



# **SKILL/JOB RECOMMENDER APPLICATION**



## **PROJECT REPORT**

*Submitted by*

**MUKESH A [19CS091]**

**SIVAKUMAR S K [19CS112]**

**GNANASHANKARAN N [19CS069]**

**KIRITHEESWARAN K [20CS515]**

*In partial fulfillment for the award of the degree*

*Of*

**BACHELOR OF ENGINEERING  
IN  
COMPUTER SCIENCE AND ENGINEERING**

**MUTHAYAMMAL ENGINEERING COLLEGE (AUTONOMOUS)**

**RASIPURAM - 637 408**

**ANNA UNIVERSITY::CHENNAI- 600 025**

**JUNE 2023**

<b>Date</b>	19 November 2022
<b>Team ID</b>	PNT2022TMID18783
<b>Project Name</b>	Project- Skill and Job Recommender
<b>Team Leader/ Team Members</b>	Mukesh A Sivakumar S K Gnanashankaran N Kiritheeswaran K

## ABSTRACT

In proposed work recommender systems have gained rural graduates popularity in recent years because they inefficiently alleviate information overload by delivering individualised job suggestions. Although they are from rural place so this platform provides several ways and practices for using job recommender systems in the literature, the majority of them fall short of recommending positions that are correctly matched to the profiles of rural graduate job searchers is the platform its very useful to them and through that they can search the job very easily according to their skills. In the Skill/Job Recommendation Application its has a various number of job opportunities for a various skills and the process of accessing the application is very useful In the last years, job recommender systems have become popular since they successfully reduce information overload by generating personalized job suggestions. Although in the literature exists a variety of techniques and strategies used as part of job recommender systems, most of them fail to recommending job vacancies that fit properly to the job seekers profiles. Thus, the contributions of this work are threefold, we: i) made publicly available a new dataset formed by a set of job seekers profiles and a set of job vacancies collected from different job search engine sites; ii) put forward the proposal of a framework for job recommendation based on professional skills of job seekers; and iii) carried out an evaluation to quantify empirically the recommendation abilities of two state-of-the-art methods, considering different configurations, within the proposed framework. We thus present a general panorama of job recommendation task aiming to facilitate research and real-world application design regarding this important issue.

<b>CHAPTER NO</b>	<b>TABLE OF CONTENTS</b>	<b>PAGE NO</b>
	<b>INDEX</b>	
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 Project Overview	8
	1.2 Purpose	10
<b>2</b>	<b>LITERATURE SURVEY</b>	12
	2.1 Existing problem	19
	2.2 References	22
	2.3 Problem Statement Definition	24
<b>3</b>	<b>IDEATION AND PROPOSED SOLUTION</b>	
	3.1 Empathy Map Canvas	26
	3.2 Ideation and Brainstorming	27
	3.3 Proposed Solution	27
	3.4 Problem Solution fit	32
<b>4</b>	<b>REQUIREMENT ANALYSIS</b>	
	4.1 Functional Requirement	34
	4.2 Non-Functional Requirements	42
<b>5</b>	<b>PROJECT DESIGN</b>	
	5.1 Data Flow Diagrams	43
	5.2 Solution and Technical Architecture	43
	5.3 User Stories	48
<b>6</b>	<b>PROJECT PLANNING AND SCHEDULING</b>	
	6.1 Sprint Planning and Estimation	51
	6.2 Sprint Delivery Schedule	59
	6.3 Reports from JIRA	60
<b>7</b>	<b>CODING AND SOLUTIONING</b>	
	7.1 Feature 1	61
	7.2 Feature 2	74
	7.3 Database Schema(if Applicable)	
<b>8</b>	<b>TESTING</b>	
	8.1 Test Cases	133
	8.2 User Acceptance Testing	134

<b>9</b>	<b>RESULTS</b>	
	9.1 Performance Metrics	141
<b>10</b>	<b>ADVANTAGES &amp; DISADVANTAGES</b>	142
<b>11</b>	<b>CONCLUSION</b>	143
<b>12</b>	<b>FUTURE SCOPE</b>	143
<b>13</b>	<b>APPINDIX</b>	
	Source Code	074
	GitHub and Project Demo Link	144

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 CLOUD COMPUTING**

Cloud computing is a recently developing paradigm of distributed computing. Though it is not a new idea that emerged just recently. In 1969 [16] L. Kleinrock anticipated, “As of now, computer networks are still in their infancy. But as they grow up and become more sophisticated, we will probably see the spread of ‘computer utilities’ which, like present electric and telephone utilities, will service individual homes and offices across the country.” His vision was the true indication of today’s utility based computing paradigm. One of the giant steps towards this world was taken in mid 1990s when grid computing was first coined to allow consumers to obtain computing power on demand. The origin of cloud computing can be seen as an evolution of grid computing technologies. The term Cloud computing was given prominence first by Google’s CEO Eric Schmidt in late 2006 (may be he coined the term) [6]. So the birth of cloud computing is very recent phenomena although its root belongs to some old ideas with new business, technical and social perspectives.

Cloud is essentially provided by large distributed data centers. These data centers are often organized as grid and the cloud is built on top of the grid services. Cloud users are provided with virtual images of the physical machines in the data centers. This virtualization is one of the key concept of cloud computing as it essentially builds the abstraction over the physical system. Many cloud applications are gaining popularity day by day for their availability, reliability, scalability and utility model. These applications made distributed computing easy as the critical aspects are handled by the cloud provider itself.

### 1.1.1 CLOUD COMPUTING BASICS

Cloud computing is a paradigm of distributed computing to provide the customers on-demand, utility based computing services. Cloud users can provide more reliable, available and updated services to their clients in turn. Cloud itself consists of physical machines in the data centers of cloud providers. Virtualization is provided on top of these physical machines. These virtual machines are provided to the cloud users. Different cloud provider provides cloud services of different abstraction level. E.g. Amazon EC2 enables the users to handle very low level details where Google App-Engine provides a development platform for the developers to develop their applications. So the cloud services are divided into many types like Software as a Service, Platform as a Service or Infrastructure as a Service. These services are available over the Internet in the whole world where the cloud acts as the single point of access for serving all customers. Cloud computing architecture addresses difficulties of large scale data processing.

### 1.1.2 TYPES OF CLOUD

Cloud can be of three types

**1. Private Cloud** – This type of cloud is maintained within an organization and used solely for their internal purpose. So the utility model is not a big term in this scenario. Many companies are moving towards this setting and experts consider this is the 1st step for an organization to move into cloud. Security, network bandwidth are not critical issues for private cloud.

**2. Public Cloud** – In this type an organization rents cloud services from cloud providers on-demand basis. Services provided to the users using utility computing model.

**3. Hybrid Cloud** – This type of cloud is composed of multiple internal or external cloud. This is the scenario when an organization moves to public cloud computing domain from its internal private cloud.

## **1.2 MOTIVATION OF JOB RECOMMENDER SYSTEMS**

The significance of Information System (IS) support in the recruitment process can be observed when considering the phases of the recruitment such as the handling of candidates' applications and the pre-selection of candidates. However, a best fit between job and candidates depends on underlying aspects that are hard to measure. These underlying aspects are a significant reason why information systems have not been extensively used in the area of personnel selection so far. Mostly, IS technology is used to pre-select applicants based on Boolean search method. This method used queries contain a combination of key words that define skill requirements in order to determine those candidates that match with search criteria. Such type of skill matching is applied in numerous e-recruiting applications.

### **1.2.1 THE RECRUITING PROCESS**

Recruiting process is a core function of human resource management treating the labor as one of the important factors of production (Färber et al., 2003). The key Al-Otaibi and Ykhlef 5 ,1. Recruiting process. objective of the recruiting process is to hire candidates who are valuable for the company (Laumer and Eckhardt, 2009). Two viewpoints are distinguished: from recruiters' and job seekers. The recruiters generate the job description by determining the set of requirements and constraints on skills, expertise levels, and degrees. The job-seeker, on the other hand, generates his/her CV by specifying the academic background, previous work experience and skills (Fazel-Zarandi and Fox, 2010). The IT support for the recruiting activities is ranging from attracting and finding talent to choose and retain candidates (Laumer et al., 2010



### **1.2.2 WHAT IS RECOMMENDER SYSTEM?**

These are the systems that help us to select out similar things whenever we select something online. The concept of understanding a user's preference by their online behaviour, previous purchases, or history in the system is called a recommender system. The need for a recommender system has grown from time to time. At First, Entertainment industries exploited the benefits of these systems. Then recommender systems were implemented in e-shopping businesses, online news, but very few companies have tried implementing it in the hiring process.

### **1.2.3 WHY DO WE USE RECOMMENDER SYSTEM ?**

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 (McAlone, 2016).

### **1.2.4 HOW TO IMPLEMENT RECSYS IN THE HIRING PROCESS?**

As we know the cycle of the hiring process, we have the opportunity to implement a recommender system to it. We can implement a recommender system from the enterprise perspective and the job seeker perspective. From the perspective of job seeker, we can implement a recommender system that could collect the user preference that is user skills, Location. We are concentrating on job-related to the Information and technology domain. These IT domain jobs require skills such as

programming languages, database skills, Framework skills, and user preference on different platforms. The author's motivation to implement a recommender system for a job seeker came from personal strive when I was looking for the job.

### **1.2.5 JOB RECOMMENDATION TECHNIQUES**

In recent years, several recommender system techniques applied in candidates/job matching problem, started by the personnel selection approach that proposed by Färber et al. (2003) who developed a probabilistic hybrid recommendation approach for candidates/job matching. Then, their model utilized and extended by Malinowski et al. (2006), Keim (2007) and Malinowski et al. (2008).

- **Hybrid Job Recommender Systems**

A probabilistic hybrid approach Färber et al. (2003) applied a recommendation system initially used to recommend objects to users such as movies or books to matching partners. The recommendation approach used both concepts: content-based filtering and collaborative filtering simultaneously. This assists partially to overcome the problem of data sparsity. Another concept that they applied is the latent aspect model described by (Hofmann and Puzicha, 1999). It understands the individual preferences as a convex combination of preference factors.

- **A Proactive Job Recommender System**

The proactive recommender system is an adaptive system that attempted to integrate the idea of recommender systems (Schafer et al., 1999) and adaptive hypermedia (Brusilovsky, 2001). This system contains five components: web spider, ontology checker, profile analyzer, preference analyzer, and user interface 5138 Int. J. Phys. Sci. generator. Web spider is a parser that periodically acquires job information from an exterior source. The ontology checker matches information with ontologies and performs the classification.

## **1.2.6 CONTENT-BASED JOB RECOMMENDER SYSTEMS**

### **Machine Learned Recommender System**

The recommendation problem treated as a supervised machine learning problem. They build an automated system that can recommend jobs to applicants based on their past job histories, in order to facilitate the process of choosing a new job. An item in this learning model represents a person who is hired in an organization. Each item is characterized by set of features extracted from the candidates' resumes. Given a person who is currently working in an organization, they want to predict the next organization. If the accuracy of such predictions is sufficiently high, the model can be used to recommend organizations to employees who are seeking for jobs.

## **1.3 OBJECTIVE**

The aim of recommender systems is to assist users in finding their way through huge databases and catalogues, by filtering and suggesting relevant items taking into account or inferring the users' preferences (i.e., tastes, interests, or priorities). These systems use information filtering techniques to process information and provide the user with potentially more relevant items.

## **1.4 SCOPE**

- In this functionality, jobs will be recommended on the basis of explicit searching by the user.
- They shorten the distance between the customer's need and satisfaction.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 LITERATURE SURVEY**

##### **2.1.1 JOB RECOMMENDER SYSTEMS**

**AUTHORS :** Corne de Ruijt, Sandjai Bhulai

**ALGORITHM :** Model-Based Methods On Shallow Embeddings (MM-SE).

From the start of the commercialization of the internet in the late 1980s, the question was raised of how this technology could be leveraged in employee recruitment to enhance job seeker - vacancy matching. Even before the start of the world wide web, Vega [116] already proposed a system to match job seekers and jobs, which could “be consulted by Minitel, using telephone number 3615 and selecting the LM/EMPLOI service”. I.e., the service allowed job seekers to send text messages in the form of search queries or their digital resume, over the telephone line, using a computer terminal called Minitel1 . The service would compare words in the query/resume to a knowledge base, which used a fixed job taxonomy to return a set of potentially interesting vacancies for the job seeker.

#### **LIMITATIONS**

- Although we believe to have given a broad overview of contributions on job recommender systems.
- Although we managed to further split JRS hybrids into smaller categories, still some classes comprise similar methods.

##### **2.1.2 JOB RECOMMENDATION BASED ON JOB SEEKER SKILLS**

**AUTHORS:** Jorge Valverde-Rebaza, Ricardo Puma, Paul Bustios, Nathalia C. Silva

## **ALGORITHM :** Term Frequency-Inverse Document Frequency (TF-IDF)

Nowadays, job search is a task commonly done on the Internet using job search engine sites like LinkedIn<sup>1</sup> , Indeed<sup>2</sup> , and others. Commonly, a job seeker has two ways to search a job using these sites. Doing a query based on keywords related to the job vacancy that he/she is looking for, or creating and/or updating a professional profile containing data related to his/her education, professional experience, professional skills and other, and receive personalized job recommendations based on this data. Sites providing support to the former case are more popular and have a simpler structure; however, their recommendations are less accurate than those of the sites using profile data.

## **LIMITATIONS**

- A variety of techniques and strategies used as part of job recommender systems, most of them fail to recommending job vacancies .
- Unpredictable Items
- Lack of Data, Perhaps the biggest issue facing recommender systems is that they need a lot of data to effectively make recommendations

## **2.1.3 JOB RECOMMENDATION SYSTEM USING PROFILE MATCHING AND WEB-CRAWLING**

**AUTHORS :** Deepali V Musale, Mamta K Nagpure, Kaumudini S Patil, Rukhsar F

## **ALGORITHM:** SMS-Based Recommendation Process

The recommender systems are being used in every possible system for example, clothes recommendation, book recommendation etc. However the type of recommendations provided may be different according to the domain of its use. In

the case of job recommendation system the case is little bit different. Here, it will be favorable to provide mostly personalized and profile based job recommendations. In job recommendation systems, there are varieties of students, having different education level and skills. Based on student's respective background details, each one of them expects to get only those job recommendations which are highly relevant for that particular student. Also, two students having similar profiles may have different job tastes. Here, job taste can be defined as the preference criterion considered before applying for a particular job. For one, preference can be of getting a job in higher company, as opposed to the other who may be interested in having a job which offers higher payment.

## **LIMITATIONS**

- It relies only on HTML scraping to crawl the job listings, which does not always work in modern web applications due to client-side rendering of reactjs, etc.
- Cold Start, Scalability
- Low behavior

### **2.1.4 RECOMMENDATION SYSTEMS FOR RECRUITMENT WITHIN AN EDUCATIONAL CONTEXT**

**AUTHORS:** Gustaf Lagerqvist, Anton Stalhandske

**ALGORITHM:** Design Science Research Methodology

Due to the digitalization of the recruitment process, the outreach is now far wider than it was before the step into the digital format. This has, in turn, generated an enormous increase in the number of applicants that apply for each open position which has made the process of screening each of these candidates a more or less impossible

task. As an example, Google received roughly two million applications for about 14,500 jobs in 2017 . With the rapid evolution of digital recruitment and its corresponding tools, the need for precision when it comes to matchmaking is evident . This thesis work sprung from a conceptual idea of an online educational and networking platform. The purpose of the platform is to function as a forum for people to learn and improve abilities instead of, and, as a complement to traditional schooling. One way to achieve this is by matching students that want to practice a certain skill, with live projects run by companies.

## **LIMITATIONS**

- Significant investments required
- The complex onboarding process
- Too many choices

### **2.1.5 EFFICIENT AND SCALABLE JOB RECOMMENDER SYSTEM USING COLLABORATIVE FILTERING**

**AUTHORS:** Ravita Mishra (&) Sheetal Rathi

**ALGORITHM:** Collaborative Filtering

Now-a-day social media is very common platform to share the data and day today's activities. With the enormous use of various internet sources likes, mobile phone and smart devices, Internet users can receive huge information about shopping and social activity of user and online learning. If the data volume and variety increases tremendously, then individual user faces various problem of excessive information, it causes problem to make the correct decisions. This framework is called as information overload. To resolve users' information overload problem, a new technique recommender system comes in pictures. Recommender system can solve various problems by effectively finding users' probable requirements and elect fascinating

items from a vast amount of applicant information. Recommender systems are mainly categorized into three main forms, i.e., content-based (CB), collaborative filtering (CF) and hybrid recommender system is combination of both resolve the drawback of content and collaborative filtering.

## **LIMITATIONS**

- Data Sparsity and cold-start problem. Data sparsity is seen as a key disadvantage of collaborative filtering
- Cold-star
- Scalability

## **2.1.6 ENHANCED JOB RECOMMENDATION SYSTEM**

**AUTHORS:** Shivraj Hulbatte, Amit Wabale, Suraj Patil, Nikhilkumar Sathe

**ALGORITHM:** Expectation Maximization (EM)

The increase in usage of Internet has heightened the need for online job hunting. According to Job site's report 2014, 68% of online job seekers are college graduates or post graduates. The key problem is that most of the job-hunting websites just display the recruitment information to website viewers. Students have to go through all the information to find the jobs they want to apply. The whole procedure is tedious and inefficient. We need an easy job recommendation system where everyone will have a fair and square chance. This saves a lot of potential time and money both, on the industrial as well as the job seeker's side. Moreover, as the candidate gets a fair chance to prove his talent in the real world it is a lot more efficient system. The basic agenda of every algorithm used in today's world, be it a traditional algorithm or a hybrid algorithm, is to provide a suitable job that the user actually seeks and wishes for. Recently, job recommendation has attracted a lot of research attention and has played an important role on the online recruiting website.



## **LIMITATIONS**

- Significant investments required.
- The complex onboarding process.
- Inability to capture changes in user behavior.

### **2.1.7 JOB RECOMMENDATION SYSTEM VIA SOCIAL MEDIA USING MACHINE LEARNING**

**AUTHORS:** Ankita Bhosale, Pratiksha More, Aradhana Sutar, Rajababu Swain

**ALGORITHM:** Collaborative Filtering

Recommender systems have been widely adopted by many online websites to help users overcome the information overload problem and make their purchase decisions. Popular recommendation techniques, such as collaborative filtering (CF) and content-based filtering, can be utilized in social media for users to search jobs. CF makes recommendations based on the assumption that users with similar ratings and interests for some items are likely to have similar preferences for other items. Content-based filtering assumes that the users' interests are able to be represented by the content of posts they have shown interest in, and those contents that have content descriptions similar to the target user's favorite jobs are recommended. In this paper, we capitalize on user posts to understand the requirements from the users' perspective and leverage user reviews to develop a job recommender system. The topics hidden in the review texts can be a type of representation of the job features. The topic distributions of apps and user preferences are both considered when producing recommendation scores of the relevant jobs for the user to make personalized .

## **LIMITATIONS**

- The stock market prices of the company vary in a daily fashion.
- The social media pattern usage of the company can be determined to find the sentiment score values.

- The demand of recommendation has aroused severely since there are huge number of choices available

### **2.1.8 PROPOSED RECOMMENDED SYSTEM FOR EMPLOYABILITY SKILLING USING NEURAL NETWORK**

**AUTHORS:** Dr. Kavita Suryawanshi, Mrs, Arati Patil and Ms. Sarika Choudhari

**ALGORITHM:** Proposed Recommendation System

It has been observed that, when the students are enrolled in the professional course, they are more concern about employability after course completion. Even after completion of course, they are lacking in different industry prospects. When student is admitted in the first year of program, if they know the industry requirements in advance and the skillset which they need to improve, then probably the unemployability rate gets reduced. In view of increasing employability rate of graduate students, we are proposing a Recommendation System for Employability Skilling which will help students to understand the grey area of skills which needs to be enhanced to become employable. This Recommendation System, will be using supervised artificial neural network to train the student data. Input to the neural network will be previous educational history, CET score conducted by the Competent Authority, Proficiency test conducted by admitted institute, Student's expectation/interest regarding employability, Communication skills, Behavior analysis data, Leadership skills, team player skills etc.

### **LIMITATIONS**

- Neural networks usually require much more data than traditional machine learning algorithms
- It can be exhausting, time-consuming, and often ineffective.

## **2.2 EXISTING SYSTEM**

The Proposed Recommended System for Employability Skilling using Neural Network is important for students to understand the enhancement skillset needed for them to be employable at the end of the completion of program as per their desired career profile. The Recommendation System will guide the Guardian Faculty Member to mentor the students according to the improvement areas suggest. This paper comprises of VI sections. Section Section-II shows the literature review of the proposed recommendation system. Section Discussion of the proposed recommendation system while Section recommendation system.

### **2.2.1 ALGORITHM**

#### **Step 1:**

The proposed recommended system will have rigorous literature review to understand various skills required for employability.

#### **Step 2:**

Requirement/data gathering for the proposed recommendation system will be collected from different users of the system viz. Student, Proficiency Coordinator, Institute's Student section and Guardian Faculty member. For example, X score, XII score, Graduation score, CET Score, Proficiency Test Score, desired employability of students and Ratings regarding different skillsets. Also data will be collected from various employers regarding their required skillsets from employees. For example, Multi-national company requirements, Government job requirements, Entrepreneurship requirements as well as requirements regarding higher education etc. For data collection, Group discussion, Interviews as well as questionnaires techniques will be used.

#### **Step 3:**

System's analysis and data interpretation will be carried out. Supervised Artificial Neural Network will be used here. Supervised learning takes place under

the supervision of a teacher. This learning process is dependent. During the training of ANN under supervised learning, Input vector is presented to the network, which will produce an output vector. In this Recommendation System, X (Input vector) will be [10th Marks, 12th Marks, Graduation Marks, CET Score, Proficiency Test Result, Student Expectation, Communication Skills, Behaviour Skills, Leadership Skills, and Team Player Skills]. And output vector is compared with the desired/target output vector. An error signal is generated if there is a difference between the actual output and the desired/target output vector.

**Table 1:** Training Data Set Sample

Traning Data Set											
Users	Communication Skills(10)	Behavior Skills(10)	leadership Skills(10)	Team Player Skills(10)	SSC Score(100)	HSC Score(100)	Graduation (100)	CET Score(100)	Proficiency Test(100)	Expectation Regarding Job	SkillSets for improvement
	Input_0	Input_1	Input_2	Input_3	Input_4	Input_5	Input_6	Input_7	Input_8	Input_9	Output_2
User_1	6	5	8	6	65	60	60	34	45	5	[1,2,5]
User_2	7	7	7	8	71	72	79	67	56	4	[1,2,3]
User_3	8	8	9	8	54	55	59	50	23	2	[1,2,5]
User_4	5	8	9	7	70	70	70	76	67	5,2	[1,2,4]
User_5	9	9	9	9	52	53	50	45	23	3	[5]

**Table 2:** Skillsets Categorization

SkillSets	
Skills	Categorization
Communication Skills	1
Behavior Skills	2
leadership Skill	3
Team Player Skills	4
Technical Skills	5

**Table 3: Expected Job Categorization**

Expected Job					
Job	Categorization	SkillSets	Job	Categorization	SkillSets
MNC Company	5	Communication Skills-above 08	Entrepreneur	2	Communication Skills-above 09
		Behavior Skills-above 08			Behavior Skills-above 09
		Leadership Skills-above 08			Leadership Skills-above 09
		Team Player Skills-above 08			Team Player Skills-above 09
		SSC Score-above-60%			SSC Score-above 50%
		HSC Scoreabove- 60%			HSC Scoreabove50%
		Graduation above -60%			Graduation above50%
		CET Scoreabove 40%			CET Scoreabove40%
Government Job	4	Proficiency Testabove- 60%	No Criteria Company	1	Proficiency Test above 70%
		Communication Skills-above 07			Communication Skills-above 08
		Behavior Skills-above 08			Behavior Skills-above 08
		Leadership Skills-above 08			Leadership Skills-above 08
		Team Player Skills-above 08			Team Player Skills-above 08
		SSC Score-above-60%			SSC Score-above 50%
		HSC Scoreabove- 60%			HSC Score- above 50%
		Graduation above -60%			Graduation -above50%
Higher Education	3	CET Scoreabove 40%			CET Score-above50%
		Proficiency Testabove- 60%			Proficiency Testabove 70%
		Communication Skills-above 08			
		Behavior Skills-above 08			
		Leadership Skills-above 08			
		Team Player Skills-above 08			
		SSC Score-above 50%			
		HSC Scoreabove 50%			
		Graduation above 50%			
		CET Scoreabove 40%			
		Proficiency Testabove 70%			

#### V. EXPECTED OUTCOME OF PROPOSED RECOMMENDED SYSTEM

##### 2.2.1.1 System Recommended Suitable Employability Profile

After training the input vector of individual students, our system will recommended suitable employability profile as per his/her existing skills set. So it will help student to understand and improve/enhance the skillset to achieve his/her own expectations about employability.

##### 2.2.1.2 Skillset for Improvement to Achieve Student's Desired Profile

This recommendation system will produce individual student improvement/enhancement skillset. Which will help student to understand lacking Technical skill, Soft Skills etc. for improvement/enhancement. By improving these recommended skillsets they can achieve their expectation about employability.

##### 2.2.1.3 Mentoring to Increase Employability Rate

As this system will help individual student to achieve his/her expectation about employability, it will also help to the institute/organization for mentoring students

based on output received from this recommendation system to enhance the performance of the individual student under the guidance of Guardian Faculty Member. Getting desired career job and being employable for the same is the vision of every student who opts for the post-graduation course. To face the global competition, every student need to enhance their required skillsets.

### **2.2.2 LIMITATIONS**

- User cold-start problem. The system cannot recommend products to new users who have not had any interaction yet
- Item cold-start problem. The system cannot also recommend an item that users never selected before
- Significant investments required
- Inability to capture changes in user behavior
- The complex onboarding process

### **2.3 REFERENCES**

[1].<https://www.financialexpress.com/jobs/employability-in-india-where-the-needle-needs-to-move-in2020/1772595/>

[2].<https://www.statista.com/statistics/812106/youth-unemployment-rate-india/>

[3]. "Employability Of Management Students In India: Some Concerns And Considerations", Sujoy Kumar Dhar, AIMA Journal of Management & Research, November 2012, Volume 6, Issue 4/4, ISSN 0974-497 Copy right © 2012 AJMR-AIMA

[4]. Enhancing employability in engineering and management students through soft skills, January 2014, Industrial and Commercial Training 46(1), DOI: 10.1108/ICT-04-2013-0023, Authors: M.S. Rao

[5]. Open Engineering Volume 10: Issue 1, The Employability Skills Needed To Face the Demands of Work in the Future: Systematic Literature Reviews, Nuryake Fajaryati, Budiyo, Muhammad Akhyar and Wiranto, DOI: <https://doi.org/10.1515/eng-2020-0072> Published online: 02 Jul 2020

[6]. P. Vanitha, Dr. A. T. Jaganathan "A Study on Enhancing Employability Skills of Graduates in India" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-2 | Issue-2, February 2018,

URL: <https://www.ijtsrd.com/papers/ijtsrd9561.pdf> Paper

URL: <http://www.ijtsrd.com/management/management-development/9561/a-study-on-enhancing-employabilityskills-of-graduates-in-india/p-vanitha>

[7]. Prof. Dr. Vinitaa Agrawal<sup>1</sup>, Santanu Dasgupta<sup>2</sup>, "Identifying the Key Employability Skills: Evidence from Literature Review", IOSR Journal of Business and Management (IOSR-JBM) e-ISSN: 2278-487X, p-ISSN: 2319-7668 PP 85-90 [www.iosrjournals.org](http://www.iosrjournals.org)

[8]. Fatwa Tentama, Muhamad Hasan Abdillah, Master of Psychology, Universitas Ahmad Dahlan, Indonesia "Student employability examined from academic achievement and self-concept", International Journal of Evaluation and Research in Education (IJERE) Vol. 8, No. 2, June 2019, pp. 243~248 ISSN: 2252-8822, DOI: 10.11591/ijere.v8i2.18128

[9]. Yasmeen Bano, Dr. S. Vasantha "Review On Employability Skill Gap", International Journal of Research in Social Sciences Vol. 9 Issue 2, February 2019, ISSN: 2249-2496 Impact Factor: 7.081 Journal Homepage: <http://www.ijmra.us>

[10]. Building employability skills into the higher education curriculum: A university-wide initiative, March 2000, Education and Training 42(2):75-83, DOI: 10.1108/00400910010331620, Stephen fallows, Christine Steven

[11]. Marta Abelha, Sandra Fernandes, Diana Mesquita, Filipa Seabra and Ana Teresa Ferreira-Oliveira,” Graduate Employability and Competence Development in Higher Education—A Systematic Literature Review Using PRISMA”, Sustainability 2020, 12, 5900; doi:10.3390/su12155900.

[12]. Roshan G. Belsare, Dr. V. M. Deshmukh, ”Employment Recommendation System using Matching, Collaborative Filtering and Content Based Recommendation”, International Journal of Computer Applications Technology and Research Volume 7–Issue 6, 215-220, 2018, ISSN:-2319-8656

[13]. Jose Aguilar, Guido Riofrio,” A general framework for intelligent recommender systems”, Applied Computing and Informatics, Volume 13, Issue 2, July 2017, Pages 147-160, Science Direct

[14]. Ioannis K. Paparrizos, Berkant Barla Cambazoglu, Aristides Gionis “Machine Learned Job Recommendation, Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011.

[15]. Nedhal Al-Saiyd, “Prediction of It Jobs Using Neural Network Technique”, Ubiquitous Computing and Communication Journal (ISSN 1992-8424)

## **2.4 PROPOSED STATEMENT DEFINITION**

### **2.4.1 Customer Problem Statement Template:**

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the



process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

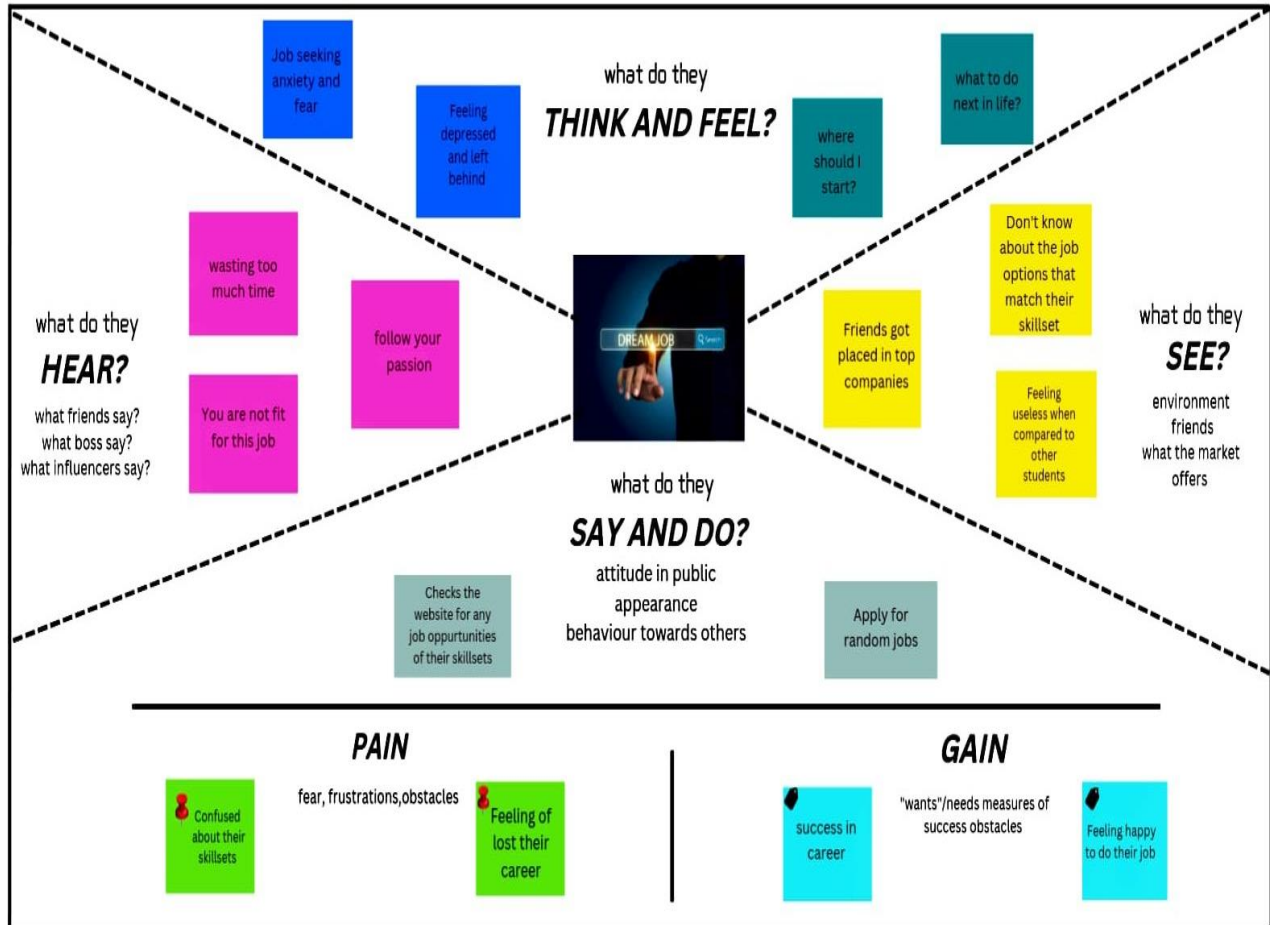
I am	<u>Graduate ,Job Seeker</u>	As a job <u>seeker</u> , I am looking for a suitable job which satisfies my self
I am trying to	Searching for job	I am searching for a job which I admired
But	Premium features	Features like instant vacancy notification and user priority gives more advantage
Because	Paid subscription	These premium features are mostly paid subscription
With makes me feel	Inequality	This advantage for paid user seems unfair so job offers are mostly taken away by the premium users which makes normal users to feel inequality among them

<b>Problem Statement (PS)</b>	<b>I am</b>	<b>I'm trying to</b>	<b>But</b>	<b>Because</b>	<b>Which makes me feel</b>
PS-1	Graduate job seeker	Searching for job	Premium features	Paid subscription	Inequality
PS-2	As a job seeker I am looking for a suitable job which satisfies myself	I am searching for a job which I admitted	Features like instant vacancy notification and user priority gives more advantages	These premium features are mostly paid subscription	This advantage for paid user seems unfair so job offers are mostly taken away by the premium users which makes normal users to feel inequality among them

## CHAPTER 3

### IDEATION & PROPOSED SOLUTION

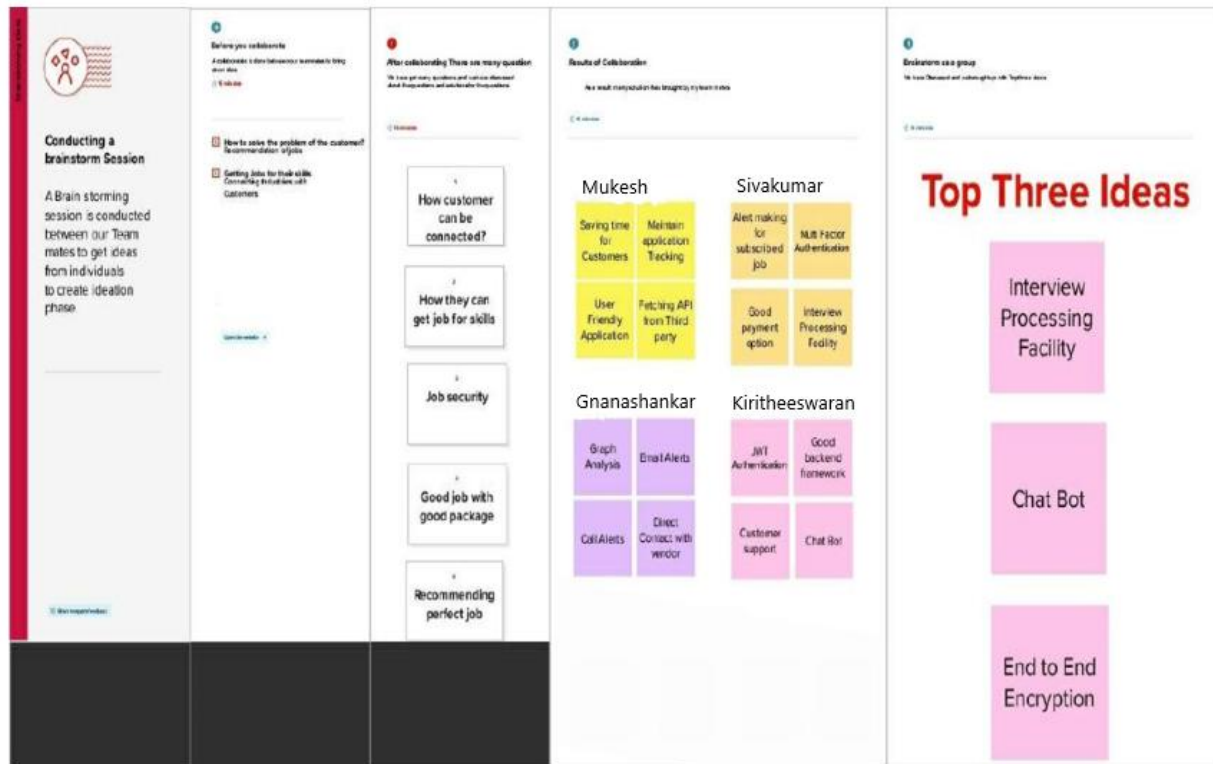
#### 3.1 EMPATHY MAP CANVAS



##### 3.1.1 Empathy Map :

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user personal, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community. How does an empathy map look like? Essentially, an empathy map is a square divided into four quadrants with the user or client in the middle.

## 3.2 IDEATION & BRAINSTORMING



## 3.3 PROPOSED SOLUTION

### 3.3.1 PARAMETERS

#### PROBLEM IN JOB RECOMMENDATION

In Society many individuals face problems in job searching , many job seekers are unable to find their dream job. Many good technical persons are unable to land their dream job, and lose their hope. So we have come up with a solution.

#### SOLUTION FOR THE PROBLEM

Our teammates, have designed a multi speciality software which helps job seekers to land in their dream job according to their skills

## **NOVELTY**

This software has designed to get Recommended job for the job seekers, the unique thing in this software is it has two types of account, one is vendor type account, another one is customer type, so The job posted by the vendors can easily meet the customers

## **FEASIBILITY**

The project is feasible and can be implemented using flask framework, and the job api can be brought from third party service, and the software can be accessed from all over the world to meet job at all ends

## **BUSINESS MODEL**

Apart from job recommendation, a revenue is important for a organization, so the required revenue can be brought up by third party ads like google ads

## **SOCIAL IMPACT**

This software solves the social impacts like making all job seekers or individuals to meet the job that meets their criteria, so this can solve social issue on job finding

## **SCALABILITY**

This software is based on SDLC, so the scalability of the software can be changed according to the needs of customers in future

### **3.3.2 ALGORITHM**

#### **3.3.2.1 COLLABORATIVE FILTERING**

Collaborative filtering (CF) and its modifications is one of the most commonly used recommendation algorithms. Even data scientist beginners can use it to build their personal movie recommender system, for example, for [a resume project](#).

When we want to recommend something to a user, the most logical thing to do is to find people with similar interests, analyze their behavior, and recommend our user the same items. Or we can look at the items similar to ones which the user bought earlier, and recommend products which are like them.

These are two basic approaches in CF: user-based collaborative filtering and item-based collaborative filtering, respectively.

In both cases this recommendation engine has two steps:

1. Find out how many users/items in the database are similar to the given user/item.
2. Assess other users/items to predict what grade you would give the user of this product, given the total weight of the users/items that are more similar to this one.

### 3.3.2.2 WHAT DOES “MOST SIMILAR” MEAN IN THIS ALGORITHM?

User / Item	Batman	Star Wars	Titanic
Bill	3	3	
Jane		2	4
Tom		5	

All we have is a vector of preferences for each user (row of the matrix R) and the vector of user ratings for each product (columns of the matrix R).

First of all, let's leave only the elements for which we know the values in both vectors. For example, if we want to compare Bill and Jane, we can mention that Bill hasn't watched Titanic and Jane hasn't watched Batman until this moment, so we can measure their similarity only by Star Wars. How could anyone not watch Star Wars, right? :)

The most popular techniques to measure similarity are cosine similarity or correlations between vectors of users/items. The final step is to take the weighted arithmetic mean according to the degree of similarity to fill empty cells in the table.

$$x_{ij} \approx \langle u_i, v_j \rangle$$

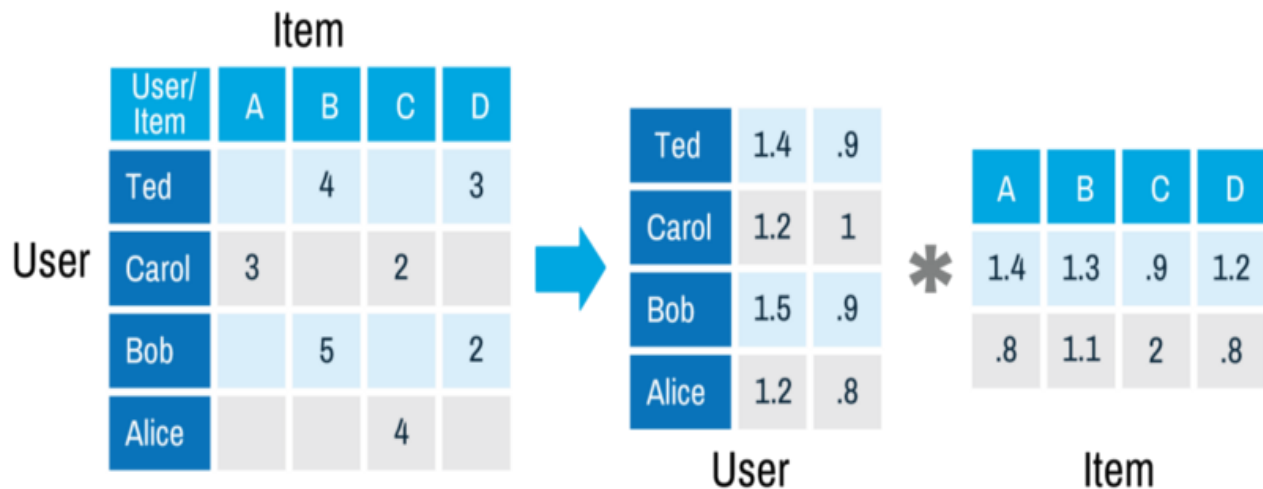
$$\sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min$$

### 3.3.2.3 MATRIX DECOMPOSITION FOR RECOMMENDATIONS

The next interesting approach uses matrix decompositions. It's a very elegant recommendation algorithm because usually, when it comes to matrix decomposition, we don't give much thought to what items are going to stay in the columns and rows of the resulting matrices. But using this recommender engine, we see clearly that  $u$  is a vector of interests of  $i$ -th user, and  $v$  is a vector of parameters for  $j$ -th film.

So we can approximate  $x$  (grade from  $i$ -th user to  $j$ -th film) with dot product of  $u$  and  $v$ . We build these vectors by the known scores and use them to predict unknown grades.

For example, after matrix decomposition we have vector  $(1.4; .9)$  for Ted and vector  $(1.4; .8)$  for film A, now we can restore the grade for film A–Ted just by calculating the dot product of  $(1.4; .9)$  and  $(1.4; .8)$ . As a result, we get 2.68 grade.



### 3.3.2.4 CLUSTERING

The previous recommendation algorithms are rather simple and are appropriate for small systems. Until this moment, we considered a recommendation problem as a supervised machine learning task. It's time to apply unsupervised methods to solve the problem.

Imagine, we're building a big recommendation system where collaborative filtering and matrix decompositions should work longer. The first idea would be clustering.

At the start of a business, there is a lack of previous users' grades, and clustering would be the best approach.

But separately, clustering is a bit weak, because what we do in fact is we identify user groups and recommend each user in this group the same items. When

we have enough data it's better to use clustering as the first step for shrinking the selection of relevant neighbors in collaborative filtering algorithms. It can also improve the performance of complex recommendation systems.

Each cluster would be assigned to typical preferences, based on preferences of customers who belong to the cluster. Customers within each cluster would receive recommendations computed at the cluster level.

## **LIMITATIONS**

- Lack of User Activity. A recommendation system majorly depends on the activities performed by multiple users
- Lack of Data. The availability of abundant data is what a recommendation system needs
- New Item Introduction
- Scalability
- Changing Trends

## **3.4 PROBLEM SOLUTION FIT**

### **3.4.1 PROBLEMS AND SOLUTIONS**

#### **HOW CUSTOMERS MEET JOB?**

The software uses two types of account, one is vendor type another is customer type, so the job posted by vendors can be easily accessed by customers

#### **HOW CUSTOMERS GET SUGGESTIONS?**

As the profile created for customers, all the experience and skill sets are gathered, so a special type of algorithm will provide suggestion about job that will match their profile



## **HOW CUSTOMERS CLARIFY THEIR PROBLEMS?**

The software uses customer support facility and chatter bot, so any questions are clarified both vendor and customer side

## **WHY JOB RECOMMENDATION APPLICATION?**

Many individuals in society are without job due to many reasons so, we come up with online application it is easy to use and all individuals can apply for the job that fit for their skill

## **TIME AND MONEY?**

As it is a online platform, the time and money can be saved, comparing to offline platform

## CHAPTER 4

### REQUIREMENT ANALYSIS

#### 4.1 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.

## **4.1.1 SYSTEM SPECIFICATIONS**

### **4.1.1.1 HARDWARE REQUIREMENTS**

- **SYSTEM** : Intel Core i3-3217U 1.8GHz
- **HARD DISK** : 512GB HDD/SSD
- **MONITOR** : HD LCD
- **MOUSE** : ANY MOUSE
- **RAM** : 8GB

### **4.1.1.2 SOFTWARE REQUIREMENTS**

- **OPERATING SYSTEM** : WINDOWS 10 (64 bit)
- **CODING LANGUAGE** : PYTHON-FLASK
- **SCRIPTING LANGUAGES** : HTML, JAVASCRIPT
- **CLOUD APPLICATIONS** : Docker ,Kubernetes, IBM Cloud, IBM Cloud Object Storage, IBM DB2,IBM Container Registry

## 4.1.2 SOFTWARE REQUIREMENTS

### PYTHON

Python is an interpreter, object-oriented, high-level programming language (Shown in Figure: 2.4.1) with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.



**Figure:2.4.1 Python Logo**

The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.

## **PYTHON - FLASK**

Website development is more of an art than a skill. It requires patience and diligence, along with perseverance, courage, and dedication to create what's necessary for it to be a real success. These days, it is essential for learners to get to speed as soon as possible.

Flask is called a "micro" framework because it doesn't directly provide features like form validation, database abstraction, authentication, and so on. Such features are instead provided by special Python packages called Flask extensions. The extensions integrate seamlessly with Flask so that they appear as if they were part of Flask itself. For example, Flask doesn't provide a page template engine, but installing Flask includes the Jinja templating engine by default. For convenience, we typically speak of these defaults as part of Flask.

In this Flask tutorial, you create a simple Flask app with three pages that use a common base template. Along the way, you experience a number of features of Visual Studio Code including using the terminal, the editor, the debugger, code snippets, and more.

Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

## **HYPER TEXT MARKUP LANGUAGE (HTML)**

HTML stands for Hyper Text Markup Language, which is the most widely used language on Web to develop web pages. HTML was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having

HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers. Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

HTML is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain.

## **APPLICATIONS OF HTML**

As mentioned before, HTML is one of the most widely used language over the web. I'm going to list few of them here:

- **Web pages development** - HTML is used to create pages which are rendered over the web. Almost every page of web is having html tags in it to render its details in browser.
- **Internet Navigation** - HTML provides tags which are used to navigate from one page to another and is heavily used in internet navigation.
- **Responsive UI** - HTML pages now-a-days works well on all platform, mobile, tabs, desktop or laptops owing to responsive design strategy.
- **Game development**- HTML5 has native support for rich experience and is now useful in gaming development arena as well.

## **JAVASCRIPT**

JavaScript often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js. Although Java and JavaScript are similar in name, syntax, and respective standard libraries.

## **4.1.2 CLOUD APPLICATIONS**

### **DOCKER**

Docker is an open source containerization platform. It enables developers to package applications into containers-standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

### **KUBERNETS**

Kubernetes, or K8s for short, is an open-source container-orchestration tool designed by Google. It's used for bundling and managing clusters of containerized applications a process known as 'orchestration' in the computing world. The name Kubernetes originates from Greek, meaning helmsman or pilot.

## **IBM CLOUD**

IBM Cloud Paks are software products for hybrid clouds that enable you to develop apps once and deploy them anywhere. Virtual Private Cloud (VPC) is available as a public cloud service that lets you establish your own private cloud-like computing environment on shared public cloud infrastructure.

## **IBM CLOUD OBJECT STORAGE**

Object storage is a data storage architecture for storing unstructured data, which sections data into units, objects and stores them in a structurally flat data environment. Each object includes the data, metadata, and a unique identifier that applications can use for easy access and retrieval.

## **IBM DB2**

IBM Db2 is a family of data management products, including the Db2 relational database. The products feature AI-powered capabilities to help you modernize the management of both structured and unstructured data across on-premises and multicloud environments.

## **IBM CONTAINER REGISTRY**

IBM Cloud Container Registry provides a multi-tenant private image registry that you can use to store and share your container images with users in your IBM Cloud account. The IBM Cloud console includes a brief Quick Start.

Container Registry is a single place for your team to manage Docker images, perform vulnerability analysis, and decide who can access what with fine-grained access control. Existing CI/CD integrations let you set up fully automated Docker pipelines to get fast feedback.



## **VISUAL STUDIO IDE**

An integrated development environment (IDE) is a feature-rich program that supports many aspects of software development. The Visual Studio IDE is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to enhance the software development process.

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

## **PYCHARM**

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

PyCharm keeps an eye on what you are currently doing and makes smart suggestions, called intention actions, to save more of your time. Indicated with a lightbulb, intention actions let you apply automatic changes to code that is correct i.e., in contrast to code inspections that provide quick-fixes for code that may be incorrect.

## 4.2 NON-FUNCTIONAL REQUIREMENTS

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

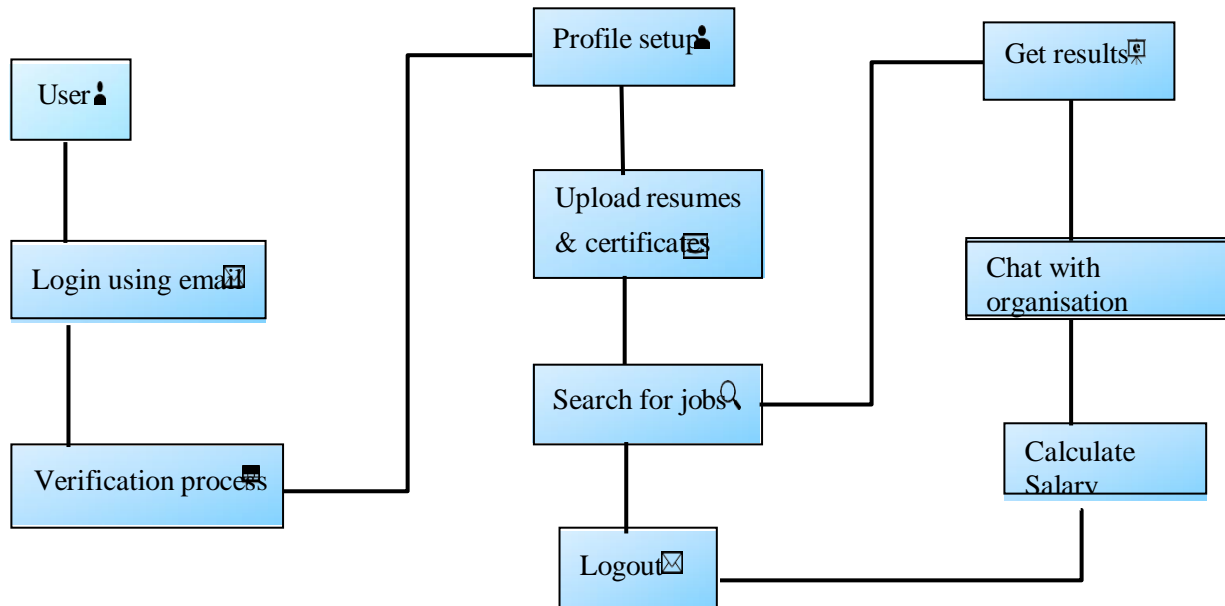
<b>FR No.</b>	<b>Non-Functional Requirement</b>	<b>Description</b>
NFR-1	<b>Usability</b>	The webpage will be designed in such a way that any non-technical user can easily navigate through it and complete the job registration work. (easy and simple design)
NFR-2	<b>Security</b>	Using of python flask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

## CHAPTER 5

### PROJECT DESIGN

#### 5.1 DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and 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

##### 5.2.1 SOLUTION ARCHITECTURE

###### INTRODUCTION

Every solution architecture design contains 6 to 7 phases these standards should be followed by all development team to ensure the standard of the software, so the software is scalable, versatile and reusable

## **REQUIREMENTS**

This project is done using the Flask framework for backend development, and other required packages like flask-login, flask-sqlalchemy, flask-form, security packages etc.. For frontend development css, HTML, javascript is used along with css framework like bootstrap. For API testing postman application is used, and for deployment IBM cloud service is used.

## **DESIGN**

All the requirements are used to design the software. The design and architecture of the software is done in a unique manner so the software can be reused and developed in future. The routers are programmed in routers.py file, The forms used in the software are developed in forms.py file. The database model is created in model.py file, the testings are done in separate tests.py file. Finally HTML files are stored in templates folder and static file is stored in static folder

## **IMPLEMENTATION**

The designing process is done and implementation is done by developing the logic by coding. All the required packages are imported and for each router specific logic is developed according to the use.

## **UNIT TESTING**

Each part of the software is developed by individual team members, and it is tested individually by the python unit testing package.

## **INTEGRATION AND TESTING**

After unit testing all parts of the software are integrated and tested finally, so the flask application can be runned in any platform. The testing process includes Alpha testing and Beta testing.

## **DEPLOYMENT**

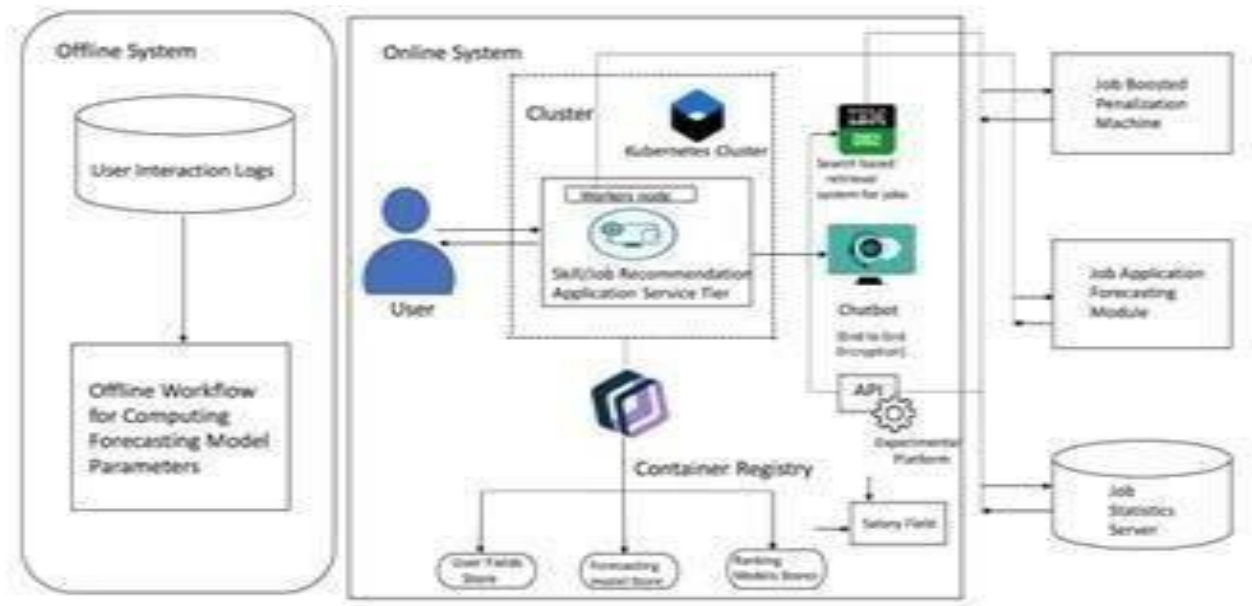
The flask application is finally deployed in IAAS platform like IBM cloud service, so it can be runned in HTTPS protocol along with SSL. In the deployment process a real time database is connected along with real time file storage.

## MAINTENANCE

After successful deployment, if there is a package update, it is implemented in the software

### 5.2.2 TECHNICAL ARCHITECTURE

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



#### Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API's etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)

**Table-1 : Components & Technologies:**

<b>S.N o</b>	<b>Component</b>	<b>Description</b>	<b>Technology</b>
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js /React Js etc.
2.	Developing Interface	Developing application for the task	Java / Python
3.	Voice Assistance	Voice commands instead of typing.	IBM Watson STT service
4.	Chatbot Assistance	Conversational Interface	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other StorageService or Local Filesystem
8.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.

**Table-2: Application Characteristics:**

<b>S.No</b>	<b>Characteristics</b>	<b>Description</b>	<b>Technology</b>
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Artificial Intelligence (AI)
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	RAID(redundant array of independent disks)
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	DRAM or flash memory

## REFERENCES:

<https://c4model.com/>

<https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/>

<https://www.ibm.com/cloud/architecture>

<https://aws.amazon.com/architecture>

<https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d>

## 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	I can receive confirmation email & click	High	Sprint-1



			have registered			
		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

## CHAPTER 6

### PROJECT PLANNING & SCHEDULING

#### 6.1 SPRINT PLANNING & ESTIMATION

Title	Description	Date
Literature Survey and Information Gathering	Gathering Information by referring the technical papers, research publications etc	2 SEPTEMBER 2022
Prepare Empathy Map	To capture user pain and gains Prepare List of Problem Statement	10 SEPTEMBER 2022
Ideation	Prioritise a top 3 ideas based on feasibility and Importance	17 SEPTEMBER 2022
Proposed Solution	Solution include novelty, feasibility, business model, social impact and scalability of solution	24 SEPTEMBER 2022
Problem Solution Fit	Solution fit document	29 SEPTEMBER 2022
Solution Architecture	Solution Architecture	1 October 2022
Customer Journey	To Understand User Interactions and experiences with application	8 October 2022
Functional Requirement	Prepare functional Requirement	14 October 2022
Data flow Diagrams	Data flow diagram	15 October 2022
Technology Architecture	Technology Architecture diagram	16 October 2022
Milestone & sprint delivery plan	Activity what we done & further plans	21 October 2022
Project Development Delivery of sprint 1,2,3 & 4	Develop and submit the developed code by testing it	24 October 2022 – 19 November 2022

### 6.1.1 SPRINT SCHEDULE

<b>Sprint</b>	<b>Functional Requirement (Epic)</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Priority</b>	<b>Acceptance criteria</b>	<b>Team Members</b>
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	Mukesh , Siva Kumar
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	Mukesh,Siva Kumar, Gnanashankaran, Kiritheeswaran
Sprint-1		USN-3	As a user, I can register for the	Medium	I can receive confirmation email &	Mukesh,Siva Kumar

			application through Gmail		click confirm	
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	Mukesh,Siva Kumar
Sprint-1	Flask	USN-5	As a user, I can access the website in a second	High	I can access my account / dashboard	Siva Kumar, Gnanashankar

<b>Functional Requirement (Epic)</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Priority</b>	<b>Acceptance criteria</b>	<b>Team Members</b>
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	Mukesh,Siva Kumar, Gnanashankaran, Kiritheeswaran
		Submission Of Sprint-1			
User Profile	USN-7	As a user, I can view and update my details	Medium	I can modify my details/data	Gnanashankaran, Kiritheeswaran
Database	USN-8	As a user, I can store my details	Medium	I can store my data	Mukesh,Siva Kumar

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	Gnanashankaran, Kiritheeswaran
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	Mukesh, Siva Kumar, Gnanashankaran,
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	Mukesh, Siva Kumar, Gnanashankaran,

<b>Sprint</b>	<b>Functional Requirement (Epic)</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Priority</b>	<b>Acceptance criteria</b>	<b>Team Members</b>
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 and skills	Mukesh, Siva Kumar
Sprint-3	Docker	USN-13	As a user, I can access the website in any device	High	I can access my account in any device	Siva Kumar, Gnanashankaran
Sprint-3	Kubernetes	USN-14	As a user, I can access the website in any device	High	I can access my account in any device	Mukesh, Siva Kumar, Gnanashankaran, Kiritheeswaran
Sprint-3	Deployment in cloud	USN-15	As a user, I can access the website	High	I can access my account in any device	Siva Kumar, Gnanashankaran, Kiritheeswaran



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.	Siva Kumar, Gnanashankara n, Kiritheeswaran
			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 without any interruption	Mukesh, Kiritheeswaran
Sprint-4	Integration testing	USN-18	As a user, I can access the website without any interruption	High	I can access the website without any interruption	Siva Kumar, Gnanashankar an

<b>Sprint</b>	<b>Functional Requirement (Epic)</b>	<b>User Story Number</b>	<b>User Story / Task</b>	<b>Priority</b>	<b>Acceptance criteria</b>	<b>Team Members</b>
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	Siva Kumar, Gnanashankar
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	Siva Kumar, Gnanashankar, Kiritheeswaran
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	Mukesh
			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

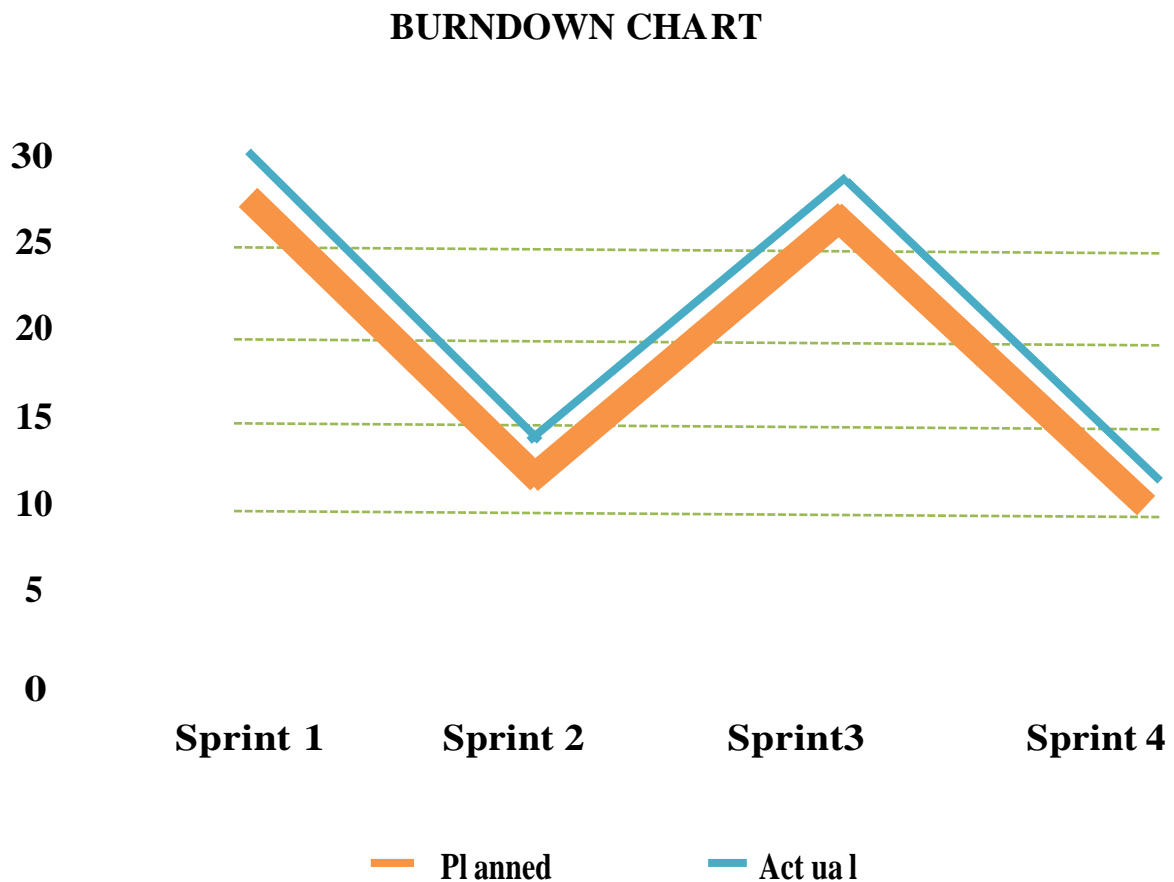
### VELOCITY:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

## BURNDOWN CHART:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



## CHAPTER 7

### CODING & SOLUTIONING

#### 7.1 FEATURE 1

##### BACK END

##### \_INIT\_.PY

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

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

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

```

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

## **MIDDLE WARE**

### **AUTH\_MIDDLEWARE.PY**

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

# Middleware function that checks for JWT token in header
# All routes that have the @token_required decorator will be protected
def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = None
        if "Authorization" in request.headers:
            token = request.headers["Authorization"].split(" ")[1]
        if not token:
            return {
                "error": "Unauthorized"
            }, 401
        try:
            # Get the user's email from the decoded token
            data = jwt.decode(
                token, config["APP_SECRET"], algorithms=["HS256"])
```

```

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

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

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

return decorated

```

## **ROUTING**

### **AUTH\_ROUTER.PY**

```

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

```



```

import ibm_db

auth = Blueprint("auth", __name__)

LOGIN_FIELDS = ('email', 'password')
SIGNUP_FIELDS = ('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_FIELDS:
        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 already in use!"}), 409

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

```

## **FILES ROUTING**

### **FILES\_ROUTER.PY**

```

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

```

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

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

```
def multi_part_upload(bucket_name, item_name, file_path):
```

```
    try:
```

```
        print("Starting file transfer for {0} to bucket: {1}\n".format(
            item_name, bucket_name))
```

```
        # set 5 MB chunks
```

```
        part_size = 1024 * 1024 * 5
```

```
        # set threshold to 15 MB
```

```
        file_threshold = 1024 * 1024 * 15
```

```
        # set the transfer threshold and chunk size
```

```
        transfer_config = ibm_boto3.s3.transfer.TransferConfig(
            multipart_threshold=file_threshold,
            multipart_chunksize=part_size
        )
```

```
        # the upload_fileobj method will automatically execute a multi-part upload
```

```

# in 5 MB chunks for all files over 15 MB
with open(file_path, "rb") as file_data:
    cos.Object(bucket_name, item_name).upload_fileobj(
        Fileobj=file_data,
        Config=transfer_config
    )

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

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

```

## REQUIREMENTS

### REQUIREMENTS.TXT

```

bcrypt==4.0.1
certifi==2022.9.24
cffi==1.15.1
charset-normalizer==2.1.1
click==8.1.3

```

colorama==0.4.5  
cryptography==38.0.1  
Flask==2.2.2  
Flask-Cors==3.0.10  
ibm-cos-sdk==2.12.0  
ibm-cos-sdk-core==2.12.0  
ibm-cos-sdk-s3transfer==2.12.0  
ibm-db==3.1.3  
idna==3.4  
itsdangerous==2.1.2  
Jinja2==3.1.2  
jmespath==0.10.0  
MarkupSafe==2.1.1  
pycparser==2.21  
PyJWT==2.6.0  
python-dateutil==2.8.2  
python-dotenv==0.21.0  
requests==2.28.1  
six==1.16.0  
urllib3==1.26.12  
waitress==2.1.2  
Werkzeug==2.2.2

## **USER ROUTING**

### **USER\_ROUTER.PY**

```
from flask import Blueprint, jsonify, request  
from backend import conn  
from backend.auth_middleware import token_required
```

```

import ibm_db

user = Blueprint("user", __name__)

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

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

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

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

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

```

```

# Get the skills from the request
if not ('skills' in request.json):
    return jsonify({"error": f"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

```

## 7.2 FEATURES 2

### 7.2.1 SOURCE CODE

#### 7.2.1.1 COMPONENTS

#### JOB CARD.JSX

```

import React, { useEffect } from "react";

const JobCard = ({ title, company, description, link }) => {
    return (
        <div className="max-w-sm flex flex-col rounded overflow-hidden shadow-lg
border-2 border-slate-200">
            <div className="px-6 py-4">
                <div className="font-bold text-xl">{title}</div>

```

```

    <div className="text mb-2 text-gray-400">{company}</div>
    <p className="text-ellipsis overflow-hidden text-gray-800 text-sm">
      {description}
    </p>
  </div>

  <div className="px-6 pt-4 pb-2 mt-auto mb-2">
    <a
      href={link}
      target="__blank"
      className="bg-transparent hover:bg-purple-400 text-purple-400 font-
semibold hover:text-white py-2 mb-0 mt-4 px-4 border border-purple-400
hover:border-transparent rounded"
    >
      Apply
    </a>
  </div>
</>
</div>
);
};

export default JobCard;

```

## LOGIN PAGE

### LOGIN.JSX

```
import { useToast } from "@chakra-ui/react";
import React, { useContext, useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import { AppContext } from "../context/AppContext";
import { loginUser } from "../proxies/backend_api";
import { emailRegex } from "../utils/helper";
```

```
const Login = () => {
  const toast = useToast();
  const { setUser } = useContext(AppContext);
```

```
  const navigate = useNavigate();
```

```
  const [inputs, setInputs] = useState({
    email: "",
    password: "",
  });
```

```
  const [error, setErrors] = useState({
    email: "",
    password: "",
  });
```

```
  const handleChange = ({ target: { name, value } }) => {
    setErrors((prev) => {
      return { ...prev, [name]: "" };
    });
```

```
setInputs((prev) => ({ ...prev, [name]: value }));  
};
```

```
const checkInputErrors = () => {  
  let status = true;  
  if (inputs.email.trim() === "" || !emailRegex.test(inputs.email.trim())) {  
    setErrors((prev) => {  
      return { ...prev, email: "Enter a valid email" };  
    });  
    status = false;  
  }  
}
```

```
if (inputs.password.trim() === "") {  
  setErrors((prev) => {  
    return { ...prev, password: "Enter a valid password" };  
  });  
  status = false;  
}
```

```
if (inputs.password.trim().length < 6) {  
  setErrors((prev) => {  
    return { ...prev, password: "Minimum 6 characters" };  
  });  
  status = false;  
}  
return status;  
};
```

```

const handleLogin = async () => {
  if (checkInputErrors()) {
    const data = await loginUser(inputs);
    if (data.error) {
      toast({
        title: data.error,
        status: "error",
        duration: 3000,
        isClosable: true,
        variant: "left-accent",
        position: "top",
      });
      return;
    }
    setUser(data);
    toast({
      title: `Welcome back ${data.name}`,
      status: "success",
      duration: 3000,
      isClosable: true,
      variant: "left-accent",
      position: "top",
    });
    localStorage.setItem("user", JSON.stringify(data));
    navigate("/dashboard");
  }
}

```

```

};

return (
  <>
    <div>
      <button className="bg-base-300 rounded-box flex flex-row justify-evenly
items-center gap-10 px-10 py-5 w-fit mx-auto">
        <span>Sign in with Github</span>
        <img src={`github-dark.png`} alt="github" width="14%" />
      </button>
      <div className="divider max-w-xs">or</div>
      <form
        onSubmit={(e) => e.preventDefault()}
        className="card bg-base-300 rounded-box flex flex-col justify-center
items-center gap-5 px-10 py-5 w-fit mx-auto"
      >
        <div>
          <input
            value={inputs.email}
            type="text"
            name="email"
            placeholder="email"
            className="input input-bordered input-primary w-full"
            onChange={handleChange}
          />
          {error.email !== "" && (
            <p className="text-sm text-red-500 mt-1 font-medium">

```

```

        {error.email}
      </p>
    )}
  </div>
  <div>
    <input
      value={inputs.password}
      type="password"
      name="password"
      placeholder="password"
      className="input input-bordered input-primary w-full"
      onChange={handleChange}
    />
    {error.password !== "" && (
      <p className="text-sm text-red-500 mt-1 font-medium">
        {error.password}
      </p>
    )}
  </div>
  <div className="text-center">
    <button
      type="submit"
      onClick={handleLogin}
      className="btn btn-sm btn-primary mb-4"
    >
      Login
    </button>
  </div>

```



```
        </div>
      </form>
    </div>
  </>
);
};
```

```
export default Login;
```

## **NAVIGATION BAR**

### **NAVBAR.JSX**

```
import { useToast } from "@chakra-ui/react";
import React, { useContext } from "react";
import { Link, useNavigate } from "react-router-dom";
import { AppContext } from "../context/AppContext";

const Navbar = () => {
  const navigate = useNavigate();

  const toast = useToast();

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

  const logout = () => {
    setUser(null);

    setSkills([]);
```

```
toast({
  title: "Logged out successfully!",
  status: "info",
  duration: 3000,
  isClosable: true,
  variant: "left-accent",
  position: "top",
});
```

```
localStorage.removeItem("user");
```

```
navigate("/");
};
```

```
return (
  <div className="navbar bg-base-100 border-b-2">
    <div className="flex-1">
      <Link
        className="btn btn-ghost normal-case text-xl"
        to={user ? "/dashboard" : "/"}
      >
        F-ing Jobs
      </Link>
    </div>
    {user && (
      <div className="flex-none gap-2">
```

```

<div className="dropdown dropdown-end">
  <label tabIndex={0} className="btn btn-ghost btn-circle avatar ">
    <div className="w-10 rounded-full ring ring-opacity-50 ring-purple-
700">
      
    </div>
  </label>
  <ul
    tabIndex={0}
    className="mt-3 p-2 shadow menu menu-compact dropdown-content bg-
base-100 rounded-box w-52"
  >
    <li>
      <a
        className="justify-between"
        onClick={() => navigate("/profile")}
      >
        Profile
      </a>
    </li>
    <li>
      <a onClick={logout}>Logout</a>
    </li>
  </ul>
</div>
</div>
)}

```

```

    </div>
  );
};

export default Navbar;

```

## SEARCH BAR

### SEARCHBAR.JSX

```

import React from "react";
import { BsSearch } from "react-icons/bs";

const SearchBar = ({ setquery, onClick }) => {
  const handlesubmit = (e) => {
    e.preventDefault();
    onClick();
  };

  return (
    <form className="flex items-center" onSubmit={handlesubmit}>
      <label htmlFor="simple-search" className="sr-only">
        Search
      </label>
      <div className="relative w-full">
        <div className="flex absolute inset-y-0 left-0 items-center pl-3 pointer-events-none">
          <BsSearch />
        </div>

```

```

    <input
      onChange={ (e) => setquery(e.target.value)}
      name="search"
      type="text"
      id="simple-search"
      className="bg-gray-50 border border-gray-300 text-gray-900 text-sm
rounded-lg focus:ring-blue-500 focus:border-blue-500 block w-full pl-10 p-2.5
dark:bg-gray-700 dark:border-gray-600 dark:placeholder-gray-400 dark:text-white
dark:focus:ring-blue-500 dark:focus:border-blue-500"
      placeholder="Search"
      required=""
    />
  </div>
  <button
    type="submit"
    className="p-2.5 ml-2 text-sm font-medium text-white bg-purple-700
rounded-lg border border-purple-700 hover:bg-purple-800 focus:ring-4
focus:outline-none focus:ring-purple-300"
  >
    <BsSearch />
    <span className="sr-only">Search</span>
  </button>
</form>
);
};

export default SearchBar;

```

## **SIGHUP BAR**

### **SIGNUP.JSX**

```
import { useToast } from "@chakra-ui/react";
import React, { useContext, useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import { AppContext } from "../context/AppContext";
import { registerUser } from "../proxies/backend_api";
import { emailRegex } from "../utils/helper";
```

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

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

```
  const navigate = useNavigate();
```

```
  const [inputs, setInputs] = useState({
    name: "",
    email: "",
    phone_number: "",
    password: "",
    confirm_password: "",
  });
```

```
  const [error, setErrors] = useState({
    name: "",
```

```
email: "",
phone_number: "",
password: "",
confirm_password: "",
});
```

```
const handleChange = ({ target: { name, value } }) => {
  setErrors((prev) => {
    return { ...prev, [name]: "" };
  });
  setInputs((prev) => ({ ...prev, [name]: value }));
};
```

```
const checkInputErrors = () => {
  let status = true;
  if (inputs.email.trim() === "" || !emailRegex.test(inputs.email.trim())) {
    setErrors((prev) => {
      return { ...prev, email: "Enter a valid email" };
    });
    status = false;
  }
}
```

```
if (inputs.name.trim() === "") {
  setErrors((prev) => {
    return { ...prev, name: "Enter a valid name" };
  });
  status = false;
}
```

```
}
```

```
if (inputs.phone_number.trim() === "") {  
  setErrors((prev) => {  
    return { ...prev, phone_number: "Enter a valid phone number" };  
  });  
  status = false;  
}
```

```
if (inputs.confirm_password.trim() === "") {  
  setErrors((prev) => {  
    return { ...prev, confirm_password: "Enter a valid password" };  
  });  
  status = false;  
}
```

```
if (inputs.password.trim() === "") {  
  setErrors((prev) => {  
    return { ...prev, password: "Enter a valid password" };  
  });  
  status = false;  
}
```

```
if (inputs.password.trim().length < 6) {  
  setErrors((prev) => {  
    return { ...prev, password: "Minimum 6 characters" };  
  });  
}
```



```

    status = false;
  }

  if (inputs.password.trim() !== inputs.confirm_password.trim()) {
    setErrors((prev) => {
      return { ...prev, confirmPassword: "Password don't match" };
    });
    status = false;
  }
  return status;
};

const handleSignUp = async () => {
  if (checkInputErrors()) {
    const data = await registerUser(inputs);
    if (data.error) {
      toast({
        title: data.error,
        status: "error",
        duration: 3000,
        isClosable: true,
        variant: "left-accent",
        position: "top",
      });
      return;
    }
    setUser(data);
  }

```

```

toast({
  title: `Your journey starts here ${data.name}`,
  status: "success",
  duration: 3000,
  isClosable: true,
  variant: "left-accent",
  position: "top",
});
localStorage.setItem("user", JSON.stringify(data));
navigate("/dashboard");
}
};

```

```

return (
  <>
  <div>
    <button className="bg-base-300 rounded-box flex flex-row justify-evenly
items-center gap-10 px-10 py-5 w-fit mx-auto">
      <span>Sign in with Github</span>
      <img src={`github-dark.png`} alt="github" width="14%" />
    </button>
    <div className="divider max-w-xs">or</div>
    <form
      onSubmit={(e) => {
        e.preventDefault();
        handleSignUp();
      }}
    >

```

```
className="card bg-base-300 rounded-box flex flex-col justify-center
items-center gap-3 px-10 py-5 w-fit mx-auto"
```

```
>
```

```
<div>
```

```
<input
```

```
  value={inputs.name}
```

```
  type="text"
```

```
  name="name"
```

```
  placeholder="name"
```

```
  className="input input-bordered input-primary w-full"
```

```
  onChange={handleChange}
```

```
/>
```

```
{error.name !== "" && (
```

```
  <p className="text-sm text-red-500 font-medium">{error.name}</p>
```

```
)}
```

```
</div>
```

```
<div>
```

```
<input
```

```
  value={inputs.email}
```

```
  type="text"
```

```
  name="email"
```

```
  placeholder="email"
```

```
  className="input input-bordered input-primary w-full"
```

```
  onChange={handleChange}
```

```
/>
```

```
{error.email !== "" && (
```

```
  <p className="text-sm text-red-500 font-medium">{error.email}</p>
```

```

    })
  </div>
  <div>
    <input
      value={inputs.phone_number}
      type="number"
      name="phone_number"
      placeholder="phone number"
      className="input input-bordered input-primary w-full"
      onChange={handleChange}
    />
    {error.phone_number !== "" && (
      <p className="text-sm text-red-500 font-medium">
        {error.phone_number}
      </p>
    )}
  </div>
  <div>
    <input
      value={inputs.password}
      type="password"
      name="password"
      placeholder="password"
      className="input input-bordered input-primary w-full"
      onChange={handleChange}
    />
    {error.password !== "" && (

```

```

    <p className="text-sm text-red-500 font-medium">
      {error.password}
    </p>
  )}
</div>
<div>
  <input
    value={inputs.confirm_password}
    type="password"
    name="confirm_password"
    placeholder="confirm password"
    className="input input-bordered input-primary w-full"
    onChange={handleChange}
  />
  {error.confirm_password !== "" && (
    <p className="text-sm text-red-500 font-medium">
      {error.confirm_password}
    </p>
  )}
</div>
<div className="text-center">
  <button
    onClick={handleSignUp}
    className="btn btn-sm btn-primary mb-4"
  >
    Sign Up
  </button>

```

```

        </div>
      </form>
    </div>
  </>
);
};

export default SignUp;

```

## **SKILL**

### **SKILL.JSX**

```

import React, { useEffect, useState } from "react";

const Skill = ({ skill, setSelectedSkills, disabled }) => {
  const [isSelected, setIsSelected] = useState(false);

  useEffect(() => {
    if (isSelected) {
      setSelectedSkills((prev) => [...prev, skill]);
    } else {
      setSelectedSkills((prev) => prev.filter((item) => item !== skill));
    }
  }, [isSelected]);

  return (
    <li className="hover:text-white flex gap-1 items-center justify-between p-1 rounded-sm">

```

```

    {skill}
    <button
      disabled={disabled}
      onClick={() => setIsSelected(!isSelected)}
      className={`cursor-pointer border-2 ${
        !isSelected ? "border-green-500" : "border-red-400"
      } p-1 rounded-lg`}
    >
      {!isSelected ? "Add" : "Remove"}
    </button>
  </li>
);
};

```

```
export default Skill;
```

### 7.2.1.2 CONTEXT

#### APPCONTEXT.JSX

```
import { createContext, useEffect, useState } from "react";
```

```
import { useNavigate } from "react-router-dom";
```

```
export const AppContext = createContext();
```

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

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

```

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

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

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

```

### 7.2.1.3 PROXIES

#### BACKEND\_API.JS

```

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

export const loginUser = async (inputs) => {
  try {
    const response = await fetch(`${BASE_URL}/auth/login`, {
      method: "POST",
      body: JSON.stringify(inputs),

```



```

    headers: {
      "Content-Type": "application/json",
    },
  });
  const data = await response.json();
  return data;
} catch (error) {
  console.error(error);
}
};

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

```

```

export const getUserSkills = async (token) => {
  try {
    const response = await fetch(`${BASE_URL}/user/skills`, {
      method: "GET",
      headers: {
        Authorization: `Bearer ${token}`,
        "Content-Type": "application/json",
      },
    });
    if (response.ok) {
      const { skills } = await response.json();
      return skills;
    } else {
      return null;
    }
  } catch (error) {
    console.error(error);
  }
};

```

```

export const saveUserSkills = async (skills, token) => {
  try {
    const response = await fetch(`${BASE_URL}/user/skills`, {
      method: "POST",
      body: JSON.stringify({ skills }),
      headers: {
        Authorization: `Bearer ${token}`,

```

```

    "Content-Type": "application/json",
  },
});
if (response.ok) {
  return true;
} else {
  return false;
}
} catch (error) {
  console.error(error);
}
};

export const removeUserSkills = async (skills, token) => {
  try {
    const response = await fetch(`${BASE_URL}/user/skills`, {
      method: "DELETE",
      body: JSON.stringify({ skills }),
      headers: {
        Authorization: `Bearer ${token}`,
        "Content-Type": "application/json",
      },
    });
    if (response.ok) {
      return true;
    } else {
      return false;
    }
  }
};

```

```

    }
  } catch (error) {
    console.error(error);
  }
};

```

```

export const updateUserDetails = async (inputs, token) => {
  try {
    const response = await fetch(`${BASE_URL}/user/profile`, {
      method: "POST",
      body: JSON.stringify(inputs),
      headers: {
        Authorization: `Bearer ${token}`,
        "Content-Type": "application/json",
      },
    });
    if (response.ok) {
      const data = await response.json();
      return data;
    } else {
      return null;
    }
  } catch (error) {
    console.error(error);
  }
};

```

#### 7.2.1.4 SCREENS

## AUTHENTICATOR

### AUTH.JSX

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

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

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

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

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

```

        <Login />
      </TabPanel>
    <TabPanel>
      <SignUp />
    </TabPanel>
  </TabPanels>
</Tabs>
</div>
);
};

```

export default Auth;

## **DASHBOARD**

### **DASHBOARD.JSX**

```

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

```

```

import { getUserSkills } from "../proxies/backend_api";
import { getBaseUrl, getBaseUrl_with_skills } from "../utils/helper";

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

  const toast = useToast();

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

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

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

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

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

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

    try {
      if (query !== "" || !posts) {
        const { data } = await axios.get(getBaseUrl(query));

        setPosts(data.results);
      }
    }
  }

```

```

    } catch (error) {
      toast({
        title: `Trying to fetch data!!`,
        description: "Something went wrong",
        status: "info",
        duration: 3000,
        isClosable: true,
        variant: "left-accent",
        position: "top",
      });
    }

    setJobsLoading(false);
  };

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

  try {
    const { data } = await axios.get(
      getBaseUrl_with_skills(query, selectedSkills)
    );

    setPosts(data.results);
  } catch (error) {
    toast({
      title: `Trying to fetch data!!`,

```



```
    description: "Something went wrong",
    status: "info",
    duration: 3000,
    isClosable: true,
    variant: "left-accent",
    position: "top",
  });
}
```

```
setJobsLoading(false);
};
```

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

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

```
useEffect(() => {
```

```

    searchJobsFromQuery();
  }, []);

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

        </div>
        {skillsLoading ? (
          <Spinner
            className="self-center my-5"
            thickness="3px"
            speed="0.65s"
            emptyColor="gray.200"
            color="black.100"
            size="lg"
          />
        ) : (
          <ul className="list-none text-gray-200 flex flex-col gap-2">
            {skills?.length === 0 ? (

```

```

    <p className="text-gray-300">
      Skills you add in the profile section will appear here!!
    </p>
  ) : (
    skills.map((skill, ind) => (
      <Skill
        skill={ skill}
        key={ ind}
        setSelectedSkills={ setSelectedSkills}
        disabled={ skillsLoading}
      />
    ))
  )}
</ul>
)}

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

<div className="mx-auto min-w-[80%] ">
  <SearchBar setquery={ setquery} onClick={ searchJobsFromQuery} />
  { query === "" ? (
    <h2 className="text-2xl mt-5">Recommended Jobs</h2>
  ) : (
    <h2 className="text-2xl mt-5">
      Search for keywords { query}

```

```

    {`,${selectedSkills.join(",")}`}
  </h2>
)}

```

```

<div className="mt-5 grid grid-cols-1 lg:grid-cols-3 md:grid-cols-2 gap-
5">

```

```

  {jobsLoading
    ? [...new Array(10)].map((_, ind) => (
      <div key={ind}>
        <SkeletonCircle size="8" className="mb-5" />
        <SkeletonText
          mt="4"
          noOfLines={8}
          spacing="4"
          color={"red"}
        />
      </div>
    ))
    : posts?.map((post, ind) => (
      <JobCard
        key={ind}
        title={post.title}
        company={post.company.display_name}
        description={post.description}
        link={post.redirect_url}
      />
    ))}

```

```

        </div>
    </div>
</div>
</>
);
};

```

```
export default Dashboard;
```

## **PROFILE**

### **PROFILE.JSX**

```

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

```

```
getUserSkills,  
removeUserSkills,  
saveUserSkills,  
updateUserDetails,  
} from "../proxies/backend_api";
```

```
const Profile = () => {  
  const toast = useToast();  
  
  const { user, setUser, skills, setSkills } = useContext(AppContext);  
  
  const [addSkill, setAddSkill] = useState("");  
  
  const [newSkills, setNewSkills] = useState([]);  
  
  const [removedSkills, setRemovedSkills] = useState([]);  
  
  const [isEditingEnabled, setIsEditingEnabled] = useState(false);  
  
  const [loading, setLoading] = useState(false);  
  
  const [userInfo, setUserInfo] = useState({  
    name: "",  
    phone_number: "",  
  });  
  
  const handleUserInfoChange = ({ target: { name, value } }) => {
```

```

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

const changeSkills = () => {
  if (
    addSkill !== "" &&
    !skills.find((item) => item.toLowerCase() === addSkill.toLowerCase())
  ) {
    setNewSkills((prev) => [...prev, addSkill.trim()]);
    setSkills((prev) => [...prev, addSkill.trim()]);
  }
  setAddSkill("");
};

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

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

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

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

  let skillsAdded = false,
    skillsRemoved = false;

```

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

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

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

setNewSkills([]);

setRemovedSkills([]);

setLoading(false);
};
```



```
const updateUserInfo = async () => {  
  setLoading(true);  
  
  const data = await updateUserDetails(userInfo, user.token);  
  
  if (data) {  
    setUser((prev) => {  
      prev = { ...prev, name: data.name, phone_number: data.phone_number };  
  
      localStorage.setItem("user", JSON.stringify(prev));  
  
      return prev;  
    });  
  
    toast({  
      title: "Profile Updated!",  
      status: "info",  
      duration: 3000,  
      isClosable: true,  
      variant: "left-accent",  
      position: "top",  
    });  
  }  
  
  setLoading(false);  
  
  setIsEditingEnabled(false);  
}
```

```

};

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

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

      if (data) setSkills(data);

      setLoading(false);
    })();

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

return (
  <>
    {loading && <Progress size="xs" isIndeterminate colorScheme={"purple"}
  />}
  <div className="my-5 mx-10">

```

```

<div className="border-2 border-blue-100 w-full h-fit rounded-xl p-5 flex
flex-col gap-3">
  <div className="flex justify-between w-full min-h-[25vh]">
    <div className="flex flex-col justify-between">
      <h1 className="md:text-2xl text-xl font-medium flex items-center gap-
4">
        Your Profile{ " "}
      <button>
        {isEditingEnabled ? (
          <AiOutlineClose
            color="#ff8977"
            onClick={() => setIsEditingEnabled(!isEditingEnabled)}
          />
        ) : (
          <RiEdit2Fill
            color="#4506cb"
            onClick={() => setIsEditingEnabled(!isEditingEnabled)}
          />
        )}
      </button>
    </h1>
  <div className="flex flex-col gap-3">
    {isEditingEnabled ? (
      <
        <input
          name="name"
          value={userInfo.name}

```

```

primary"
    className="input input-bordered w-full input-xs p-3 text-lg input-
type="text"
placeholder="name"
onChange={handleUserInfoChange}
/>
<input
  disabled
  value={user?.email}
  className="input input-bordered w-full input-xs p-3 text-lg input-
primary"
  type="text"
  placeholder="name"
/>
<input
  name="phone_number"
  value={userInfo.phone_number}
  className="input input-bordered w-full input-xs p-3 text-lg input-
primary"
  type="number"
  placeholder="phone number"
  onChange={handleUserInfoChange}
/>
<button
  className="btn btn-xs btn-outline btn-primary"
  onClick={updateUserInfo}
>

```

```

        Update
      </button>
    </>
  ) : (
    <>
      <h2 className="md:text-2xl xl:text-2xl sm:text-xl">
        {user?.name}
      </h2>
      <p className="md:text-xl sm:text-md text-gray-700">
        {user?.email}
      </p>
      <span className="text-gray-700">{user?.phone_number}</span>
    </>
  )}
</div>
</div>
<div className="flex flex-col justify-end w-fit gap-4">
  
</div>
</div>
<div className="divider my-2"></div>
<div className="flex flex-col">
  <div className="flex justify-between gap-2 flex-col">

```

```

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

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

```

```

        emptyColor="gray.200"
        color="blue.500"
        size="md"
        className="m-3"
    />
) : (
    <ul className="flex gap-2 flex-wrap">
        {skills?.map((addSkill, ind) => (
            <li
                className="bg-indigo-100 rounded p-2 flex gap-2 items-center"
                key={ind}
            >
                {addSkill}
                <MdDeleteForever
                    color="#ff8977"
                    onClick={() => removeSkills(addSkill)}
                    size={20}
                />
            </li>
        ))}
    </ul>
)}

<button
    className="btn btn-sm w-fit btn-primary"
    type="button"
    onClick={updateSkills}

```

```

>
  Save
</button>
</div>
<div className="divider my-2"></div>
<div className="flex justify-between gap-2 flex-col">
  <h4 className="text-xl">Resume/Portfolio</h4>
  <div className="flex gap-5">
    <input
      className="input input-bordered w-full input-primary max-w-xl input-
sm"
      type="text"
      placeholder="paste the link"
    />
    <button className="btn btn-primary btn-sm">update</button>
  </div>
</div>
<div className="divider my-2"></div>
<div className="flex gap-2 flex-col">
  <h3 className="text-xl">Socials</h3>
  <div className="flex flex-col gap-2">
    <div className="flex gap-5 items-center">
      <GoMarkGithub size={20} />
      <input
        type="text"
        placeholder="paste the link"

```



```

        className="border-2 border-gray-300 rounded-md px-3 my-1 max-
w-md"

        />
    </div>
    <div className="flex gap-5 items-center">
        <BsLinkedin size={20} />
        <input
            type="text"
            placeholder="paste the link"
            className="border-2 border-gray-300 rounded-md px-3 my-1 max-
w-md"

            />
    </div>
    <div className="flex gap-5 items-center">
        <TfiTwitterAlt size={20} />
        <input
            type="text"
            placeholder="paste the link"
            className="border-2 border-gray-300 rounded-md px-3 my-1 max-
w-md"

            />
    </div>
    <button className="btn btn-primary btn-sm max-w-fit">
        save
    </button>
</div>
</div>

```

```

        </div>
    </div>
</div>
</>
);
};

```

export default Profile;

#### **7.2.1.4 UTILS**

##### **HELPER.JS**

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

```

let api_keys = [];
api_keys[0] = {
    id: import.meta.env.VITE_ADZUNA_API_ID_1,
    key: import.meta.env.VITE_ADZUNA_API_KEY_1,
};
api_keys[1] = {
    id: import.meta.env.VITE_ADZUNA_API_ID_2,
    key: import.meta.env.VITE_ADZUNA_API_KEY_2,
};
api_keys[2] = {
    id: import.meta.env.VITE_ADZUNA_API_ID_3,
    key: import.meta.env.VITE_ADZUNA_API_KEY_3,
};

```

```
export const urlRegex =
```

```

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

```

```

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

```

```

const getRandomNumber = () => {
  const randomnumber = Math.floor(Math.random() * 3);
  return randomnumber;
};

```

```

export const getBaseUrl = (query) => {
  const rand = getRandomNumber();
  return
`http://api.adzuna.com/v1/api/jobs/in/search/1?app_id=${api_keys[rand].id}&app_key=${api_keys[rand].key}&results_per_page=15&what=${query}&content-type=application/json`;
};

```

```

export const getBaseUrl_with_skills = (query, skills) => {
  const rand = getRandomNumber();
  return `http://api.adzuna.com/v1/api/jobs/in/search/1?app_id=${
    api_keys[rand].id
  }&app_key=${
    api_keys[rand].key
  }&results_per_page=15&what=${query}&what_and=${skills.join(
    " "
  )}`;
};

```

```
)}&&content-type=application/json`;
```

```
};
```

## **APP**

### **APP.JSX**

```
import { useEffect } from "react";
```

```
import { HashRouter, Route, Routes } from "react-router-dom";
```

```
import Navbar from "../components/Navbar";
```

```
import { AppProvider } from "../context/AppContext";
```

```
import Auth from "../screens/Auth";
```

```
import Dashboard from "../screens/Dashboard";
```

```
import Profile from "../screens/Profile";
```

```
function App() {
```

```
  useEffect(() => {
```

```
    window.watsonAssistantChatOptions = {
```

```
      integrationID: import.meta.env.VITE_WATSON_INTEGRATION_ID, // The  
ID of this integration.
```

```
      region: import.meta.env.VITE_WATSON_REGION, // The region your  
integration is hosted in.
```

```
      serviceInstanceID:
```

```
import.meta.env.VITE_WATSON_SERVICE_INSTANCE_ID, // The ID of your  
service instance.
```

```
      onLoad: function (instance) {
```

```
        instance.render();
```

```
      },
```

```
    };
```

```
    setTimeout(function () {
```

```

const t = document.createElement("script");
t.src =
  "https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
  (window.watsonAssistantChatOptions.clientVersion || "latest") +
  "/WatsonAssistantChatEntry.js";
document.head.appendChild(t);
});
}, []);
return (
  <HashRouter>
    <AppProvider>
      <Navbar />
      <Routes>
        <Route path="/" element={<Auth />} />
        <Route path="/dashboard" element={<Dashboard />} />
        <Route path="/profile" element={<Profile />} />
      </Routes>
    </AppProvider>
  </HashRouter>
);
}

```

export default App;

## INDEX

## INDEX.CSS

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

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

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

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

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

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

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

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

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

## **MAIN**

### **MAIN.JSX**

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

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

```
    </ChakraProvider>
  </React.StrictMode>
);
```

### **7.2.2 DOCKER FILE**

```
# 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"]
```



### 7.2.3 INDEX FILE

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <link rel="icon" type="image/svg+xml" href="cv.png" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Job Search</title>
</head>

<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>

</html>
```

### 7.2.4 MAIN PYTHON FILE

```
from backend import create_app
import os

app = create_app()

port = os.environ.get("PORT", 5000)
```

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

### **7.2.5 POSTCSS.CONFIG.CJS**

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

### **7.2.6 TAILWIND.CONFIG.CJS**

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

### **7.2.7 CONFIGURATION**

#### **VITE.CONFIG.JS**

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

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

### 7.2.8 PACKAGE.JSON

```
{
  "name": "react-flask-app",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "start": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "server": "cd backend && flask --debug run"
  },
  "dependencies": {
    "axios": "^1.1.3",
    "daisyui": "^2.33.0",
```

```
"react": "^18.2.0",
"react-dom": "^18.2.0",
"react-icons": "^4.6.0",
"react-router-dom": "^6.4.2"
},
"devDependencies": {
  "@types/react": "^18.0.17",
  "@types/react-dom": "^18.0.6",
  "@vitejs/plugin-react": "^2.1.0",
  "autoprefixer": "^10.4.12",
  "postcss": "^8.4.18",
  "tailwindcss": "^3.1.8",
  "vite": "^3.1.0"
}
}
```

## CHAPTER 8

### TESTING

#### 8.1 TEST CASES

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation (Y/N)	BUG ID	Executed By
1	LoginPage_TC_001	Functional	Login page	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		Users should be navigated to registration page	Working as expected	Pass	Excellent	N		MUKESH
2	LoginPage_TC_002	UI	Home Page	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	email: c@gmail.com password: 24122001	Users should be notified of login process	Not working	pass	Excellent	N		SIVAKUMAR
3	LoginPage_TC_003	Functional	Home page	Verify user is able to log into application with Valid credentials	sign up done	1. Open the website 2. Enter details and press login 3. Verify that users are logged into website properly	email: c@gmail.com password: 24122001	User should be logged into website properly	Working as expected	Pass	Good	N		GNANASHANKAR
4	HomePage_TC_001	Functional	Home Page	Verify that categories of skills and jobs are shown in homepage	Updated skills in dashboard	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	Pass	Good	N		KIRITHEESWARAN
5	HomePage_TC_002	Functional	Home page	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		MUKESH
6	HomePage_TC_003	Functional	Home page	Verify that when clicked on jobs it is redirected to correct page	API limit	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		SIVAKUMAR

## 8.2 USER ACCEPTANCE TESTING

### 1. PURPOSE OF DOCUMENT

The purpose of this document is to briefly explain the test coverage and open issues of the Skills and Job Recommendation project at the time of the release to User Acceptance Testing(UAT).

### 2. 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
ByDesign	5	0	0	0	5
Duplicate	1	1	0	1	3
External	2	2	0	1	5
Fixed	8	3	0	2	13
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	8	6	1	2	14

### 3. TEST CASE ANALYSIS

This reports hows the number of testcases that have passed, failed, and untested.

Section	TotalCases	NotTested	Fail	Pas s
PrintEngine	7	0	1	6
ClientApplication	51	0	0	51
Security	4	0	2	2

Outsource Shipping	15	0	1	14
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

## USERS

The screenshot shows the IBM Db2 on Cloud web interface. On the left, there's a sidebar with 'Data objects' and 'Saved objects' tabs. Below them is a search bar 'Filter objects' and a list of objects, including 'XGF62208'. The main area displays a SQL editor with the query: `select user_id,name,phone_number from users;`. Below the editor, the 'Results' tab is active, showing 'Result set 1' with a table of 4 rows. The table has columns: USER\_ID, NAME, and PHONE\_NUMBER.

USER_ID	NAME	PHONE_NUMBER
16	kiritheeswaran	9566759843
11	test_account	9486896070
12	sivakumar	7904559310
17	gnanashankaran	9384174585

## OUTPUT

F-ing Jobs

Login

SignUp

Sign in with Github

or

email

password

LOGIN



**Your Skills**

HTML

[Include your skills](#)

 Search

## Recommended Jobs

Sr QA Bangalore, Contract to Hire.

HORIGINE CONSULTING PVT. LTD.

Hi &nbsp; Opening for&nbsp;Sr QA , Bangalore, Contract to Hire. Location : Bangalore. Yrs of experience : 8 to 14 yrs 8 years of experience Skillsets Experience with Retail / E Commerce domain is a must Manual Testing - Functional, End to End , Test data management, Test Case Authoring, Test Execution Management, API and UI based Testing Test Case Automation -

Sr. Software Engineer Java, Panchkula

Orcapod Consulting Services Pvt. Ltd.

Sr. Software Engineer Java, Panchkula Job description Skills Experience with developing highly scalable, secure, and resilient applications utilizing Java, Spring Core, Spring Boot, REST APIs, JPA, Hibernate, and Swagger Proficiency with Object-Oriented Design (OOD) and Test-Driven Development (TTD). Hands on experience with test automating frameworks such as

Post- Associate Manager / Deputy Manager

Syngene International Limited

Designation: Associate Manager/Deputy Manager Job Location: Bangalore Department: Chemical Development Process Engineering About Syngene Syngene International Ltd. (BSE: 539268, NSE: SYNGENE, ISIN: INE398R01022), is an integrated research, development and manufacturing services company serving the global pharmaceutical, biotechnology



IBM Db2 on Cloud

Data objects Saved objects

Filter objects

XGF82208

\*Untitled - 4

Syntax assistant

Run all

```
1 select * from skills;
```

History Results

Result set 1 Details

Filter table Total:3

SKILL_ID	USER_ID	NAME
44	12	HTML
53	12	CSS
54	12	cloud





## Your Profile

mukesh  
mukesh30012002@gmail.com  
9486896070



## Skills

 +

HTML

SAVE

## Resume/Portfolio

 UPDATE

## Socials




SAVE


<http://169.51.207.195:32478/#/dashboard>


## Your Skills

HTML

CSS

cloud

 Include your  
skills

## Recommended Jobs

Mulesoft Developer-  
BANGALORE

PERSONAL NETWORK Hiring  
For Client of PERSONAL  
NETWORK

About Company : An MNC which provides the technology that transforms the way we all work and live. But they are more than a technology company they are a people company. They also provide them with unparalleled growth and development opportunities & Designation :- MULESOFT & Developer & Sr. Developer & Location :-

## Shopify Developer

Gfl Recruitment Private  
Limited

Job Description Shopify Designer Developer Roles and Responsibilities Collaborating with the UX and UI design teams to produce seamless, robust, and innovative front-end user experiences Generating custom-tailored Shopify themes, and altering pre-existing templates Working closely with project managers to deploy project requirements Managing multiple projects simultaneously, and be able to address their specific

Npd Engineer Manager in  
Jaipur

DANGI DIGITAL MEDIA LLP  
Hiring For Automobile  
Manufacturing company

Hi & We are hiring New Product Development Head in Jaipur for Automobile Manufacturing company, apply here & Position - New Product Development Head Location - Right now Newai later stage Bagru, Jaipur Qualification - Diploma/BE CTC- 6 to 10 Lac P.A. Skill sets Must have worked in the company handling products for



[Explore](#)
[Repositories](#)
[Organizations](#)
[Help](#)

[Upgrade](#)
mukesh3001

Mukesh30012002
Repositories
**ibm\_project**

Using 0 of 1 private repositories. [Get more](#)

[General](#)
[Tags](#)
[Builds](#)
[Collaborators](#)
[Webhooks](#)
[Settings](#)

**Mukesh30012002/ibm\_project**

Description

Images for react-flask applications.

Last pushed: an hour ago

Docker commands

To push a new tag to this repository,

```
docker push Mukesh30012002/ibm_project:tagname
```

[Public View](#)

Tags and scans

VULNERABILITY SCANNING - DISABLED [Enable](#)

This repository contains 4 tag(s).

Tag	OS	Type	Pulled	Pushed
v5		Image	an hour ago	an hour ago
v4		Image	an hour ago	2 hours ago
v3		Image	an hour ago	2 hours ago
v1		Image	—	18 days ago

[See all](#)
[Go to Advanced Image Management](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions.

[Upgrade](#)
[Learn more](#)

Clusters /

Normal
Expires in 29 days
[Add tags](#)

[Help](#)
[Kubernetes dashboard](#)

Actions...

Overview

Worker nodes

Worker pools

DevOps New

Pool: Filter...

Search

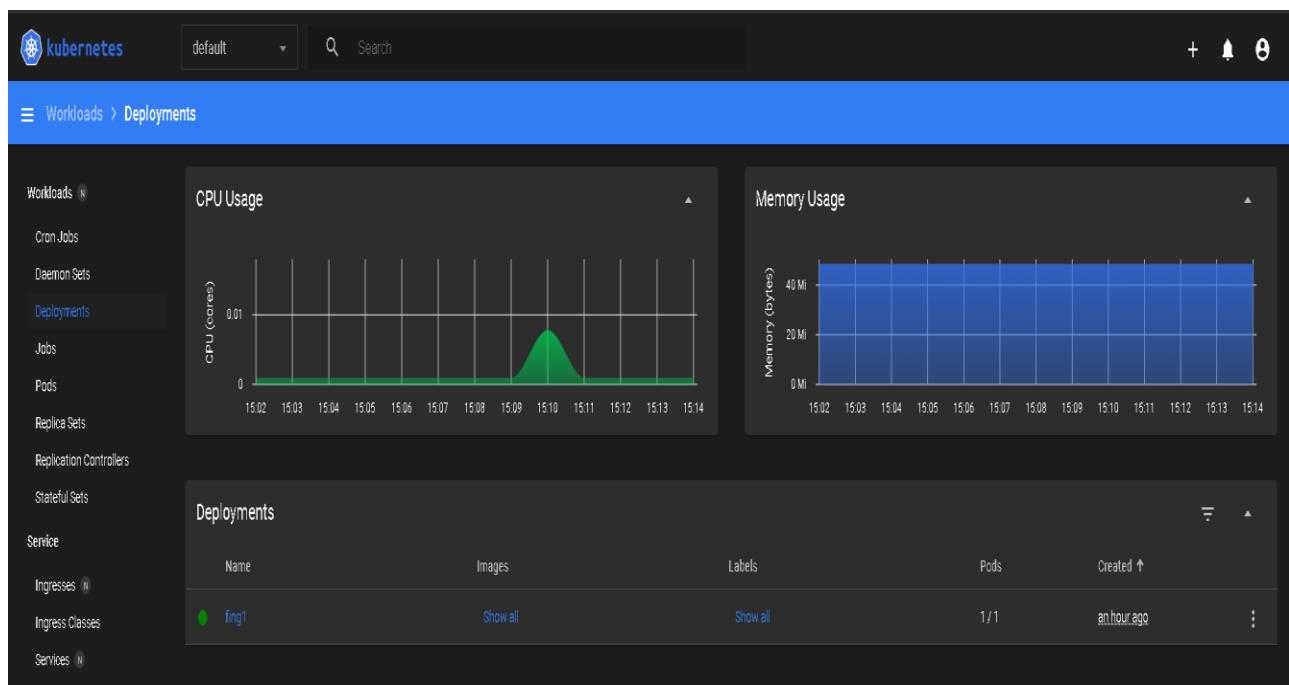
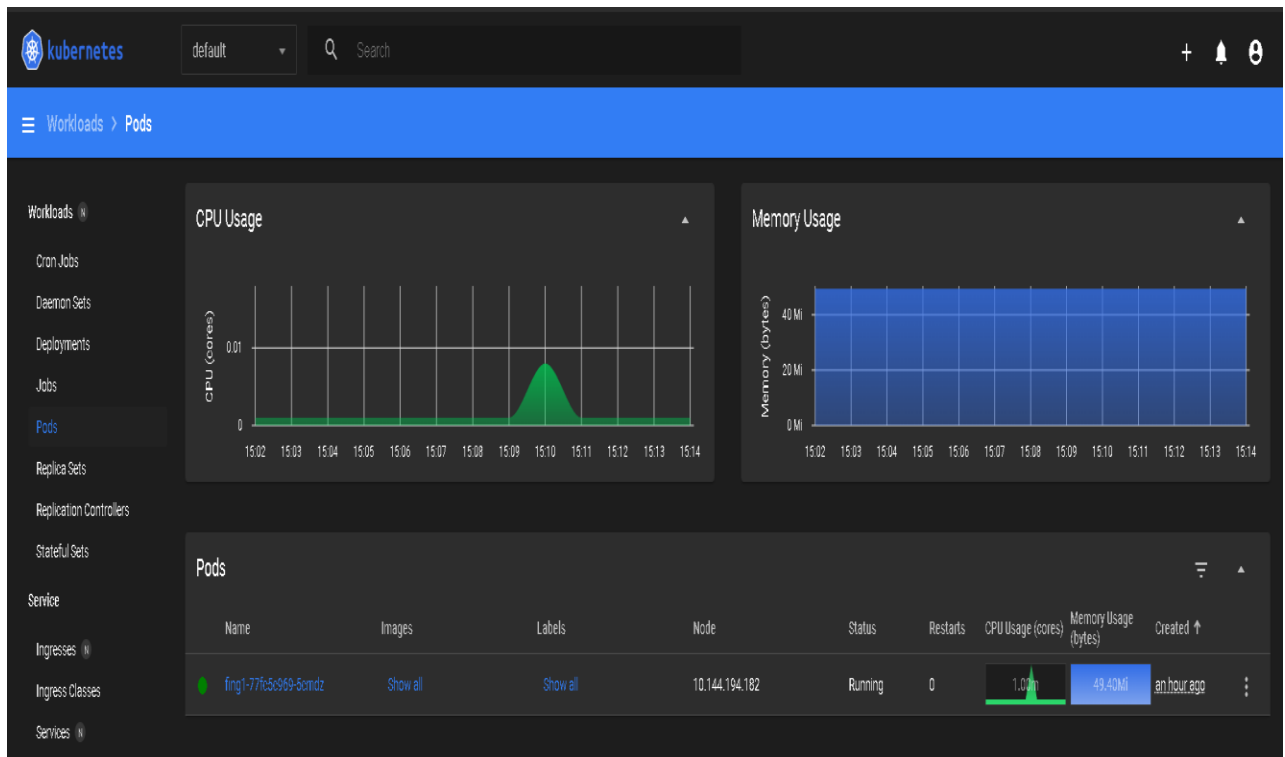
Add +

	Name	Status	Worker pool	Zone	Private IP	Public IP	Version
<div> <div></div> 0000007b </div>	Normal	default	Milan 01	10.144.194.182	169.51.207.195	1.24.7_1543	

Items per page: 25

1-1 of 1 item

1 1 of 1 page



Browser tabs: Fwd: Fir x IBM x IBM-Pro x Downlo x jira in ib x IBM-Pro x Project x Job Sear x Downlo x Downlo x

Address bar: Not secure | 169.51.207.195:32478/#/profile


### F-ing Jobs

#### Your Profile

Mukesh A

c@gmail.com

9486896070



---

#### Skills

+

cloud

html

SAVE


---

#### Resume/Portfolio

UPDATE

---

#### Socials

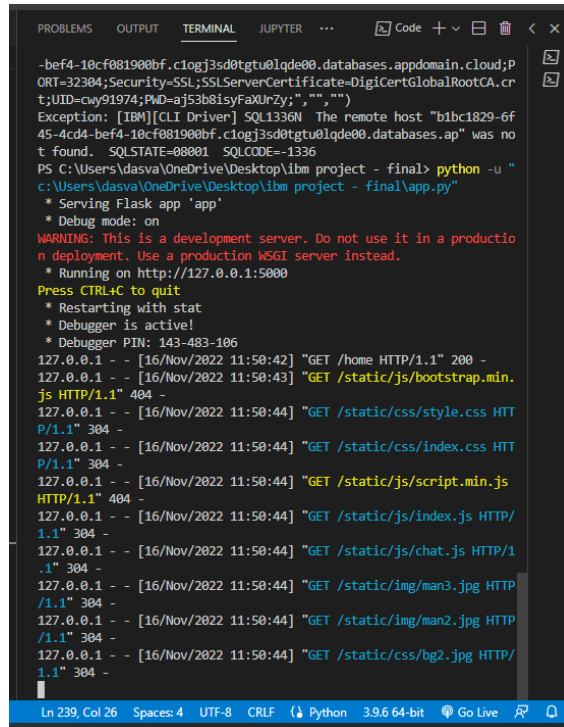


Windows taskbar: Type here to search, 28°C, 4:40 PM, 11/19/2022

# CHAPTER 9

## RESULTS

### 9.1 PERFORMANCE METRICS



```
PROBLEMS OUTPUT TERMINAL JUPYTER ... Code + - - - - -  
-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;P  
ORT=32304;Security=SSL;SSLServerCertificate=DigiCertGlobalRootCA.cr  
t;UID=cwy91974;PWD=aj53b8isyFaXUrZy;""")  
Exception: [IBM][CLI Driver] SQL1336N The remote host "b1bc1829-6f  
45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.ap" was no  
t found. SQLSTATE=08001 SQLCODE=-1336  
PS C:\Users\dasva\OneDrive\Desktop\ibm project - final> python -u "  
c:\Users\dasva\OneDrive\Desktop\ibm project - final\app.py"  
* Serving Flask app 'app'  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a productio  
n deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 143-483-106  
127.0.0.1 - - [16/Nov/2022 11:50:42] "GET /home HTTP/1.1" 200 -  
127.0.0.1 - - [16/Nov/2022 11:50:43] "GET /static/js/bootstrap.min.  
js HTTP/1.1" 404 -  
127.0.0.1 - - [16/Nov/2022 11:50:44] "GET /static/css/style.css HT  
P/1.1" 304 -  
127.0.0.1 - - [16/Nov/2022 11:50:44] "GET /static/css/index.css HT  
P/1.1" 304 -  
127.0.0.1 - - [16/Nov/2022 11:50:44] "GET /static/js/script.min.js  
HTTP/1.1" 404 -  
127.0.0.1 - - [16/Nov/2022 11:50:44] "GET /static/js/index.js HTTP/  
1.1" 304 -  
127.0.0.1 - - [16/Nov/2022 11:50:44] "GET /static/js/chat.js HTTP/1  
.1" 304 -  
127.0.0.1 - - [16/Nov/2022 11:50:44] "GET /static/img/man3.jpg HTTP  
/1.1" 304 -  
127.0.0.1 - - [16/Nov/2022 11:50:44] "GET /static/img/man2.jpg HTTP  
/1.1" 304 -  
127.0.0.1 - - [16/Nov/2022 11:50:44] "GET /static/css/bg2.jpg HTTP/  
1.1" 304 -  
Ln 239, Col 26 Spaces: 4 UTF-8 CRLF Python 3.9.6 64-bit Go Live
```

## CHAPTER 10

### ADVANTAGES & DISADVANTAGES

#### ADVANTAGES

- When recruiting externally, hiring teams find candidates, evaluate them and, if all goes well, persuade them to join their company. All of which takes time.
- Everyone needs some time to adjust to a new role, but internal hires are quicker to onboard than external hires.
- May be familiar with people in their new team, especially in smaller businesses.
- Know how your company operates and most of your policies and practices.

#### DISADVANTAGES

- Employees who were considered for a role could feel resentful if a colleague or external candidate is eventually hired.
- While your company may have a lot of qualified candidates for specific positions, this isn't necessarily true for every open role.

#### ADVANTAGE COMPARISON

	EXISTING SYSTEM	PROPOSED SYSTEM
Detection type	needs recursive calculations	use single neural network
Recall	91.32%	96%
Accuracy	91%	97.75%

## **CHAPTER 11**

### **CONCLUSION**

By the end of this project we will

- know fundamental concepts and techniques of recommender system.
- gain a broad understanding of databases and cloud.
- know how to build a web application using the Flask framework.
- know how to build chatbot.
- know how to containerize the application.

## **CHAPTER 12**

### **FUTURE SCOPE**

- AI is revolutionizing the recommender systems.
- The popularity of LinkedIn and Google for jobs has proved that there is a future for job boards if effectively managed to provide solutions
- Right pricing strategies for online recruitment advertising are essential to get an effective response.
- Recruiters and job seekers are experiencing an entirely automated process of searching and connecting. All job boards should be perfectly indexed, highly responsive, and exhaustive in job descriptions to establish their credibility and reliability.

## CHAPTER 13

## APPENDIX

**GITHUB :** [IBM-EPBL/IBM-Project-32092-1660207996](#)



