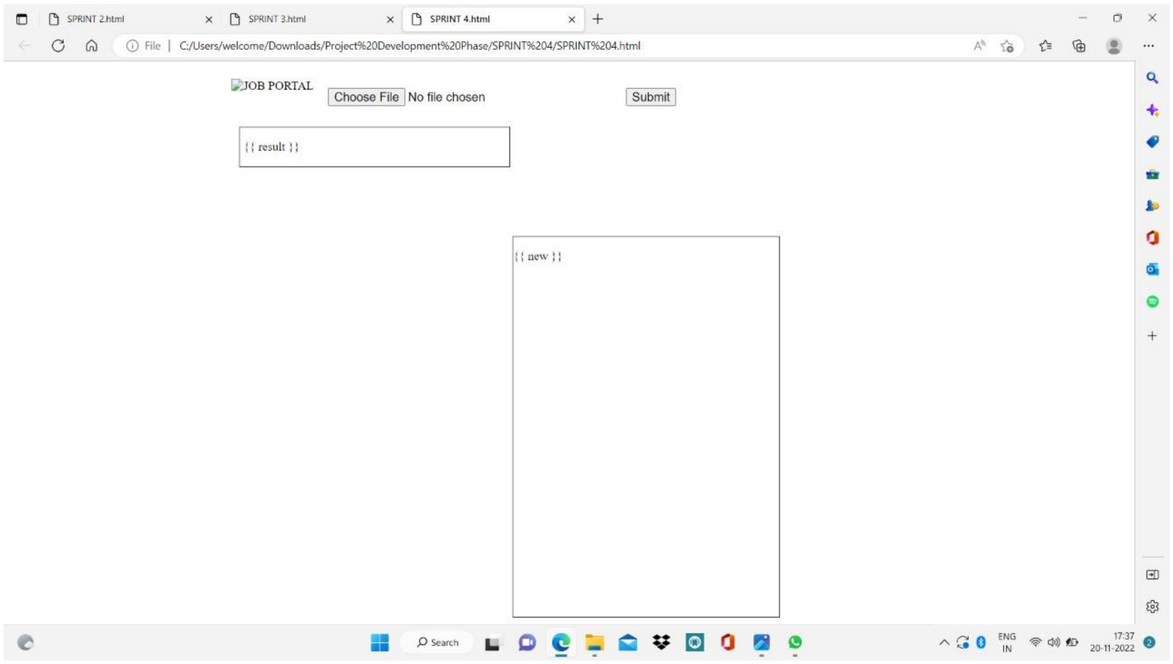
Application Name

Lorem ipsum dolor sit amet consectetur adipisicing elit. Repellat exercitationem veritatis non provident, doloribus alias recusandae maxime voluptatem voluptatum sit reiciendis, dolorem velit ut esse. Sed expedita esse ab recusandae.

Skillset

- html
- css
- javascript

View Application



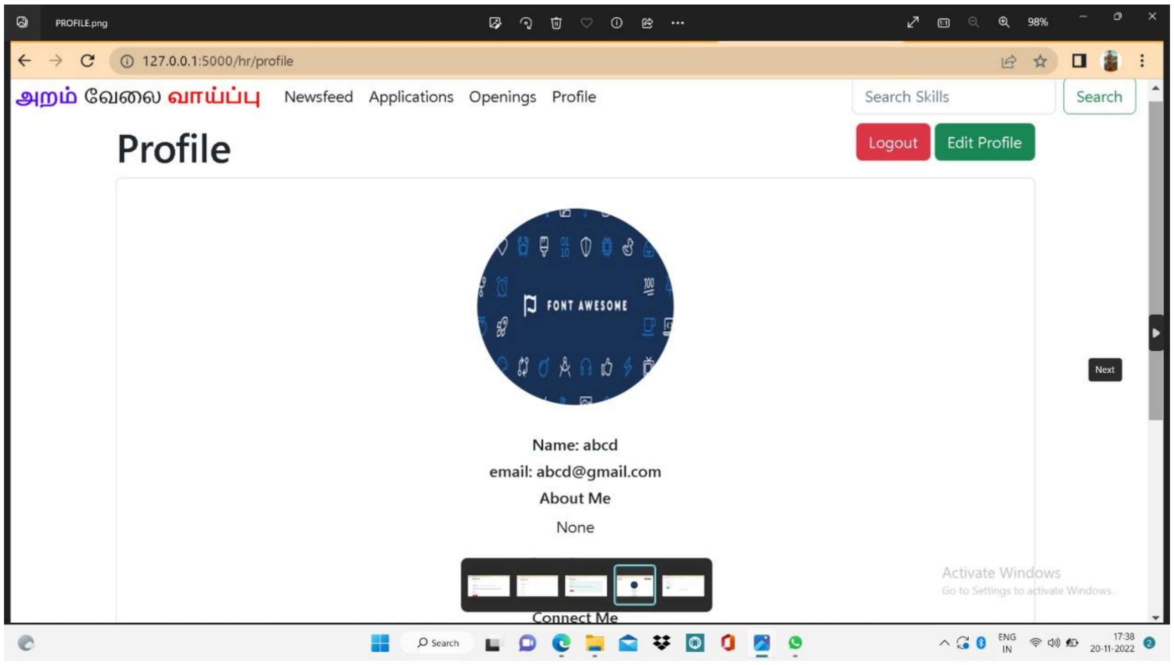
SPRINT 2.html SPRINT 3.html SPRINT 4.html

C:/Users/welcome/Downloads/Project%20Development%20Phase/SPRINT%204/SPRINT%204.html

JOB PORTAL Choose File No file chosen Submit

{ { result } }

{ { new } }



PROFILE.png

127.0.0.1:5000/hr/profile

அறம் வேலை வாய்ப்பு Newsfeed Applications Openings Profile Search Skills Search

Profile

Logout Edit Profile

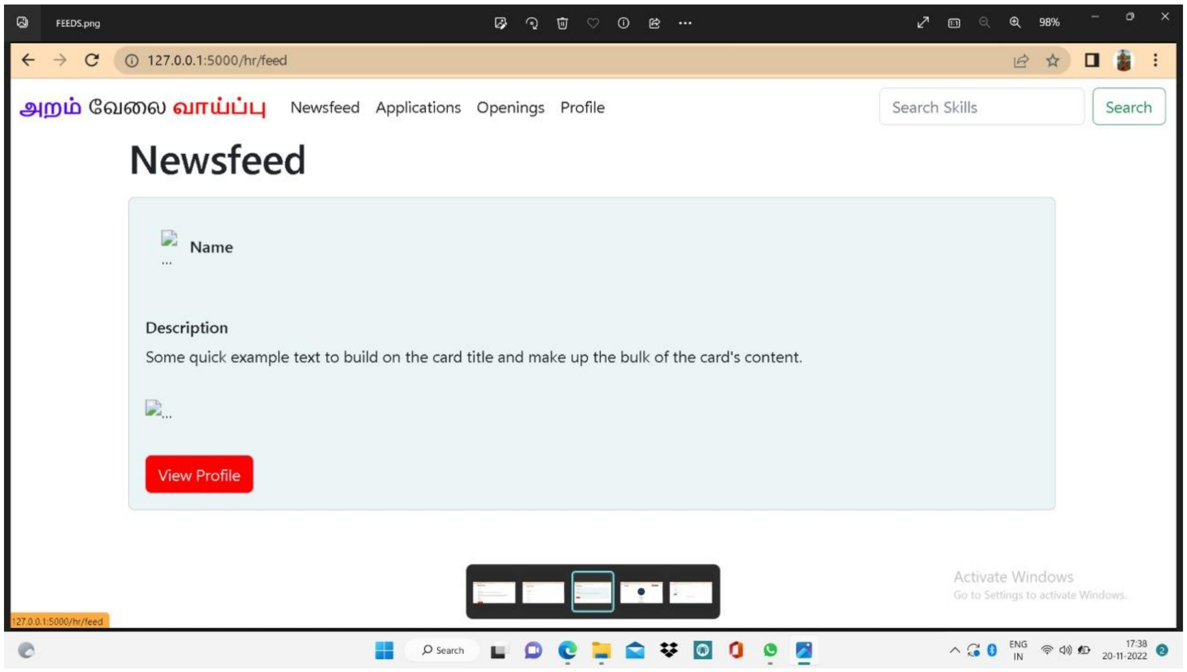
Name: abcd

email: abcd@gmail.com

About Me

None

Connect Me



FEEDS.png

127.0.0.1:5000/hr/feed

அறம் வேலை வாய்ப்பு Newsfeed Applications Openings Profile Search Skills Search

Newsfeed

Name

Description

Some quick example text to build on the card title and make up the bulk of the card's content.

View Profile

CHAPTER -7
7 CODING & SOLUTIONING

7.1 Feature 1

The software has an In-built “Chat Bot” which can help assist with ongoing queries and provide fast and effective solutions to user problems which may occur and also redirect to management attention if need be there any complications the customer service will be available 24*7 to assist in case of any controversial issues arise. The pages we have:

i. Login page
ii. Registeration page
iii. Profile page
iv. Job recommender page

7.2 Feature 2

In this project we have created the dashboard page to view the jobs available and to make ease to access the website

- They communicate information quickly.
- They display information clearly and efficiently.
- They show trends and changes in data over time.
- They are easily customizable.
- The most important widgets and data components are effectively presented in a limited space.

CHAPTER-8

TESTING

8.1 Test Cases

The practise of analysing and validating that a software product or programme accomplishes what it is designed to do is known as software testing. Testing has advantages such as bug prevention, lower development costs, and better performance. This software has undergone successful testing and evaluation.

8.2 User Acceptance Testing

Goal of the Document: This document's goal is to provide a concise explanation of the project's test coverage and open issues as of the time of User Acceptance Testing (UAT)

Acceptance by users: The testing is done in a different testing environment. A modification, an upgrade, or a new feature is asked for and created. There are both unit and integration tests. Things appear to be in order. But after it is made available to the public, significant issues start to emerge. When that occurs, rework and retesting are not the most costly outcomes.

1. Defect Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
ClientApplication	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3

This report shows the number of resolved or closed bugs ateach severity level, and howthey were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
ByDesign	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'tFix	0	5	2	1	8
Totals	24	14	13	26	77

3. Test Case Analysis

Thisreport showsthenumberoftestcasesthathavepassed,failed,anduntested

ExceptionReporting	9	0	0	9
FinalReportOutput	4	0	0	4
VersionControl	2	0	0	2

CHAPTER-9

RESULTS

9.1PERFORMANCE METRICS:

A performance metric assesses an organization's behaviour, operations, and performance. It should accommodate the needs of a variety of stakeholders, including customers, shareholders, and staff. Metrics may also be focused on performance in relation to client requirements and value, even if traditionally many metrics are finance-based and internally focused on the success of the organisation. Performance metrics are used in project management to evaluate the state of the project and consist of assessing seven criteria: safety, time, money, resources, scope, quality, and actions.

It usually involves the following steps to develop performance metrics:

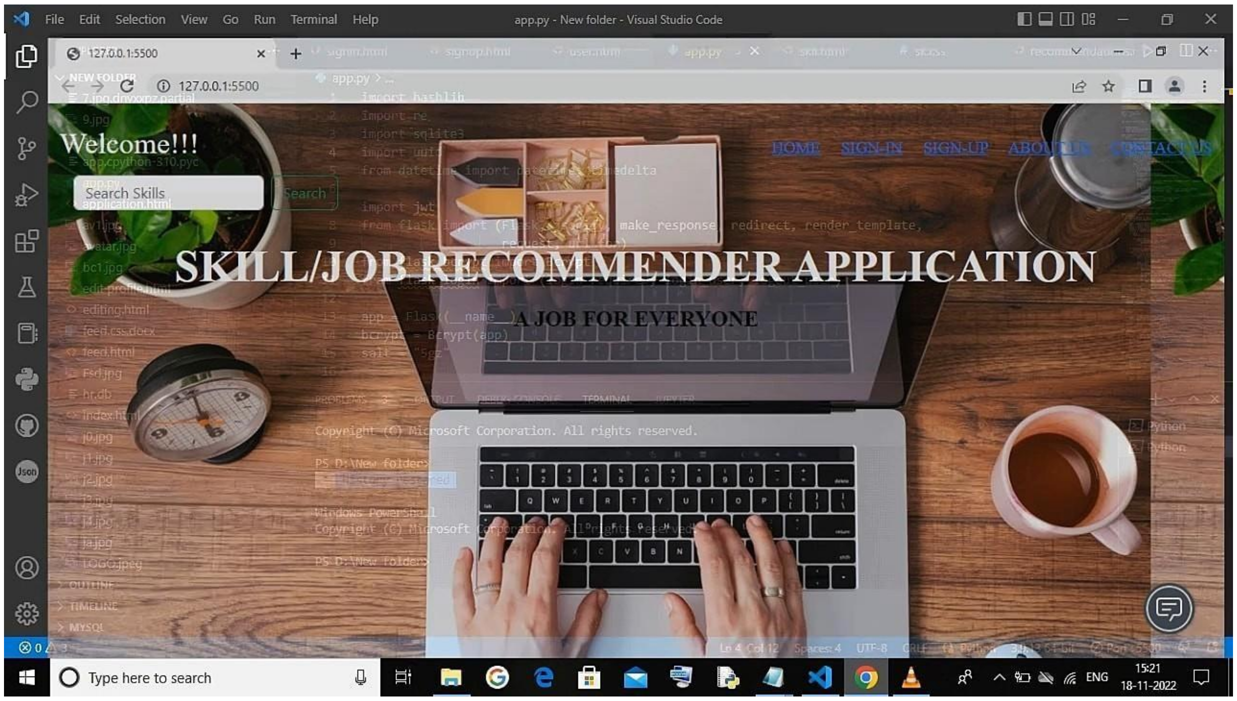
- defining important process and customer needs
- identifying precise, measurable work results.
- establishing goals to measure performance against

The average cosine similarity of every item in a list of suggestions is known as intra-list similarity. This computation determines the degree of similarity by using characteristics of the suggested items, such as movie genre. The list of suggested items' diversity can be measured using this statistic.

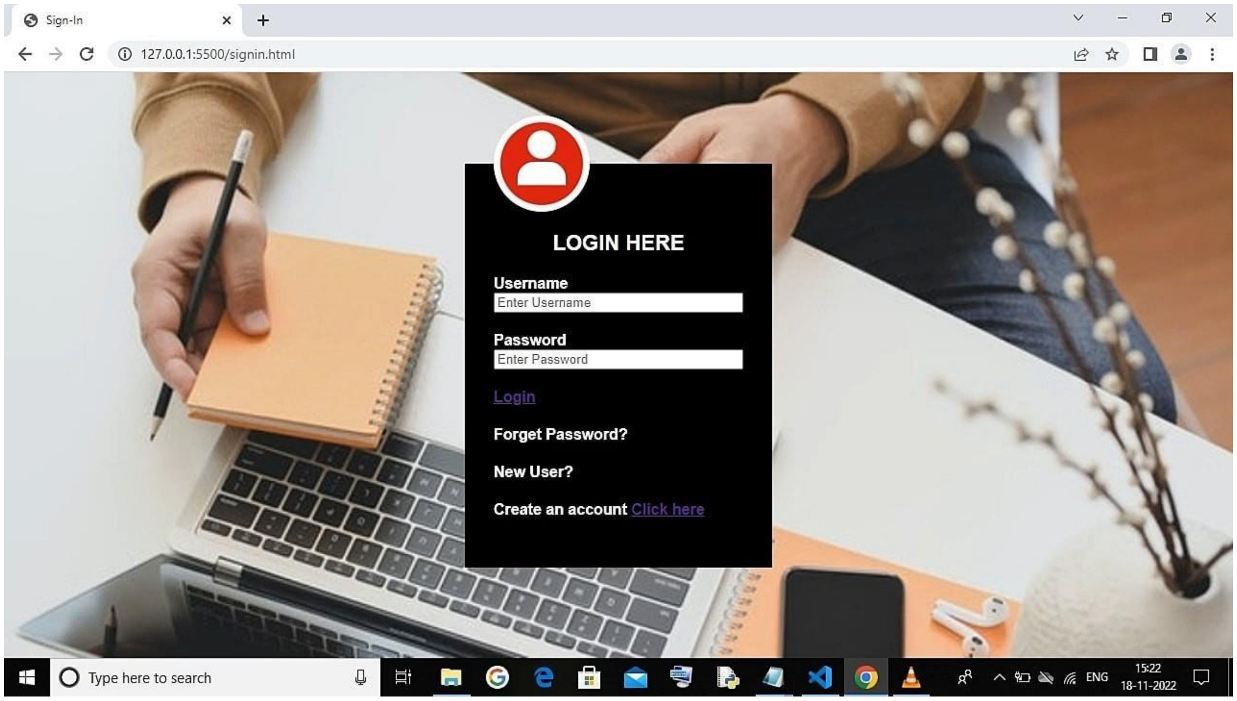
Parameters: a ranking of the best recommendation feature df is a dataframe that contains the skill feature.

Returns: The average similarity between lists of suggestions

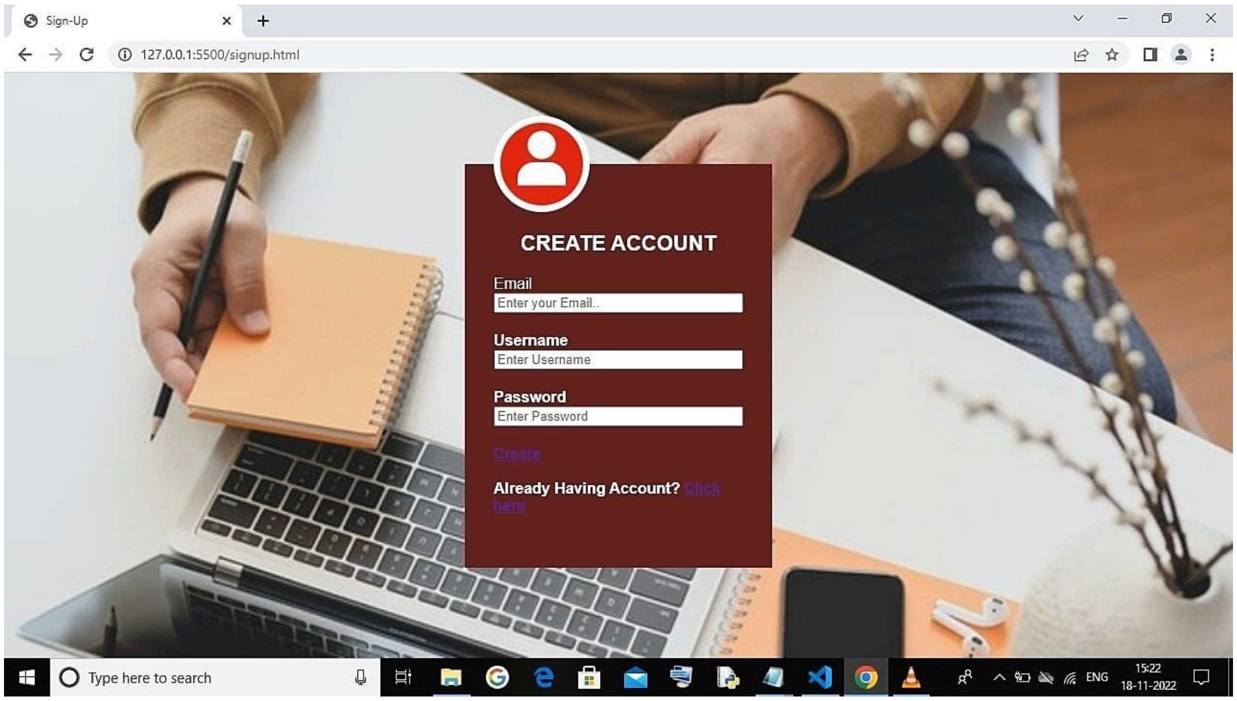
9.2 HOME PAGE:



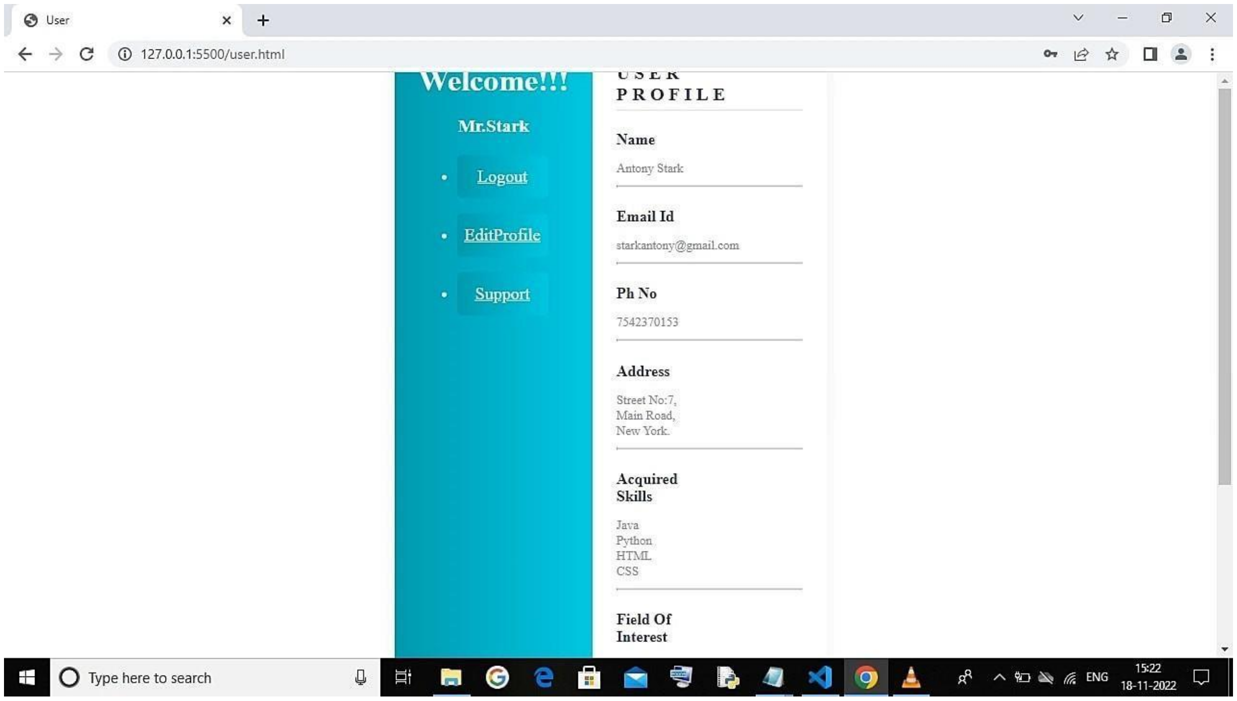
9.3 LOGIN PAGE:



9.4 SIGN UP PAGE:



9.5 USER-PROFILE PAGE:



Name
Antony Stark

Email Id
starkantony@gmail.com

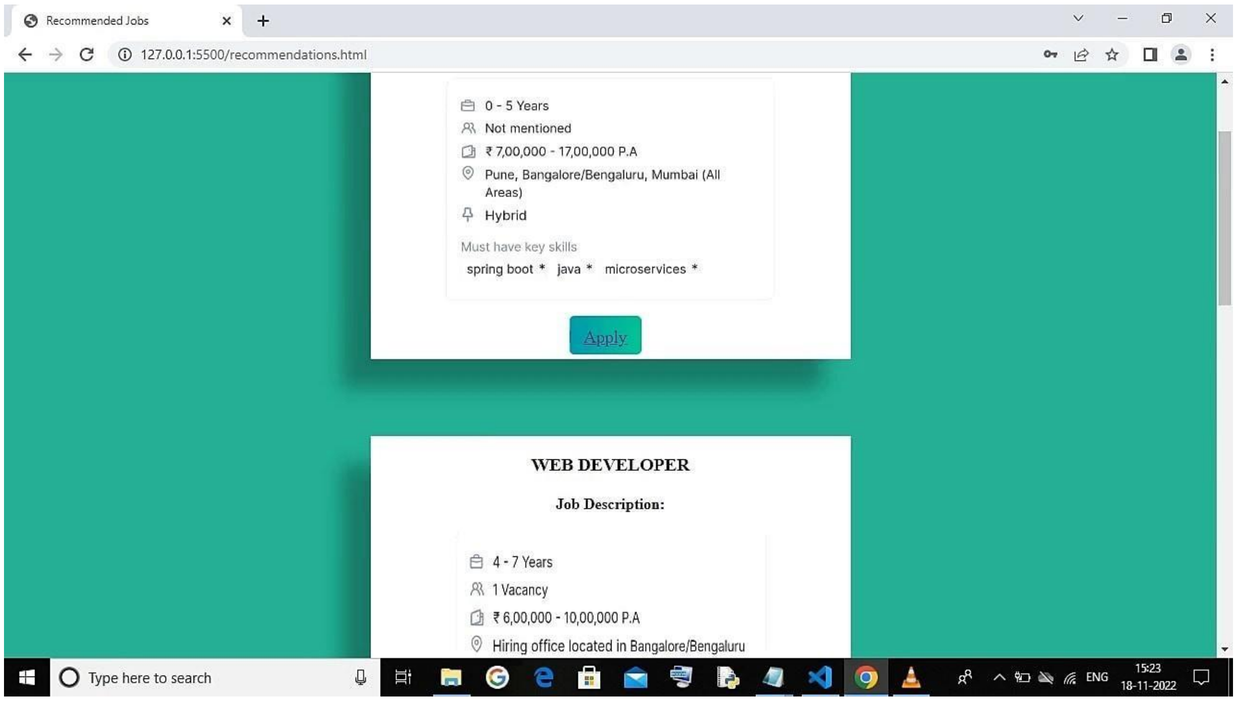
Ph No
7542370153

Address
Street No:7,
Main Road,
New York.

Acquired Skills
Java
Python
HTML
CSS

Field Of Interest

9.6 JOB RECOMMENDER PAGE:



Recommended Jobs

127.0.0.1:5500/recommendations.html

0 - 5 Years

Not mentioned

₹ 7,00,000 - 17,00,000 P.A

Pune, Bangalore/Bengaluru, Mumbai (All Areas)

Hybrid

Must have key skills

spring boot * java * microservices *

Apply

WEB DEVELOPER

Job Description:

4 - 7 Years

1 Vacancy

₹ 6,00,000 - 10,00,000 P.A

Hiring office located in Bangalore/Bengaluru

CHAPTER-10
ADVANTAGES AND DISADVANTAGES

10.1 ADVANTAGES

- Since the model's recommendations are tailored to one user, it doesn't require any information about other users.
- This enables scaling to a big number of users easier
- The model may identify a user's individual preferences and recommend specialized goods in which only a small percentage of other users are also interested.

10.2 DISADVANTAGES:

- This technique necessitates extensive domain knowledge because the feature representation of the items is to some extent hand-engineered. The quality of the model is therefore limited to the hand-engineered elements
- The model can only give recommendations based on the user's existing interests
- In other words, the model's potential to build on the users' existing interests is limited.

CHAPTER-11

CONCLUSION

This experiment compared Content-Based Filtering versus Collaborative Filtering of recommendations. In addition, an aggregation and recommender system has been developed.

The results of Content-Based Filtering are recommended based on matching the user's own preferences with the given material, whereas collaborative filtering is recommended based on the preferences of fellow users. After examining both systems, it was determined that a hybrid system of both overcomes the constraints of both and improves ranking efficiency. Cold start, sparse database, scalability, and trend suggestion issues have been resolved. The proposal is to create a job recommendation system that values quality over quantity. While there are websites and job listing portals that already propose jobs to job searchers based on their profiles, this research on aggregate quality recommendations was accomplished by crawling selectively, overcoming the restrictions. A fully functional user interface was created to integrate everything and provide the user with a seamless experience.

CHAPTER -12

FUTURE SCOPE

There are numerous opportunities to enhance this online application. The API interface can be improved with additional parameters to enable multi-parameter search and provide the user's information more specifically.The chatbot can be developed to sound more human-like and have more intents and conversations.More learning resources, webapp notifications, chat capabilities, and even a feed post feature where job searchers can mingle can all be added to the graphical user experience.The employer interface can be improved with tools like candidate ranking, graphic job posting, custom dashboards with metrics, and direct user interaction.

CHAPTER 13

APPENDIX

Source Code:

App.py:

```
 import hashlib
 import re
 import sqlite3
 import uuid
 from datetime import datetime, timedelta
 import jwt from flask import (Flask, jsonify, make_response, redirect,
 render_template, request, url_for)
 from flask_bcrypt import Bcrypt
  from flask_login import (login_required, login_user, logout_user)
 app = Flask(_name_)
 bcrypt = Bcrypt(app)
 salt = "5gz"
  app.config["KEY"] = "Hello" def verify(token):
     data = jwt.decode(token, "Hello", algorithms='HS256')
 return data["email"] @app.route('/') def home():
     return render_template('./sign/hrsignin.html')
 @app.route("/hr/signin", methods=['GET', 'POST']) def
 hrSignIn():
     if request.method == "GET":
         return render_template("./sign/hrsignin.html")
     else:
         email = request.form["email"] password =
         request.form["password"] with
         sqlite3.connect('hr.db') as connection:
             cursor = connection.cursor() cursor.execute(
"SELECT email FROM RECRUITER WHERE email=?", (email,)) user

             = cursor.fetchone() if user
             == None: print("No user")
```

```
            return
            redirect("/hr/profile")
            else:
                db_password    =    password+salt    pw_hash    =
                hashlib.md5(db_password.encode())
                cursor.execute(
                        "SELECT email,password FROM RECRUITER WHERE email=?",

(email,)
  )             details = cursor.fetchone()
                print(details)if
                pw_hash.hexdigest() == details[1]:
                    token    =    jwt.encode({"email":    email,    'exp':    datetime.utcnow(
                    )+timedelta(minutes=30)},    "Hello",    algorithm='HS256')    print(token)
                    response = make_response(
                        render_template("./feed/feed.html"))
                    response.set_cookie('token', token)
                return response else:
                    return "wrong password"
@app.route("/hr/signup",   methods=["GET",   "POST"])
def hrSignUp():
    if request.method == "GET":
        return render_template("./sign/hrsignup.html")
    else:
        name = request.form["name"] email =
        request.form["email"]       phone       =
        request.form["phone"]    password    =
        request.form["password"]   confirm  =
        request.form["re-password"] if email:
            regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b' def check(email):
                if (re.fullmatch(regex, email)):
                    print("valid email")
                else:
                    return "not an valid email"
        if password != confirm:
            return "Password mismatch"
        else:
            with sqlite3.connect('hr.db') as connection:
                cursor = connection.cursor() cursor.execute(
                    """ SELECT email FROM RECRUITER WHERE email=? """, (email,))
                user = cursor.fetchone() print(user) if user == None:
                    key       =       uuid.uuid1().hex
                    print(key)


                    db_password = password+salt
                    pw_hash  =  hashlib.md5(db_password.encode())  cursor.execute("INSERT
                    INTO RECRUITER
(name,email,phone,password,id)   VALUES   (?,?,?,?,?)",   (   name,
                        email, phone, pw_hash.hexdigest(), key))
                    connection.commit()
                    return redirect("/hr/signin")
                else:
                    print("exists")    return
                    redirect("/hr/signin")
@app.route("/hr/logout")
def logout():
    response  =  make_response(render_template("./sign/hrsignin.html"))
response.set_cookie('token',  '')  return  response  @app.route('/hr/feed')
def hrFeed(): try:
        token      =      request.cookies.get('token')      data      =
        jwt.decode(token, "Hello", algorithms='HS256') return
        render_template("./feed/feed.html")
    except:
        return render_template("./feed/feed.html")
@app.route("/hr/feed/<i
d>") def hrOneFeed(id):
try:
        token      =      request.cookies.get('token')      data      =
        jwt.decode(token,      "Hello",      algorithms='HS256')
        print(id)                                              return
        render_template("./feed/oneFeed.html")
    except:
        return redirect("/hr/signin")
```

```
@app.route("/hr/applica
tion") def
hrApplication(): try:
        token = request.cookies.get('token') email =
        verify(token) print(email) return
        render_template("./application/applications.html")
    except:
       return render_template("./application/applications.html")

@app.route("/hr/application/<id
>") def hrOneApplication(id): try:
        token = request.cookies.get('token') email = verify(token)
        print(email)                                       return
        render_template("./application/oneApplication.html")
    except:
        return
redirect("/hr/signin")
@app.route("/hr/profile")
def hrProfile():
    try:
        token = request.cookies.get('token') email
        = verify(token) print(email) with
        sqlite3.connect('hr.db') as connection:
            cursor=connection.curso
            r() cursor.execute("""
            SELECT name, email,
            about_me, designation,
            experience ,
            url ,
            company_name ,
            company_descrip
            tion , location ,
            website , in_url
             FROM RECRUITER WHERE email=?""",
            (email,)) data=cursor.fetchone() print(data) if
            not data:
                return redirect("/hr/logout")
            else:
            return render_template("./profile/viewProfile.html",data=data)

    except Exception as e:
        print(e) return
redirect("/hr/signin")
@app.route("/hr/profile/ed
it") def hrProfileEdit(): try:
        token = request.cookies.get('token') email
        =      verify(token)      print(email)      with
        sqlite3.connect('hr.db') as connection:
            cursor=connection.curso
            r() cursor.execute("""
            SELECT name, email,
            about_me, designation,
            experience ,
            url ,
            company_name     ,
            company_descript
            ion,      location      ,
            website , in_url , id
             FROM      RECRUITER      WHERE      email=?""",
            (email,))                         data=cursor.fetchone()
            print(data[11]) if not data:
                return redirect("/hr/logout")
            else:
              return render_template("./profile/editProfile.html",data=data)

        return render_template("./profile/editProfile.html")
    except:
        return redirect("/hr/signin")
@app.route("/hr/profile/edit/<id>",methods=("POST","GET"))
def profileEditIID(id):
    if request.method=="POST":
        token   =   request.cookies.get('token')   print("post")
        try:
            print(email) email = verify(token)
            name=request.form["name"],
            about_me=request.form["about_me"],
            designation=request.form['designation'],
            experience=request.form['experience'],
            url=request.form['url'],
            company_name=request.form['company_name'
            ],
            company_description=request.form['company_
            description'], location
            =request.form['location'],
            website=request.form['website'],
            in_url=request.form['in_url'] , if not id:
                return redirect("/hr/profile")
            with sqlite3.connect('hr.db') as connection:
                cursor=connection.cursor() cursor.execute("""SELECT id
                FROM RECRUITER WHERE
    email=?""",(email,))
                if
                data[11]
                ==id:
                    cursor.exec
                ute("""
                SELECT  name,
                email,
                about_me,
                designation,
                experience ,
                url ,
                company_name ,
                company_descrip
                tion , location ,
                website , in_url ,
                id
                FROM RECRUITER WHERE email=?""",
                (email,)) data=cursor.fetchone()
                cursor.execute("""
                    UPDATE RECRUITER SET
                    name=?, about_me=?,
                    designation=?, experience=?,
                    url=?,
                    company_name=?    ,
                    company_descriptio
                    n=?   ,   location   =?,
                    website=?, in_url=? ,
                     FROM RECRUITER WHERE email=?""",
                    (name, about_me, designation, experience,
                    url,
                    company_name,
                    company_descrip
                    tion    ,    location,
                    website,      in_url
                    ,email))
                  connection.commit()
                  return "Success"
              except Exception as e:
                print(e)
                return
                "failed"
          @app.route("/hr/profile/pwd", methods=("GET", "POST")) def
          hrProfileEditPWD():
          if request.method == "GET":
              try:
                token = request.cookies.get('token') email =
              verify(token) print(email) return
              render_template("./profile/passwordReset.html") except:
                return redirect("/hr/signin")
            else:
              try:
                token  =  request.cookies.get('token')  email  =
                verify(token)     print(email)     password     =
                request.form["password"]       newPWD       =
                request.form['newPassword']  confirmPWD  =
                request.form['confirmPassword']
                print(password,     newPWD,     confirmPWD)
                return redirect("/hr/profile/pwd")
              except:
                return redirect("/hr/signin")
          #VIEWING OPENING
```

```
@app.route("/hr/open
ings") def
hrOpenings(): try:
    token = request.cookies.get('token') email
    =                verify(token)               with
    sqlite3.connect('hr.db') as connection:
      cursor=connection.cursor()  cursor.execute("""
      SELECT
id,title,company_name,designation,salary_range,skills_required,roles_respo
nsibilities,company_description,location,website,author FROM OPENINGS
WHERE author=?""", (email,)) data=cursor.fetchall() data.reverse()
      connection.commit() return
      render_template("./openings/viewOpening.html",data=data)
  except Exception as e:
    # return redirect("/hr/signin")
    return render_template("./openings/viewOpening.html")
# CREATION NEW OPENING
@app.route("/hr/openings/new", methods=('GET', 'POST')) def
hrOpeningsCreate():
  if request.method == 'GET':
    try:
      token    =    request.cookies.get('token')    email    =
      verify(token)                                         return
      render_template("./openings/oneOpening.html")
    except:
      return redirect("/hr/signin")
  else:
    try:
      token                              =
      request.cookies.get('token')
      email  =  verify(token)  title  =
      request.form["title"]
      company_name = request.form["company_name"] designation =
      request.form["designation"]               salary_range               =
      request.form["salary_range"]             skills_required             =
      request.form["skills_required"]       roles_responsibilities       =
      request.form["roles_responsibilities"]   company_description   =
      request.form["company_description"]           location            =
            request.form["location"]    website    =    request.form["website"]
            author = email with sqlite3.connect('hr.db') as connection:
                key    =    uuid.uuid1().hex    cursor    =
                connection.cursor()
                cursor.execute("INSERT INTO OPENINGS
(id,title,company_name,designation,salary_range,skills_required,roles_respo
nsibilities,company_description,location,website,author) VALUES (?,?,?,?,?,?,?,?,?,?,?)", (key,
title, company_name, designation, salary_range, skills_required, roles_responsibilities,
company_description, location, website, author)) connection.commit() print("created
successfully") return redirect('/hr/openings')
        except Exception as e:
            print(e)
            return redirect('/hr/openings') #
DELETEING THE OPENINGS
@app.route("/hr/opening/<i
d>") def deleteOpening(id):
try:
        token = request.cookies.get('token') email
        =                 verify(token)                 with
        sqlite3.connect('hr.db') as connection:
            cursor=connection.cursor()    cursor.execute("""    SELECT    id    FROM
            OPENINGS WHERE id=?""",(id,)) data=cursor.fetchone() if not data:
                return redirect("/hr/openings")
            else:
                print(data[0]) cursor.execute(""" DELETE FROM OPENINGS
                WHERE id=?
""",(data[0],))
                connection.commit()  return
                redirect("/hr/openings")
    except    Exception
        as                e:
        connection.com
        mit()    print(e)
        return      "null"
        @app.route("/h
        r/openings/edi
        t/<id>",method
        s=('GET','POST'
        ))              def
        hrOpeningsOne
        (id):
    if request.method=="GET":
        try:
            token =
            request.cookies.get('token')
            email = verify(token)
            print(email) if not id:
                return render_template("./openings/oneOpening.html") with

            sqlite3.connect('hr.db') as connection:
                cursor=connection.cursor() cursor.execute("""SELECT id,author FROM
                OPENINGS WHERE id=?
""",(id,))
                data=cursor.fetchon
                e() if not data :
                    return redirect("/hr/openings")
                elif email== data[1]:
                    cursor.execute("""
                    SELECT id,
                    title,company_name,
                    designation,
                    salary_range,
                    skills_required,
                    roles_responsibilities,
                    company_description,
                    location,website, author
                    FROM OPENINGS WHERE id=?""",
                    (id,)) data=cursor.fetchone()
                    connection.commit() print(data)
                    return render_template("./openings/editing.html",data=data)
                else:
                    return redirect("/hr/openings")
            return render_template("./openings/oneOpening.html")
        except Exception as e:
            print(e)            return
            redirect("/hr/signin")
    else:
        token                              =
        request.cookies.get('token')
        email  =  verify(token)  title  =
        request.form["title"]
        company_name = request.form["company_name"] designation =
        request.form["designation"]              salary_range              =
        request.form["salary_range"]             skills_required            =
        request.form["skills_required"]       roles_responsibilities       =
        request.form["roles_responsibilities"]   company_description   =
        request.form["company_description"]            location             =
        request.form["location"]    website    =    request.form["website"]
        author = email with sqlite3.connect('hr.db') as connection:
            cursor=connection.cursor()
            cursor.execute("""SELECT id,author FROM OPENINGS WHERE id=?
""",(id,))
            data=cursor.fetchon
            e() if not data :
              return redirect("/hr/openings")
            elif            email==            data[1]:
              cursor.execute("""
              UPDATE OPENINGS SET
              title=?,
              company_name=?,designatio
              n=?, salary_range=?,
              skills_required=? ,
              roles_responsibilities=? ,
              company_description=?,
              location=?, website=?
              WHERE id=? """,(title,
              company_name,designa
              tion, salary_range,
              skills_required ,
              roles_responsibilities ,
              company_description,
              location, website, id
))
              data=cursor.fetchone()
              connection.commit()
```

```
                print(data) return
                redirect("/hr/openings")
            else:
                return redirect("/hr/openings")
  if__name_____== "_main__":
app.run(host="0.0.0.0", port=8081, debug=True)
```

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-39021-1660389280

PROJECT DEMO LINK:

https://github.com/IBM-EPBL/IBM-Project-39021-1660389280/blob/main/Final%20deliverables/demo-link-PNT2022TMID31521.mp4