Skill / Job Recommender Application

Team ID: PNT2022TMID34860

JOB RECOMMENDATION APPLICATION

NALAIYA THIRAN PROJECT BASED LEARNING ON

PROFESSIONAL READLINESS FOR INNOVATION,

EMPLOYNMENT AND ENTERPRENEURSHIP

A PROJECT REPORT

PRIYA DHARSHINI B (962819104066)

REESHMA SHAMA M (962819104072)

SOBHA S (962819104078)

UPASANA B (962819104084)

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE

AND ENGINEERING

University College of Engineering

NAGERCOIL – 629 504

Skill / Job Recommender Application

Team ID: PNT2022TMID34860

INDEX

1. INTRODUCTION :

1.1 Project Overview

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

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

1.2 Purpose

The job recommendation application helps job seekers to find jobs. It supports discovering a job that is suitable for the user's interest. A personalized service is provided to the user that can help them find ideal jobs quickly and conveniently. The users may forget about the deadline to fill in the application So, this platform alerts the users through mail. The users can search for jobs based on their preference of location. They can connect with their friends and share the opportunities they get. The students can checkout internship options available. The users get job recommendations based on their skill, knowledge, and qualification.

2.LITERATURE SURVEY

2.1.Existing problem :

Nowadays finding and getting a job is more difficult So, job recommending application is the most important one. This is the platform which helps job seekers to find a job related to their interests, qualifications, skills and also helps recruiters to look for candidates whose qualities match with their needs. Job recommendation app provides the opportunities to know more about the available jobs. To reduce laborious work, we design and implement a recommendation system for online job-hunting. It plays a significant role in connecting employees and employers.

2.2.References

[1]. TECHNICALJOB RECOMMENDATION SYSTEM USING APIs AND WEB CRAWLING

Naresh Kumar, Maish Gupta, Deepak Sharma, and Isaac Ofari,2022.

There has been a sudden boom in the technical industry and an increase in the number of good start-ups. Keeping track of various appropriate job openings in top industry names has become increasingly troublesome. This leads to deadlines and hence important opportunities being missed. Through this research paper, the aim is to automate this process to eliminate this problem. To achieve this, Puppeteer and Representational State Transfer (REST) APIs for web crawling have been used. A hybrid system of Content-Based Filtering and Collaborative Filtering is implemented to recommend these jobs. The intention is to aggregate and recommend appropriate jobs to job seekers, especially in the engineering domain. The entire process of accessing numerous company

websites hoping to find a relevant job opening listed on their career portals is simplified. The proposed recommendation system is tested on an array of test cases, with a fully functioning user interface in the form of a web application. It has shown satisfactory results, outperforming the existing systems. It thus testifies to the agenda of quality over quantity.

[2].ENHANCED DSSM (DEEP SEMANTIC STRUCTURE MODELLING) TECHNIQUE FOR JOB RECOMMENDATION

Ravita Mishra, Sheetal Rathi Thakur College of Engg. and Technology, Computer Science and Engg. Department, Shyamnarayan Thakur Marg, Takur Village, Kandivali East, Mumbai, Maharashtra 400101, India,2021.

Now a day's recommendation system takes care of the issue of the massive amount of information overload problem and it provides the services to the candidates to concentrate on relevant information on job domain only. The job recommender system plays an important role in the recruitment process of fresher as well as experienced today. Existing job recommender system mainly focuses on content-based filtering to extricate profile content and on collaborative filtering to capture the behavior of the user in the form of rating. The dynamic nature of the job market leads to cold start and scalability issues. This problem can be addressed by item-based collaborative filtering with a machine learning technique, it learns job embedding vectors and finds similar jobs content-wise. The existing model in the job recommender domain uses the confining model to address the cold start and scalability issue and provide better recommendations, but they fail to accept the complex relationships between job description and candidate profile. In this paper, we are proposing a Deep Semantic Structure Algorithm that overcomes the issue of the existing system. The deep semantic structure modelling (DSSM) system uses the semantic representation of sparse data and it represents job description and skill entities in character trigram format which increases the efficacy of the system. We are comparing the results to three variations of DSSM model with two different datasets (Naukari.com and
CareerBuilder. com) and it gives satisfactory results. Experimental results shows that the DSSM Embedding model and its other variants are provides promising results in solving cold start problem in comparison with several variants of embedding model. We used Xavier initializer to initialize the model parameter and Adam optimizer to optimize the system performance.

[3]. REVIEW ON RESUME ANALYSIS AND JOB RECOMMENDATION USING AI

Prof. Sneha A. Khaire, Ms. Ruchita B. Birari, Ms. Riddhi S. Jagale, Ms. Sanskruti A. Patil, Ms. Surovika T Paul, Sandip Institute of Technology and Research Centre, 2021.

Applications of AI technology-based 60 minutes meeting have step by step attracted public attention and have become fashionable. The system developed within the same study may be explained in 2 elements, particularly from the attitude of the work person which of the work recruiter. we will essentially take the majority of input resume from the consumer company which consumer company will offer the necessity and therefore the constraints to that the resume ought to be hierarchical by our system. This study depends on resume analysis and therefore the technique adopted area unit machine learning and text-mining primarily based artificial intelligence-NLP. The system works as talent recommendation system for the businesses and job recommendation system for the work candidates.

[4]. JOB RECOMMENDATION SYSTEM

Bhavya Chawla, Naitik Kansara, Sakshie Pathak, Mr. S. B. Nikam, 2021

Recommendation Systems are omnipresent on the web nowadays. Most websites today are striving to provide quality recommendations to their customers in order to increase and retain their customers. In this paper, we present our approaches to style employment recommendation system for a career based social networking websites. We take a bottom-up approach: we start with deeply understanding and exploring the info and gradually build the smaller bits of the system. We also consider traditional approaches of advice systems like collaborative filtering and discuss its performance. Our experiments show the efficacy of our approaches.

[5]. THE MULTI AGENT SYSTEM FOR JOB RECOMMENDATION

Meilany Nonsi Tentua, Azhari Azhari, Aina Musdholifah Teknik Informatika, Fakultas Teknik, Universitas PGRI Yogyakarta, Indonesia, Department of Computer Science and Electronics, 2020.

The number of available job portals causes abundant information. Therefore, a system of recommendations is needed by job seekers to find jobs that fit their profile. Offering job vacancies on job portals are changing every time because there are always additional job search data and job opening data. A multi-agent system is a technology that can be used to handle information changes. This article proposed a recommendation system that is expected to help job seekers to get jobs in accordance with the field of science they have. The system will monitor what work is offered by online job portals. From the results of observations, the job content offered will be used as a reference so that if new content is entered, the agent will automatically provide input to the job seeker. Based on the results of the implementation of the recommendation system using a multi-agent system can provide search results that are in accordance with what is inputted by the user based on the profile they have. These search results can recommend jobs that match the job seeker’s profile.

[6]. IMPLICIT SKILLS EXTRACTION USING DOCUMENT EMBEDDING AND ITS USE IN JOB RECOMMENDATION

Akshay Gugnani, Hemant Misra, IBM Research – AI, 2020

This paper presents a job recommender system to match resumes to job descriptions (JD), both of which are nonstandard and unstructured/semi-structured in form. First, the paper proposes a combination of natural language processing (NLP) techniques for the task of skill extraction. The performance of the combined techniques on an industrial scale dataset yielded a precision and recall of 0.78 and 0.88 respectively. The paper then introduces the concept of extracting implicit skills – the skills which are not explicitly mentioned in a JD but may be implicit in the context of geography, industry, or role. To mine and infer implicit skills for a JD, we find the other JDs like this JD. This similarity match is done in the semantic space. A Doc2Vec model is trained on 1.1 million JDs covering several domains crawled from the web, and all the JDs are projected onto this semantic space. The skills absent in the JD but present in similar JDs are obtained, and the obtained skills are weighted using several techniques to obtain the set of final implicit skills. Finally, several similarity measures are explored to match the skills extracted from a candidate’s resume to explicit and implicit skills of JDs. Empirical results for matching resumes and JDs demonstrate that the proposed approach gives a mean reciprocal rank of 0.88, an improvement of 29.4% when compared to the performance of a baseline method that uses only explicit skills.

[7]. RECOMMENDATION SYSTEM FOR WORKERS & CUSTOMERS FOR INFORMAL JOBS

Manasi Purkara, Omkar Joshia , Abhishek Salapea , Ankush Patila , Varad Kulkarnia Dr. Pravin Futaneb, 2021

Recommender systems are software applications that provide or suggest items to users. These systems use filtering techniques to provide recommendations. The Purpose of this recommendation system was to provide services to small or part-time workers. It has been observed by the team that recommendation systems that are studied have not focused on small workers like electricians and carpenters but the corporate people. Recent trends in technology have made us dependent on technology too much itself. To cope up with the problems of urbanization and employment trends in this ever-changing world, a well-suited system for corporate workers as well as skilled laborers should coexist. This model is designed to help recruiters to hire employees based solely on work type and rating. A Recruiter can hire a person for a specific task or time frame dependent on what type of employee he/she is expecting. It is designed in such a way that it will help to reduce the gap between both of them leading to a hassle-free experience for an employer as well as an employee. The content-based technique is adopted as it is used to know the content of both user and item. A Manually generated dataset is used by taking the reference of the standard dataset of the formal employees present on the Kaggle website. This system is based on the Vector Space Model and TF-IDF vectorizer. A literature survey of several research papers in the same domain was conducted. The recommendation system has been implemented in the python programming language. The results obtained were quite accurate which helps to recommend jobs to workers and workers to customers in the required work field.

[8]. JOB RECOMMENDATION SYSTEM USING MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING

Jeevan Krishna, Data Analytics, Dublin Business School, 2020

The rise of digital communication and the spread of the

made an enormous impact in every industry. One such domain is the Hiring process, where a job seeker applies to a job by creating a profile on a job portal by providing all his/her work preferences. These work preferences of each user can be collected from each user and provide job recommendations based on their preference. There had been work done in this field, where researchers have implemented RecSys using the Hybrid filtering method as user data had previous interaction with item (Rafter et al., 2000). In this dissertation, we have approached the problem with the three-tier approach design.

Data acquired for our study has no previous interaction between the user data and Job listing data. With such a dataset, we have addressed the issue of cold start from both User and Job perspective. Also, recommend the top-n job to the user by analysing and measuring similarity between the user preference and explicit features of job listing using Content-based filtering, which is devised in support of natural language processing and cosine similarity. The Recommender System is then evaluated using precision, recall, and F1 score (Barrón-Cedeno et al., 2009). The top-n recommendation made to the user is presented in the third tier of the design, a web app deployed in the local server. The presentation layer web-app is developed using Plotly's dash web framework.

[9]. IMPLEMENTATION OF AN AUTOMATED JOB RECOMMENDATION SYSTEM BAES ON CANDIDATE PROFILES .

V DesaiD BahlS VibhandikI Fatma Desai, V., Bahl, D., Vibhandik, S. and Fatma, I., 2017. Int. Res. J. Eng. Technol, 4(5), pp.1018-1021.

This work is an attempt to collate the data and discover the foremost relevant candidate-job association mapping concurring to the skills, interests, and preferences of a user and to provide a possible job

opportunity as an efficient solution. Recommender framework aims to assist in searching for jobs that coordinate user preferences and it has a successful usage in a wide range of applications to deal with problems related to information overload efficiently. This work will analyze issues for building personalized recommender frameworks for candidates and work matching. An attempt has been made to formulate this study of recommendation as a supervised machine learning problem.

[10]. JOB RECOMMENDATION SYSTEM USING PROFILE MATCHING AND WEB-CRAWLING.

Deepali V Musale , Mamta K Nagpure, Kaumudini S Patil, Rukhsar F Sayyed. Computer Science & Engineering, K K Wagh College of Engineering, Nashik, India1,2017.

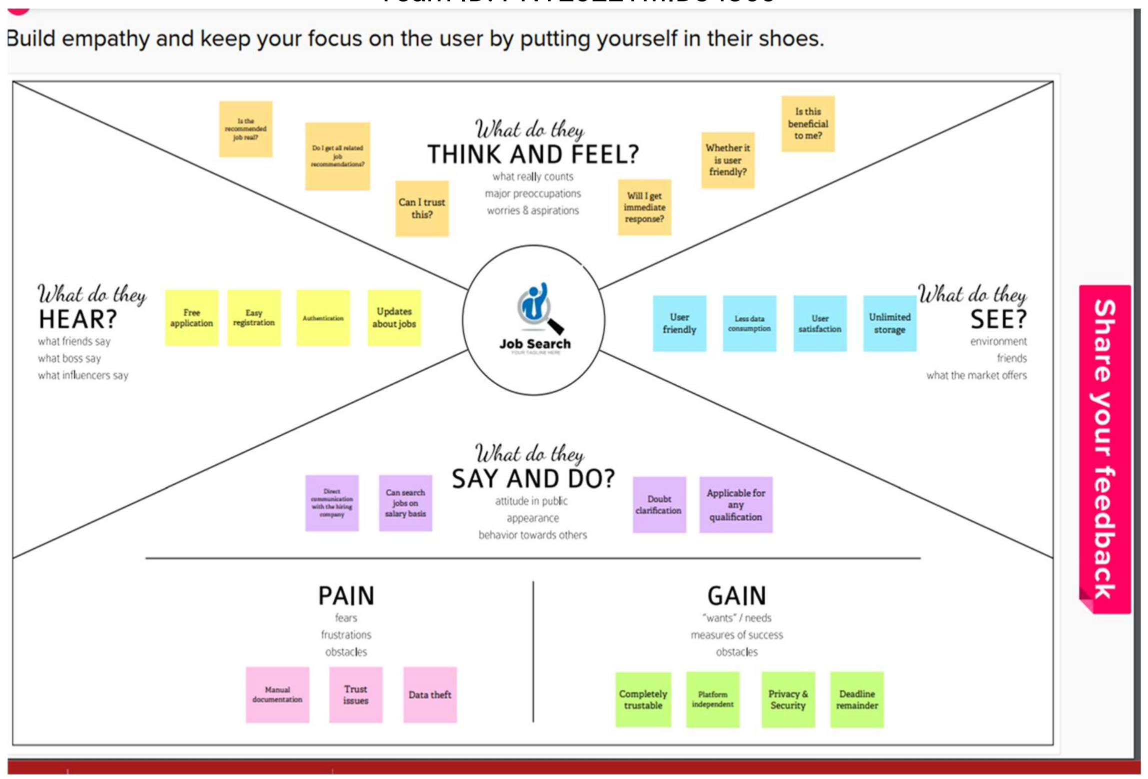
The developed system is job recommendation system for campus recruitment which helps college placement office to match company's profiles and student's profiles with higher precision and lower cost. For profile matching, two matching methods are used: semantic matching, tree-based knowledge matching and query matching. These methods are integrated according to representations of attributes of students and companies, and then the profile similarity degree is acquired. Based on profile similarity degree, preference lists of companies and students are generated. Also students can perform keyword based search for job profiles from various job recruitment sites (e.g. Naukari.com,indeed.com). For obtaining data from online recruitment sites system uses web crawling. With loop matching, matching results would be further optimized and provide more effective guidance for recommendation

2.3 Problem Statement Definition:

Nowadays the unemployment rate is increasing and finding job is getting difficult. In order to reduce difficulty in searching for jobs, a platform that recommends job for any domain will be more helpful. A job recommendation application that intimates about job offers and recommends best job related to the user's skillset will fulfill all these necessities

3. DEATION & PROPOSED SOLUTION

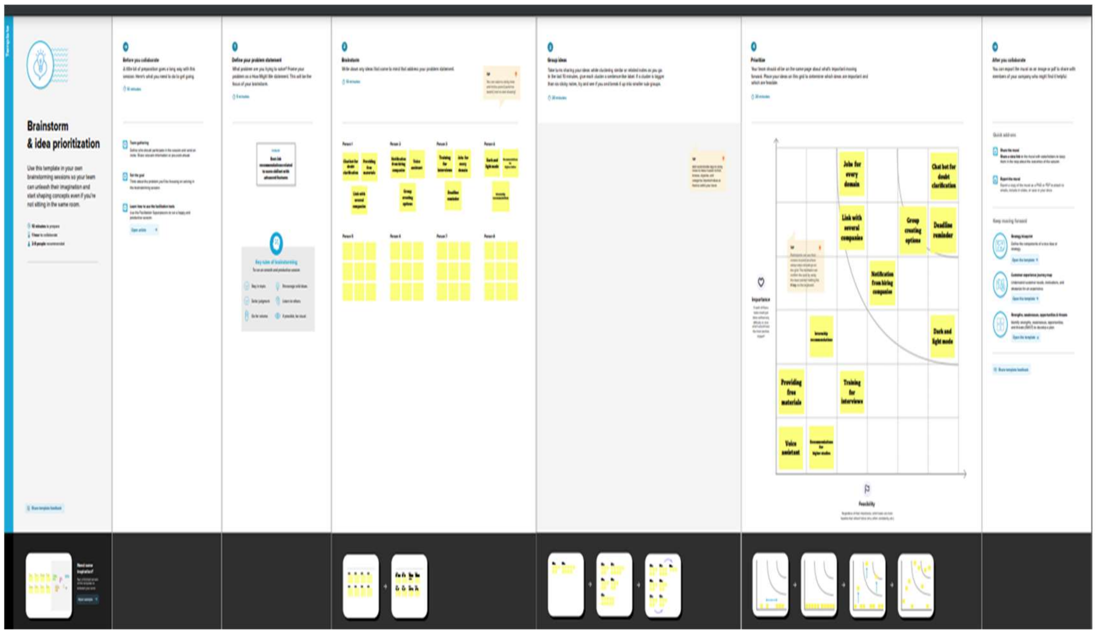
3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

Skill / Job Recommender Application

Team ID: PNT2022TMID34860

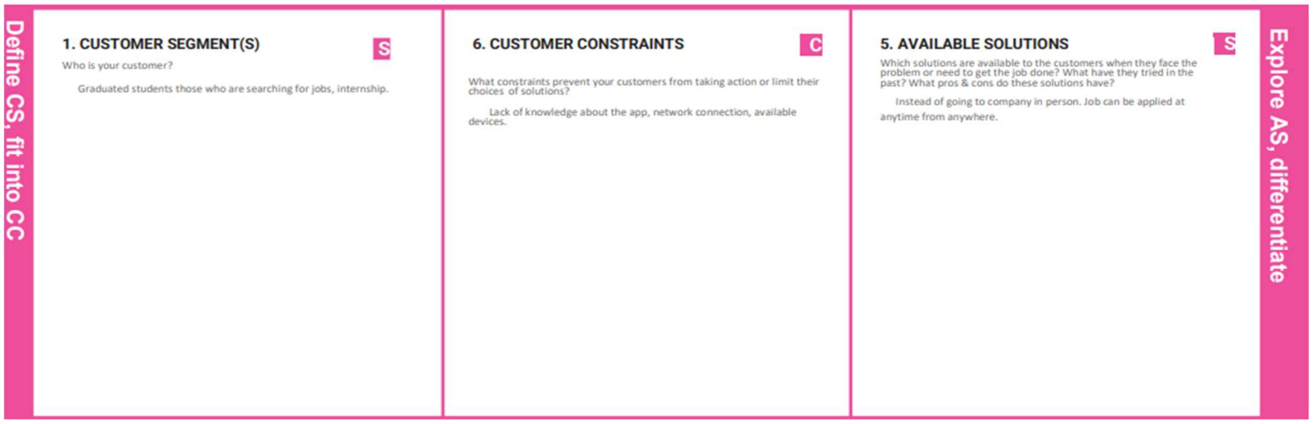


Proposed Solution

S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	Nowadays the unemployment rate is increasing and finding job is getting difficult. In order to reduce difficulty in searching for jobs a platform that recommends jobs for any domain will be more helpful. A Job recommendation application will fulfill all these necessities.
2	Idea / Solution description	Job recommendation application with job and internship recommendations related to the user's skillset and it also contains information about the company's interview process.

3	Novelty / Uniqueness	The users will not only receive recommendations for jobs, besides they will also get recommendations for internships. This application also contains details about the company's interview process
4	Social Impact / Customer Satisfaction	This reduces unemployment and it is not necessary for job seekers to visit companies for jobs. Hence, facilitating their comfortability.
5	Business Model (Revenue Model)	The application can be linked with all the organizations and can be used by the organizations to find recruiters.
6	Scalability of the Solution	Scalability of the Solution This web application uses cloud storage so it is flexible, more data can be stored and can be used in any devices.

Problem Solution fit:


Define CS, fit into CC
Explore AS, differentiate

1. CUSTOMER SEGMENT(S) S

Who is your customer?

Graduated students those who are searching for jobs, internship.

6. CUSTOMER CONSTRAINTS C

What constraints prevent your customers from taking action or limit their choices of solutions?

Lack of knowledge about the app, network connection, available devices.

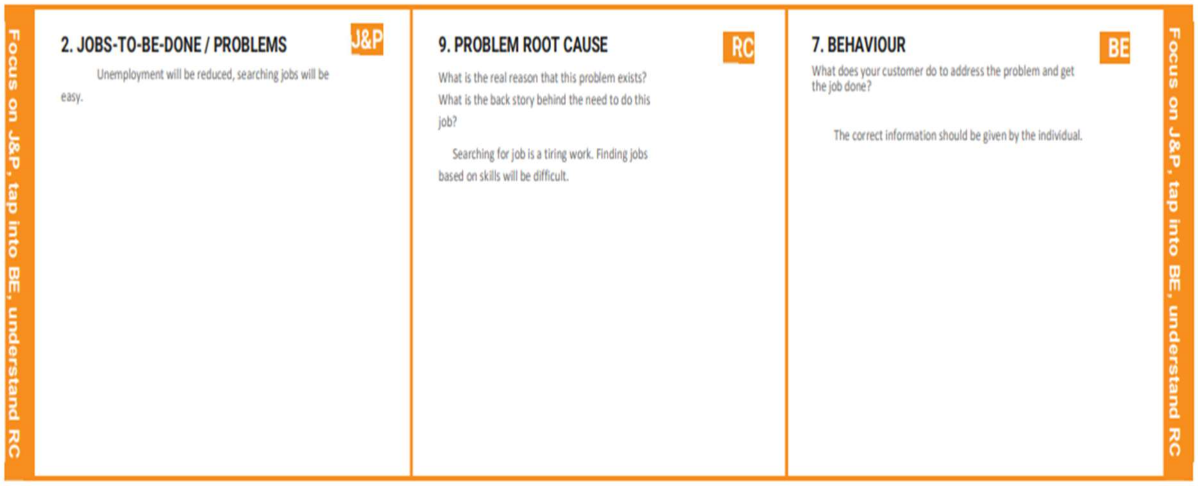
5. AVAILABLE SOLUTIONS S

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have?

Instead of going to company in person. Job can be applied at anytime from anywhere.

Skill / Job Recommender Application

Team ID: PNT2022TMID34860



Focus on J&P, tap into BE, understand RC

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

Unemployment will be reduced, searching jobs will be easy.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists? What is the back story behind the need to do this job?

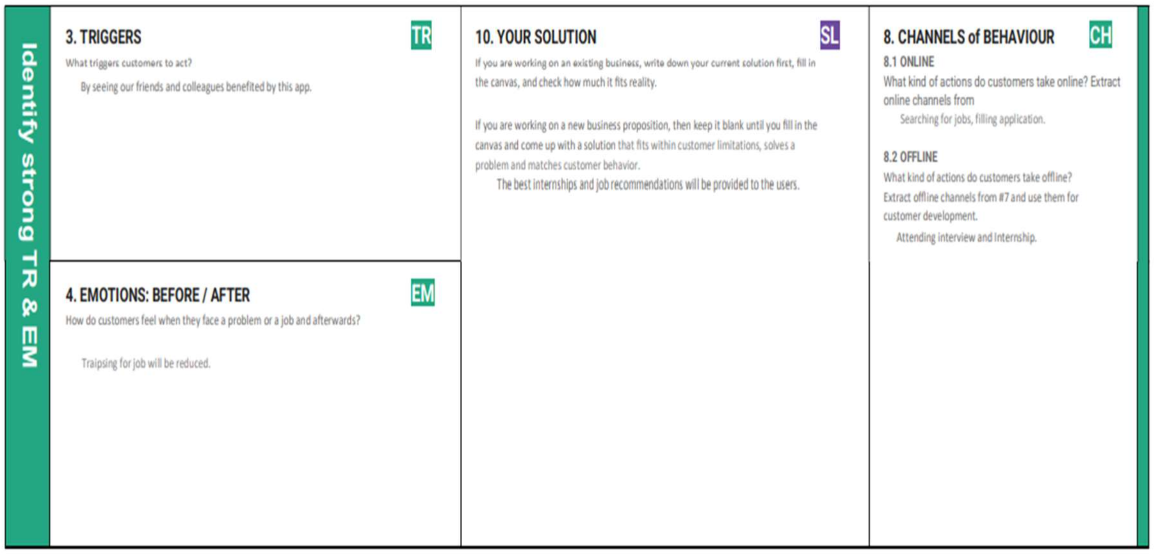
Searching for job is a tiring work. Finding jobs based on skills will be difficult.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?

The correct information should be given by the individual.

Focus on J&P, tap into BE, understand RC



Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act?

By seeing our friends and colleagues benefited by this app.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?

Traipsing for job will be reduced.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.

If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behavior.

The best internships and job recommendations will be provided to the users.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from

Searching for jobs, filling application.

8.2 OFFLINE

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

Attending interview and Internship.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	New user needs to enter their credentials to register.
FR-2	User Confirmation	Confirmation mail send to the users. If the user forgot the password, there will be an option to reset the password via email.
FR-3	Login	Only registered users can login. The account will be log out only if the user logged out the account.
FR-4	Create profile	User can upload their education qualification and skills to create the professional profile. These details help them to get the job recommendations in related fields.
FR-5	Job Search API	We can use a job search API to get the current job opening in the marke, which will fetch the data directly from the web page.
FR-6	Chatbot	Users can interact with the Chatbot. It will help them to clarify the doubts about use cases.

4.2 Non-functional Requirements:

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

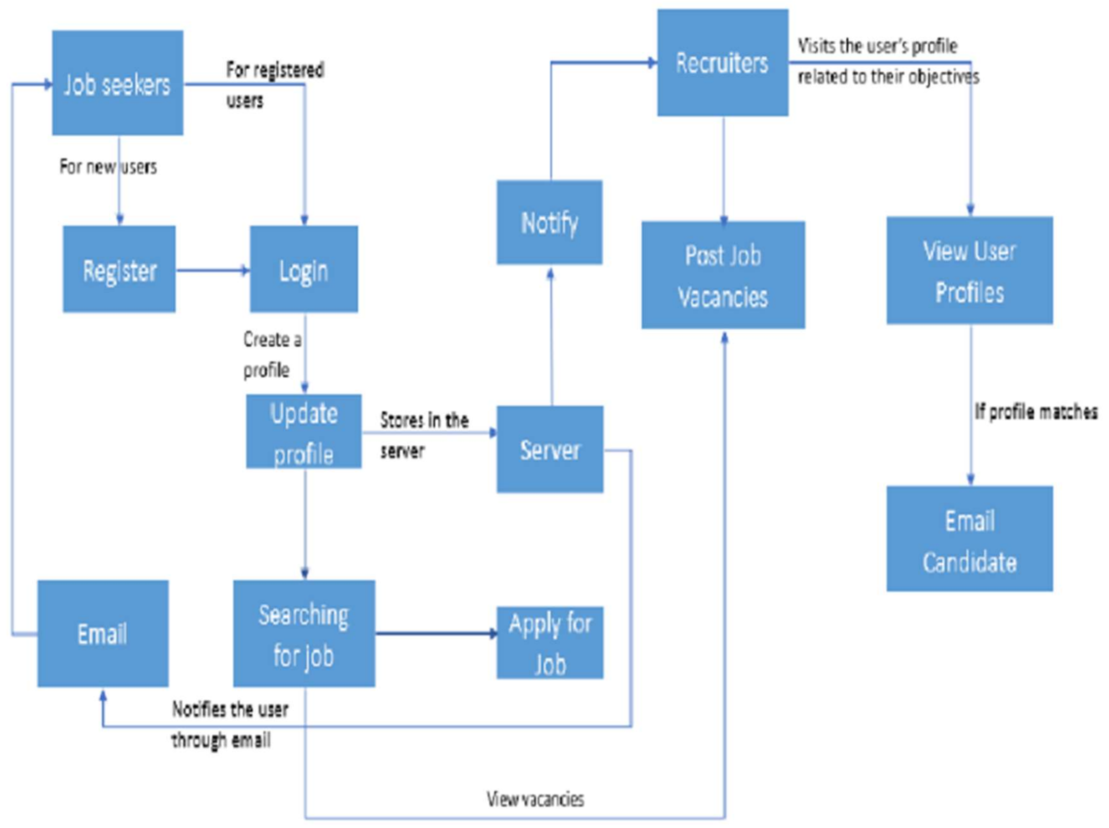
FR No.	Non-Functional Requirement	Description
FR-1	Usability	Users can easily access the web application. They can easily understand the use case.

FR-2	Security	The user who enters the correct user's name, password or security questions. After a certain number of login attempts, the security system may lock an account to protect a user's information from potential hackers.
FR-3	Reliability	Reliability specifies the system that is highly reliable functions with the same or similar efficiency after extensive use. The system must perform without failure in ninety percent of use cases during a month.
FR-4	Performance	Notifications will be delivered on time. Job and internship recommendation will be based on the user's skillset. Easy interaction with the organization members. Fast performance and less internet consumption
FR-5	Availability	The web application is platform independent. The user can search job based on their area of interest and skill. Internship is also provided.
FR-6	Scalability	Increase in users and load wouldn't affect the web application. High data storage capacity and system independent.

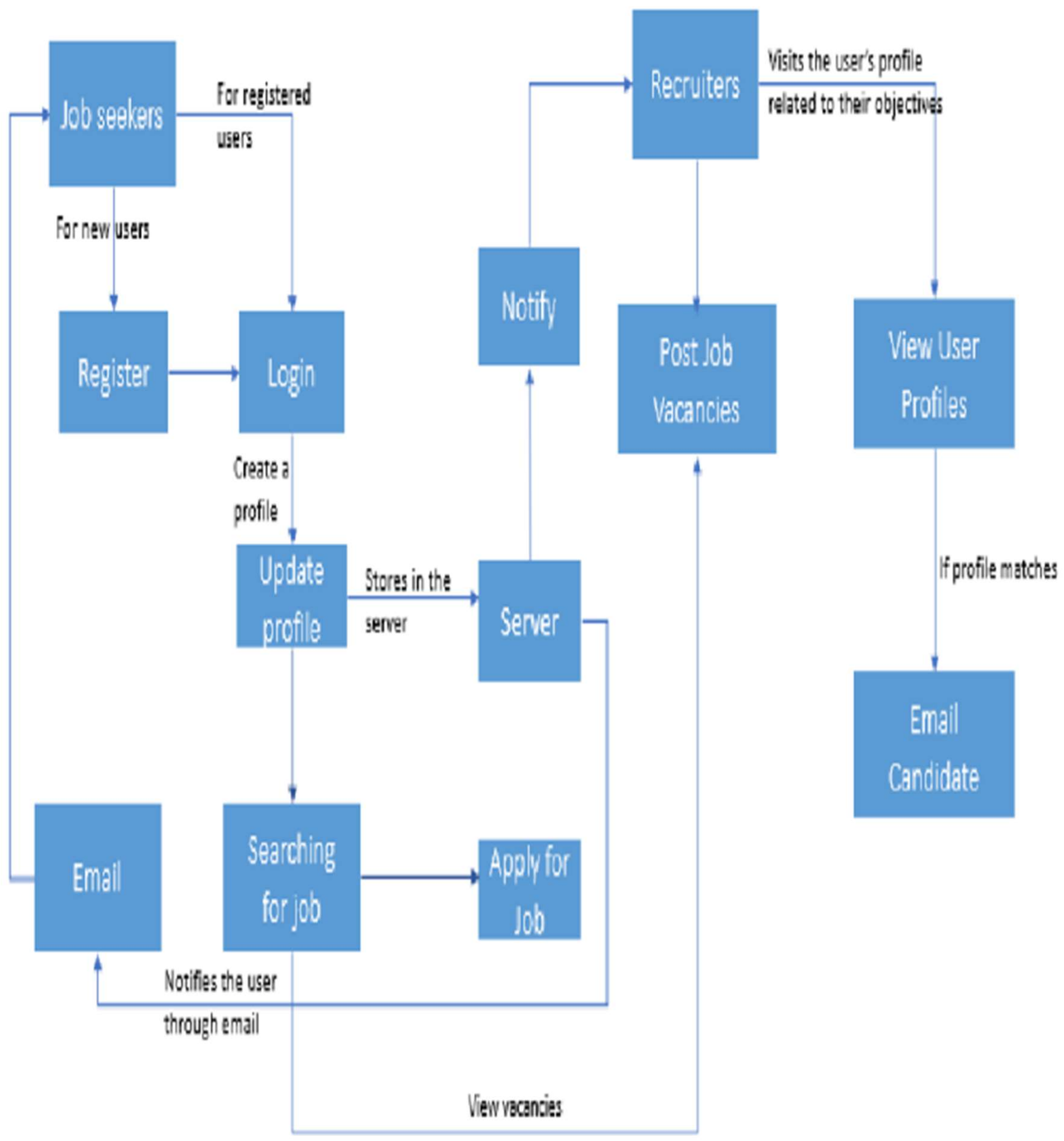
5. PROJECT DESIGN

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.



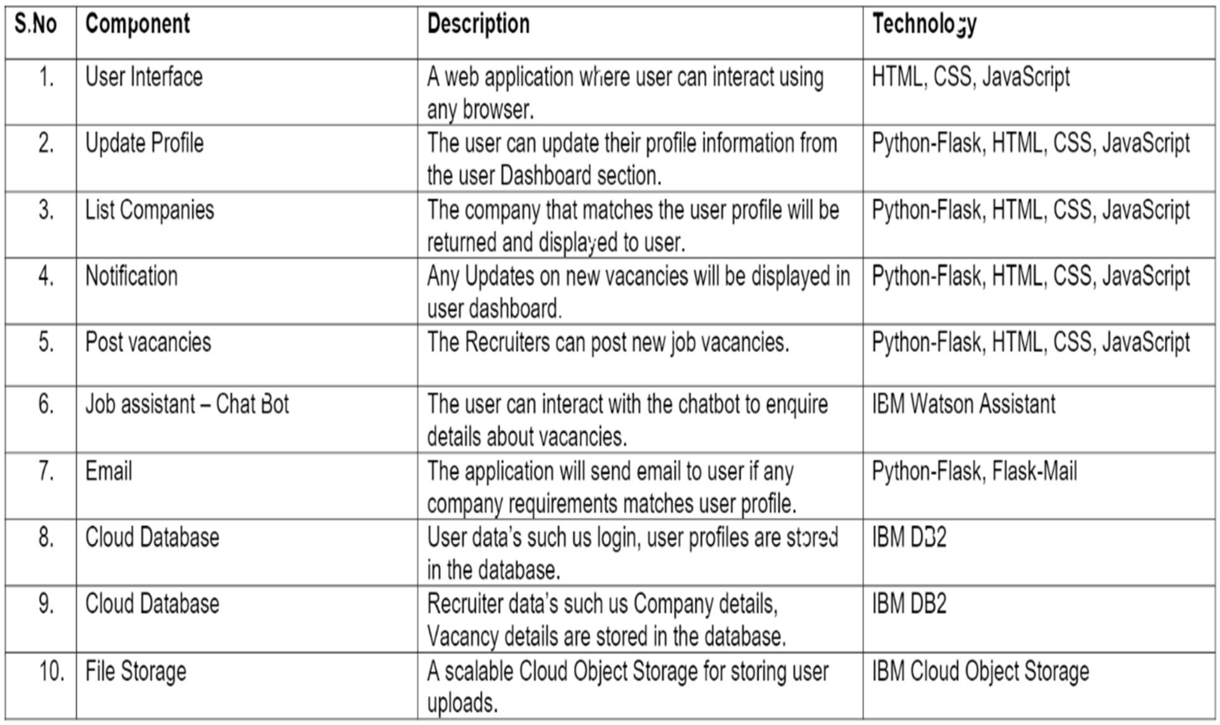
5.2 Solution & Technical Architecture



Skill / Job Recommender Application

Team ID: PNT2022TMID34860

Table-1 : Components & Technologies:



S.No	Component	Description	Technology
1.	User Interface	A web application where user can interact using any browser.	HTML, CSS, JavaScript
2.	Update Profile	The user can update their profile information from the user Dashboard section.	Python-Flask, HTML, CSS, JavaScript
3.	List Companies	The company that matches the user profile will be returned and displayed to user.	Python-Flask, HTML, CSS, JavaScript
4.	Notification	Any Updates on new vacancies will be displayed in user dashboard.	Python-Flask, HTML, CSS, JavaScript
5.	Post vacancies	The Recruiters can post new job vacancies.	Python-Flask, HTML, CSS, JavaScript
6.	Job assistant – Chat Bot	The user can interact with the chatbot to enquire details about vacancies.	IBM Watson Assistant
7.	Email	The application will send email to user if any company requirements matches user profile.	Python-Flask, Flask-Mail
8.	Cloud Database	User data's such us login, user profiles are stored in the database.	IBM DB2
9.	Cloud Database	Recruiter data's such us Company details, Vacancy details are stored in the database.	IBM DB2
10.	File Storage	A scalable Cloud Object Storage for storing user uploads.	IBM Cloud Object Storage

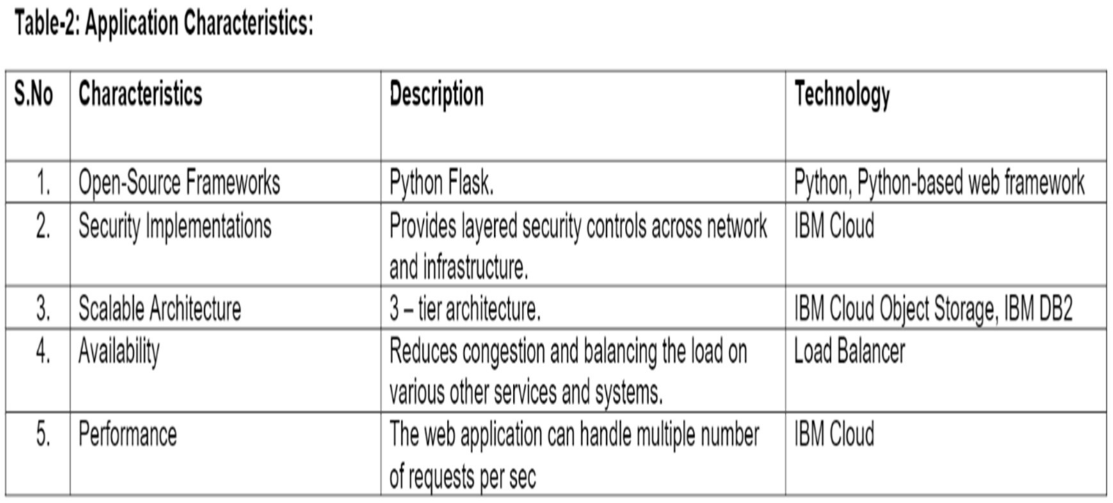
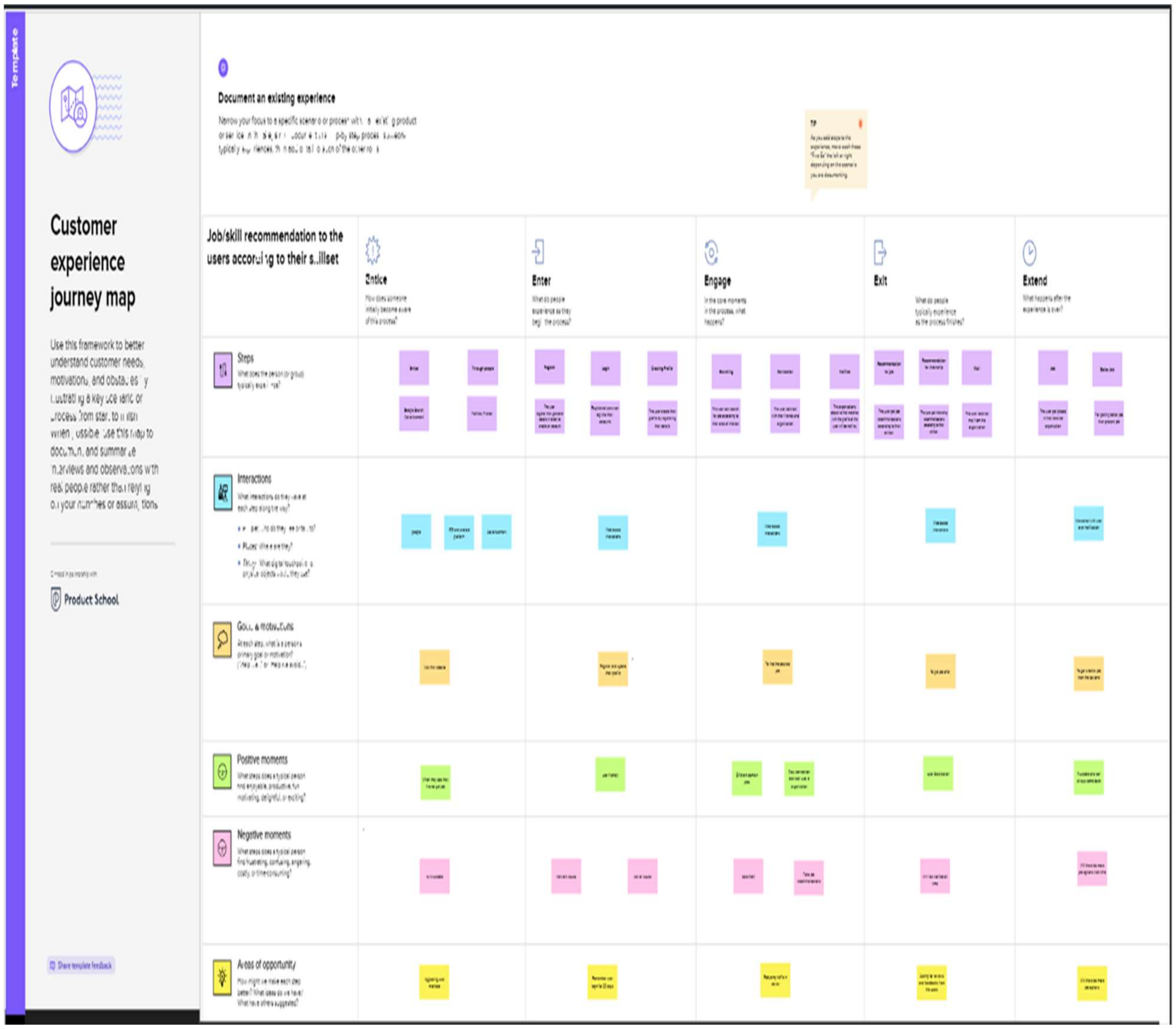


Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Python Flask.	Python, Python-based web framework
2.	Security Implementations	Provides layered security controls across network and infrastructure.	IBM Cloud
3.	Scalable Architecture	3 – tier architecture.	IBM Cloud Object Storage, IBM DB2
4.	Availability	Reduces congestion and balancing the load on various other services and systems.	Load Balancer
5.	Performance	The web application can handle multiple number of requests per sec	IBM Cloud

5.3 User Stories


Template
Customer experience journey map
Use this framework to better understand customer needs, motivations, and obstacles by illustrating a key user scenario or process from start to finish. When possible, use this map to document and summarize interviews and observations with real people rather than relying on your hunches or assumptions.
Product School
Share template feedback
Document an existing experience
Job/skill recommendation to the users according to their skillset
Entice
How does someone initially become aware of this process?
Enter
What do people experience as they begin the process?
Engage
In the core moments in the process, what happens?
Exit
What do people typically experience as the process finishes?
Extend
What happens after the experience is over?
Steps
What does the person (or group) typically experience?
Interactions
What interactions do they have at each step along the way?
Goals & motivations
At each step, what is a person's primary goal or motivation?
Positive moments
What steps does a typical person find enjoyable, productive, fun, motivating, delightful, or exciting?
Negative moments
What steps does a typical person find frustrating, confusing, angering, costly, or time-consuming?
Areas of opportunity
How might we make each step better? What ideas do we have? What have others suggested?

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

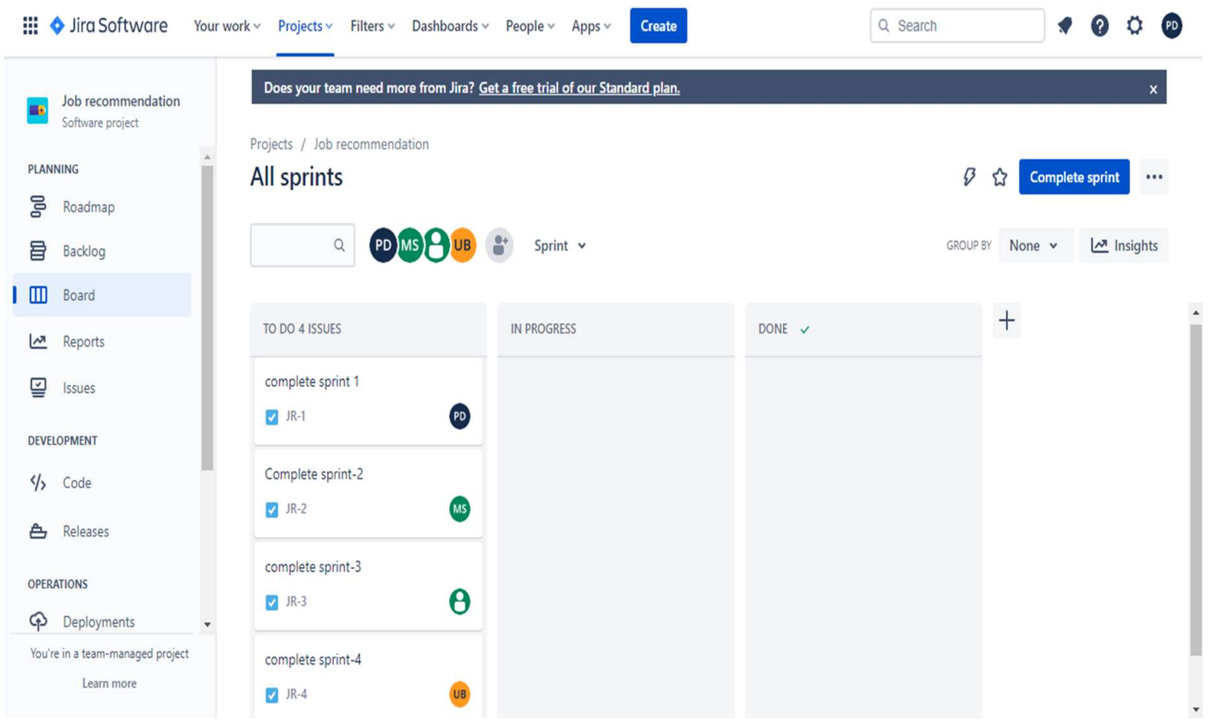
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering the required details.	2	High	Reeshma Shama
Sprint-1		USN-2	As a user, I can see a pop-up message that appears to indicate the confirmation of registration.	1	Medium	Upasana
Sprint-1	Login	USN-3	As a user, I can log into the application by entering my email & password	2	High	Sobha
Sprint-1		USN-4	As a user, after logging in once I can enter directly for 30 days	2	High	Priya Dharshini
Sprint-2	Update Profile	USN-5	As a user, I can update my profile by entering my details.	2	High	Upasana
Sprint-2	Connectivity	USN-6	As a user, I can connect with more people to expose myself.	1	Medium	Reeshma Shama
Sprint-2	Job Search	USN-7	As a user, I can search for various jobs that are related to my skill set	2	High	Sobha
Sprint-2	Apply for Job	USN-8	As a user, I can apply for jobs related to my skill set.	2	High	Priya Dharshini

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Management	USN-9	As an administrator, I can manage job seekers and recruiters.	2	High	Sobha
Sprint-3	Email	USN-10	As an administrator, I can send notifications about job vacancies through email.	1	High	Upasana
Sprint-3	Job Vacancies	USN-11	As a recruiter, I can post job vacancies available in our organization	1	High	Reeshma Shama
Sprint-4	View Profile	USN-12	As a recruiter, I can view the candidate's profile that relates to our organization's objective.	2	High	Priya Dharshini
Sprint-4	Email	USN-13	As a recruiter, I can send mail to the candidates who fit our requirements.	1	Medium	Upasana

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	27 Oct 2022	01 Nov 2022	20	06 Nov 2022
Sprint-2	20	6 Days	02 Nov 2022	07 Nov 2022		09 Nov 2022
Sprint-3	20	6 Days	08 Nov 2022	13 Nov 2022		15 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		19 Nov 2022

3. Reports from JIRA



Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / Job recommendation

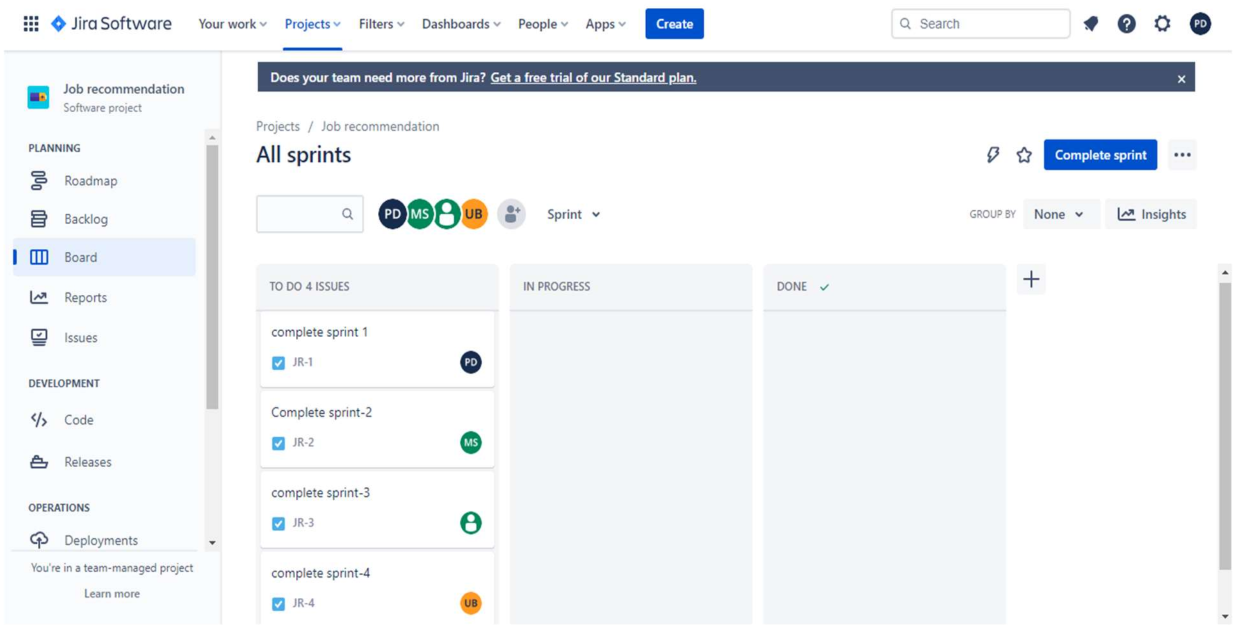
All sprints

TO DO 4 ISSUES

- complete sprint 1 — JR-1 — PD
- Complete sprint-2 — JR-2 — MS
- complete sprint-3 — JR-3
- complete sprint-4 — JR-4 — UB

IN PROGRESS

DONE ✓



7.CODING & SOLUTIONING

1. Feature 1

A **chatbot** is an artificial intelligence - powered software application that simulates human-like conversation with real humans. It is a computer program that simulates human conversation through text chats. This tool helps add convenience for customers- they are automated programs that interact with customers like a human would and cost little to nothing to engage with. The important features of the chatbot are

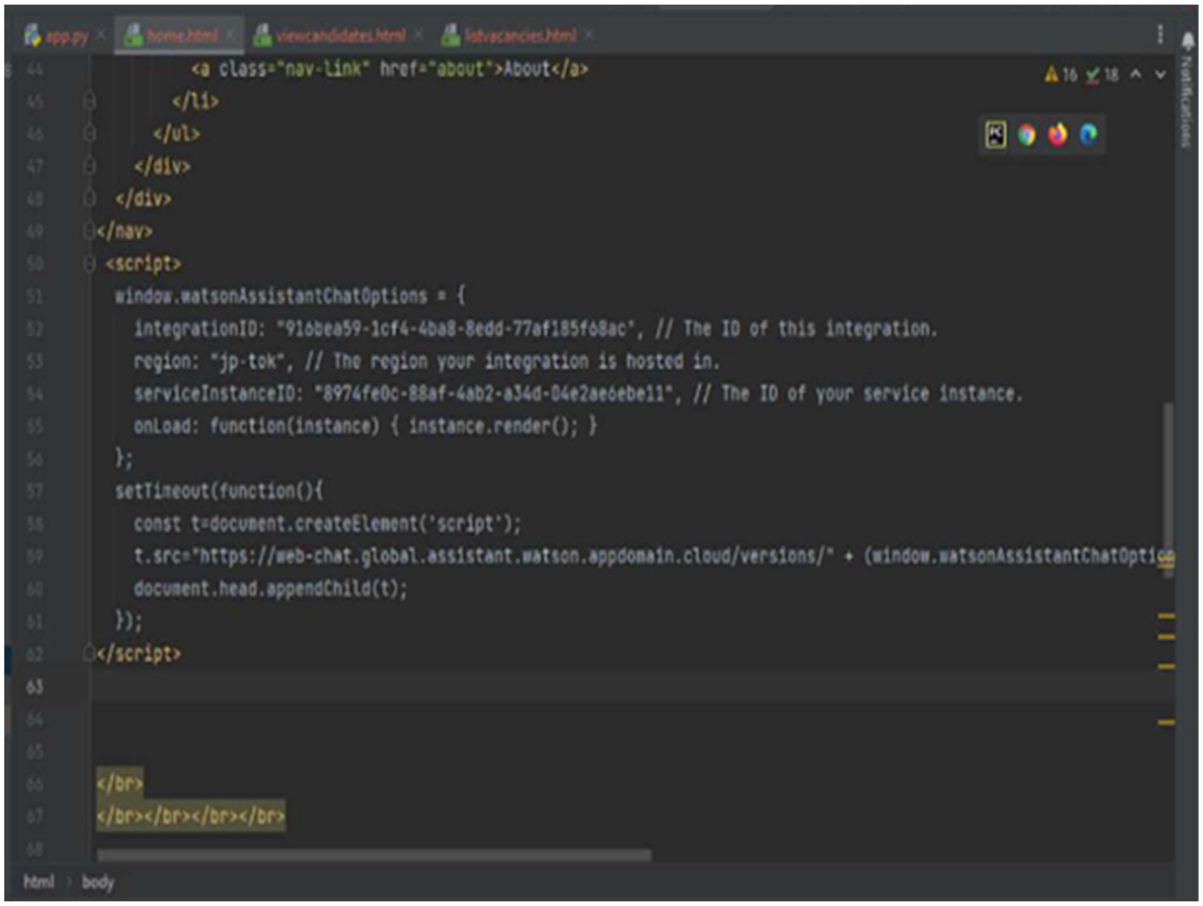
◊ Chatbots attend to customers at all times of the day and week and are not limited by time or a physical location.

◊ This makes its implementation appealing to a lot of businesses that may not have the manpower or financial resources to keep employees working around the clock.

◊ The users can ask any queries to the chat bot and it replies within an instance of time.

◊ If there is any further queries an email id will be provided to them through the chatbot they can use that email id to ask any doubt or file any issues etc.

◊ The chatbot can be used to know about the use case of the job recommendation application.

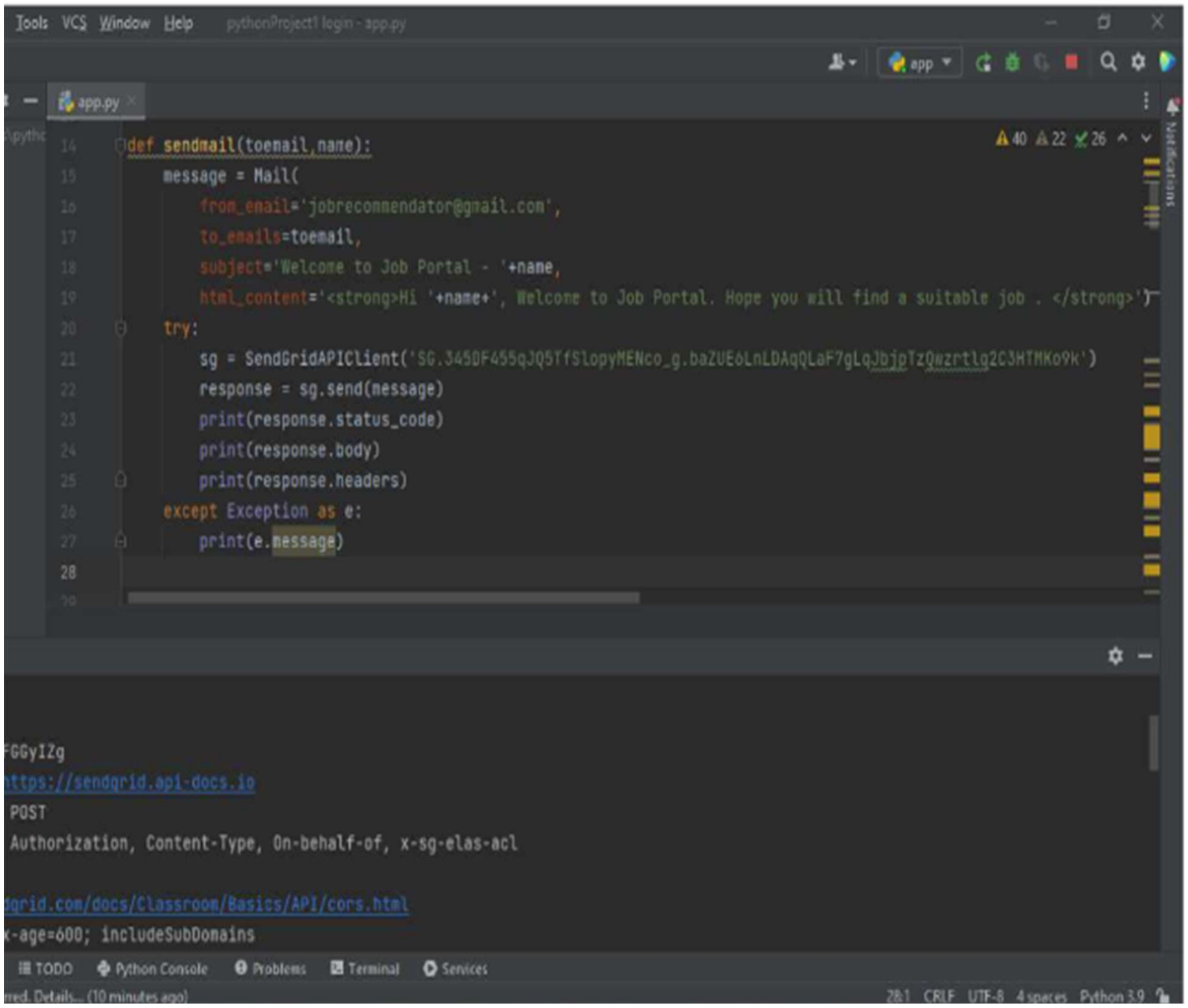


```
            <a class="nav-link" href="about">About</a>
          </li>
        </ul>
      </div>
    </div>
</nav>
 <script>
  window.watsonAssistantChatOptions = {
    integrationID: "916bea59-1cf4-4ba8-8edd-77af185f68ac", // The ID of this integration.
    region: "jp-tok", // The region your integration is hosted in.
    serviceInstanceID: "8974fe0c-88af-4ab2-a34d-04e2ae6ebe11", // 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.watsonAssistantChatOptio
    document.head.appendChild(t);
  });
</script>



</br>
</br></br></br></br>
```

2. Feature 2:

Email notifications will be sent to the users through send grid mail system. Most notifications are transactional, meaning a recipient's action or account activity triggers them. Notifications emails tend to perform well because the content is highly relevant to the recipient to know this is if you state the content clearly in the subject line.

- ◊ Once the users signup into the platform using their email id and their password.
- ◊ If the user forget their password and they want to reset it. The verification code will be sent to the registered email.
- ◊ If someone else tried to login using the user's credentials then the notification will be sent to the email.
- ◊ When there is a job vacancy that suits the user's profile then the user will be notified about that.
- ◊ The user will be notified about the deadline to fill in the application or submit the resume.
- ◊ The user will also be notified about the internship option available.



```python
def sendmail(toemail,name):
    message = Mail(
        from_email='jobrecommendator@gmail.com',
        to_emails=toemail,
        subject='Welcome to Job Portal - '+name,
        html_content='<strong>Hi '+name+', Welcome to Job Portal. Hope you will find a suitable job . </strong>')
    try:
        sg = SendGridAPIClient('SG.345DF455qJQ5TfSlopyMENco_g.baZUE6LnLDAqQLaF7gLqJbjpTzQwzrtlg2C3HTMKo9k')
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e.message)
```

```
FGGyIZg
https://sendgrid.api-docs.io
 POST
 Authorization, Content-Type, On-behalf-of, x-sg-elas-acl

dgrid.com/docs/Classroom/Basics/API/cors.html
x-age=600; includeSubDomains
```

3. Database Schema

Table 1- users

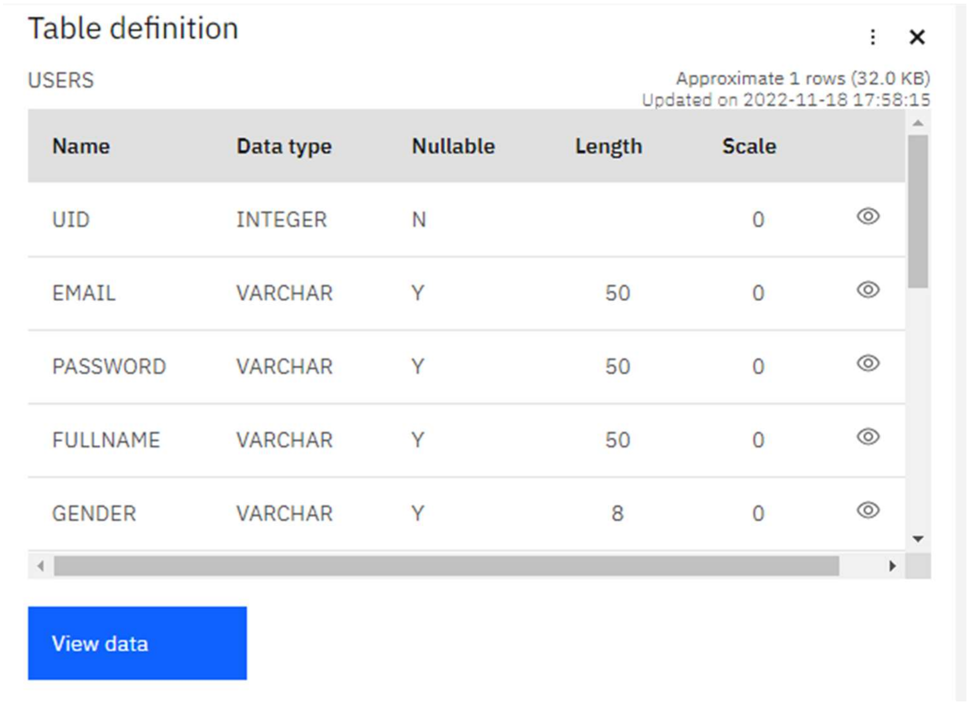


Table definition

USERS

Approximate 1 rows (32.0 KB)
Updated on 2022-11-18 17:58:15

Name	Data type	Nullable	Length	Scale
UID	INTEGER	N		0
EMAIL	VARCHAR	Y	50	0
PASSWORD	VARCHAR	Y	50	0
FULLNAME	VARCHAR	Y	50	0
GENDER	VARCHAR	Y	8	0

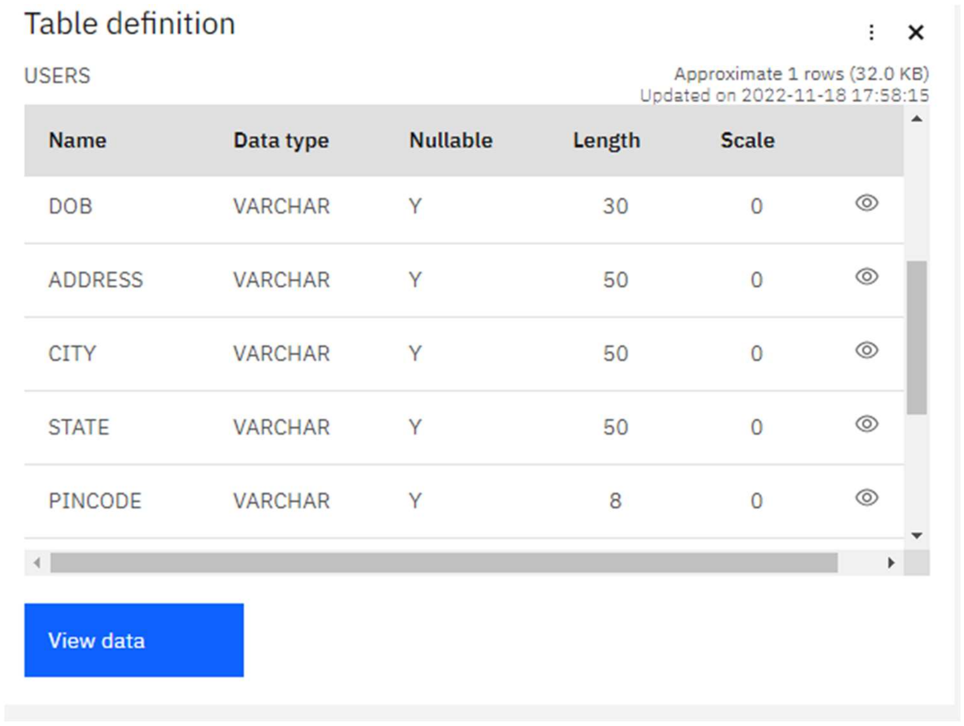


Table definition

USERS

Approximate 1 rows (32.0 KB)
Updated on 2022-11-18 17:58:15

Name	Data type	Nullable	Length	Scale
DOB	VARCHAR	Y	30	0
ADDRESS	VARCHAR	Y	50	0
CITY	VARCHAR	Y	50	0
STATE	VARCHAR	Y	50	0
PINCODE	VARCHAR	Y	8	0

Skill / Job Recommender Application

Team ID: PNT2022TMID34860

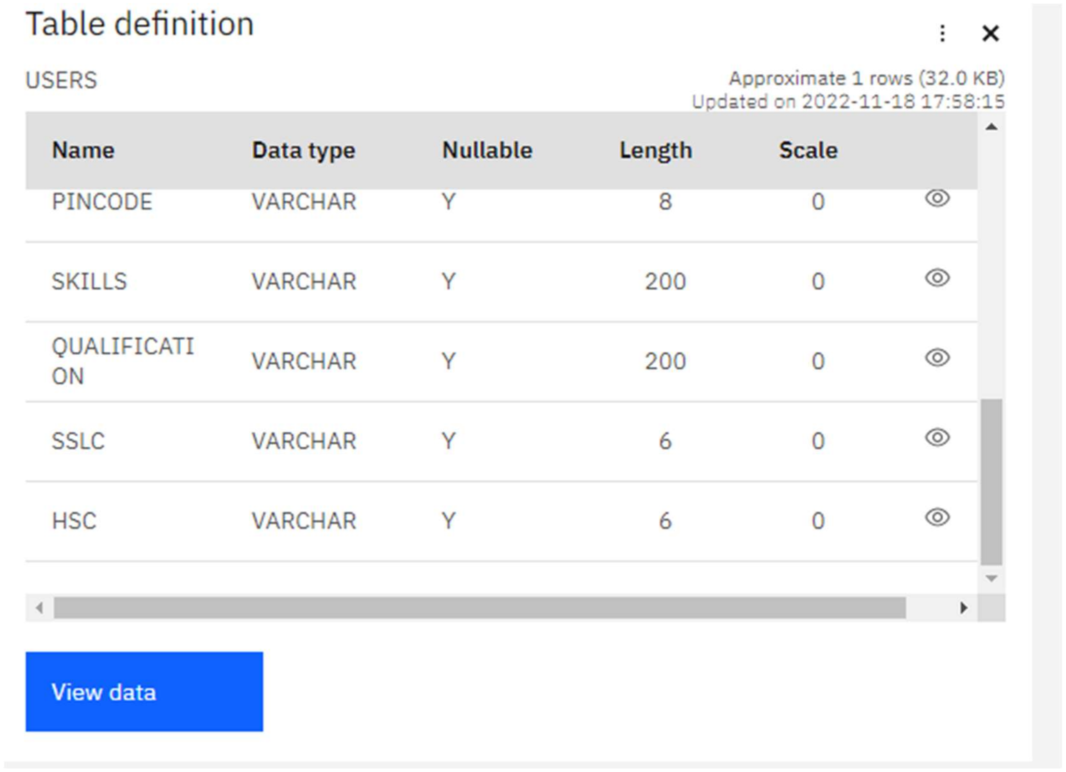


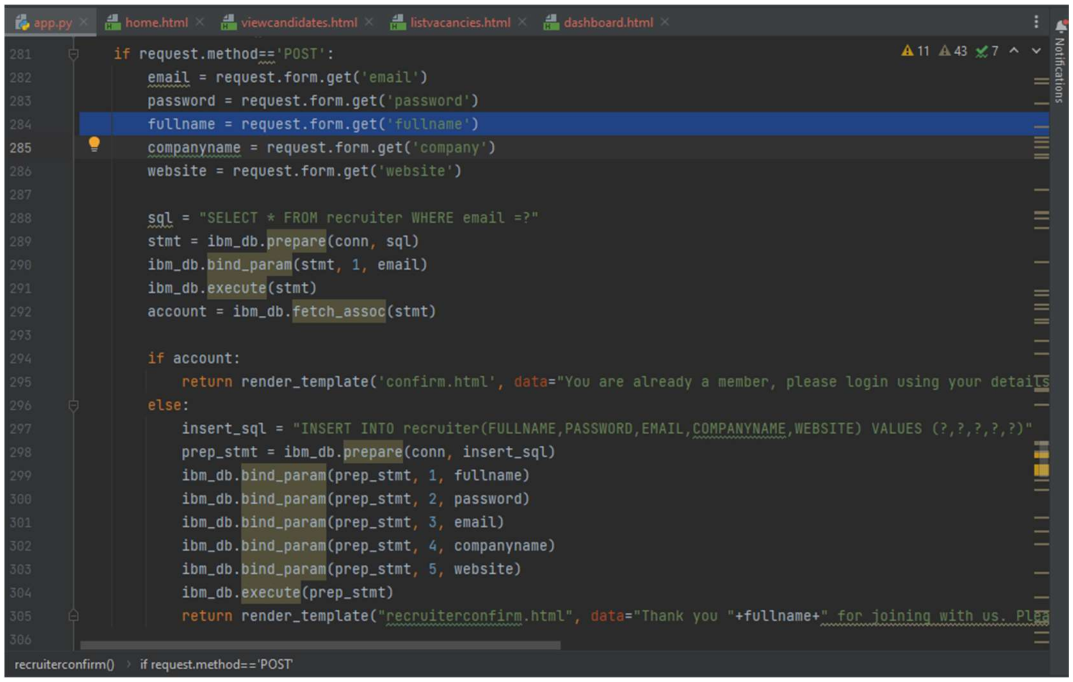
Table definition

USERS

Approximate 1 rows (32.0 KB)
Updated on 2022-11-18 17:58:15

Name	Data type	Nullable	Length	Scale
PINCODE	VARCHAR	Y	8	0
SKILLS	VARCHAR	Y	200	0
QUALIFICATION	VARCHAR	Y	200	0
SSLC	VARCHAR	Y	6	0
HSC	VARCHAR	Y	6	0

Table 2- Recruiters



```
    if request.method=='POST':
        email = request.form.get('email')
        password = request.form.get('password')
        fullname = request.form.get('fullname')
        companyname = request.form.get('company')
        website = request.form.get('website')

        sql = "SELECT * FROM recruiter WHERE email =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
            return render_template('confirm.html', data="You are already a member, please login using your details
        else:
            insert_sql = "INSERT INTO recruiter(FULLNAME,PASSWORD,EMAIL,COMPANYNAME,WEBSITE) VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, fullname)
            ibm_db.bind_param(prep_stmt, 2, password)
            ibm_db.bind_param(prep_stmt, 3, email)
            ibm_db.bind_param(prep_stmt, 4, companyname)
            ibm_db.bind_param(prep_stmt, 5, website)
            ibm_db.execute(prep_stmt)
            return render_template("recruiterconfirm.html", data="Thank you "+fullname+" for joining with us. Plea
```

z

Skill / Job Recommender Application

Team ID: PNT2022TMID34860

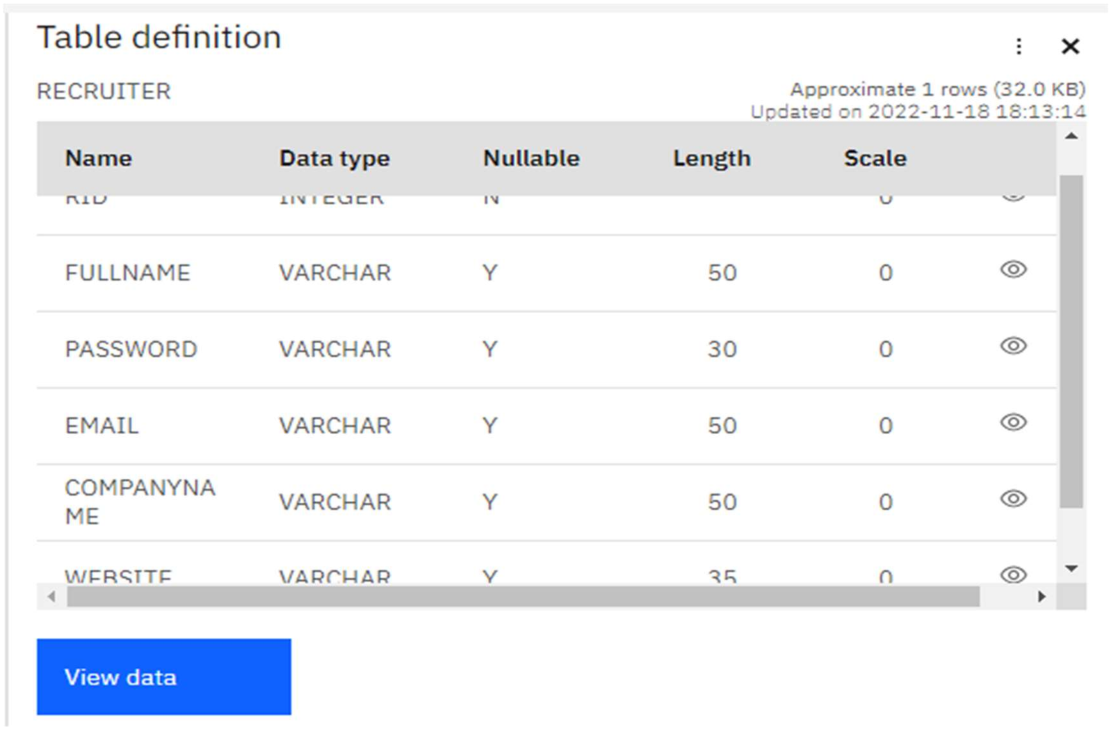


Table definition

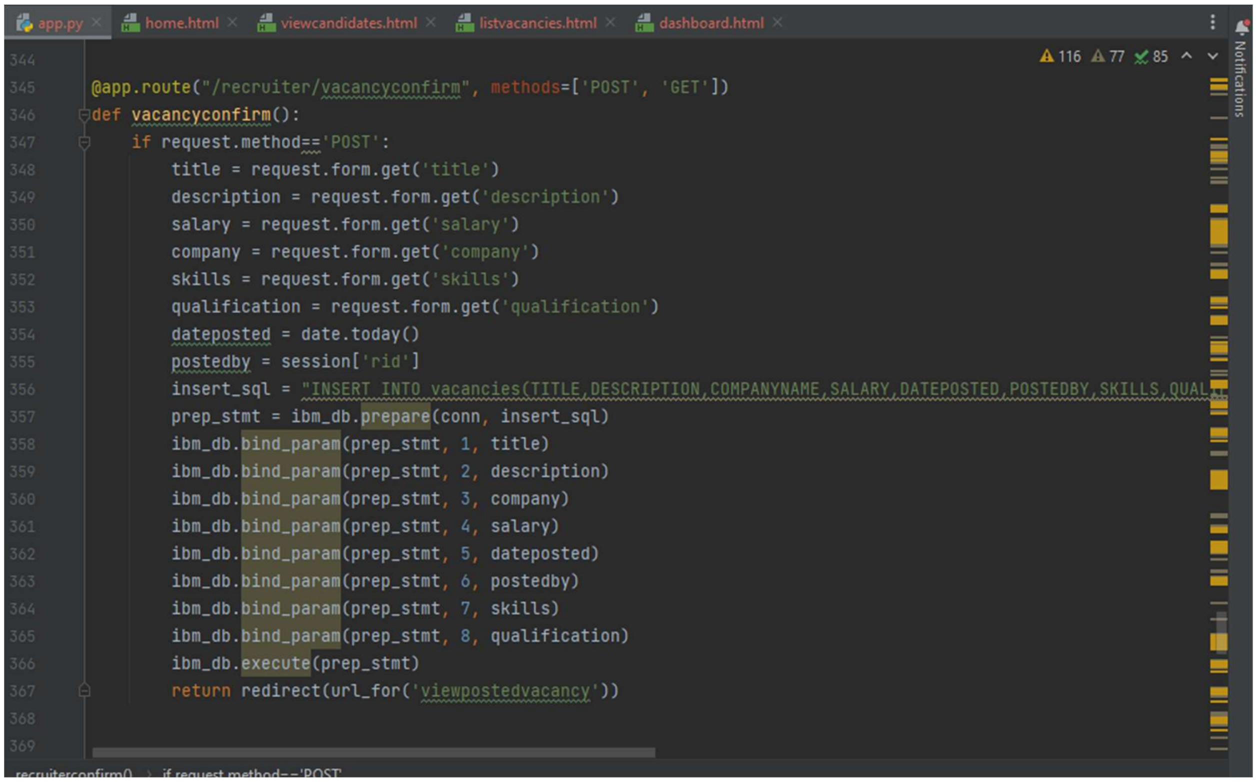
RECRUITER

Approximate 1 rows (32.0 KB)
Updated on 2022-11-18 18:13:14

Name	Data type	Nullable	Length	Scale
RID	INTEGER	N		0
FULLNAME	VARCHAR	Y	50	0
PASSWORD	VARCHAR	Y	30	0
EMAIL	VARCHAR	Y	50	0
COMPANYNAME	VARCHAR	Y	50	0
WEBSITE	VARCHAR	Y	35	0

View data

Table 3- Vacancies



```
@app.route("/recruiter/vacancyconfirm", methods=['POST', 'GET'])
def vacancyconfirm():
    if request.method=='POST':
        title = request.form.get('title')
        description = request.form.get('description')
        salary = request.form.get('salary')
        company = request.form.get('company')
        skills = request.form.get('skills')
        qualification = request.form.get('qualification')
        dateposted = date.today()
        postedby = session['rid']
        insert_sql = "INSERT INTO vacancies(TITLE,DESCRIPTION,COMPANYNAME,SALARY,DATEPOSTED,POSTEDBY,SKILLS,QUAL
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, title)
        ibm_db.bind_param(prep_stmt, 2, description)
        ibm_db.bind_param(prep_stmt, 3, company)
        ibm_db.bind_param(prep_stmt, 4, salary)
        ibm_db.bind_param(prep_stmt, 5, dateposted)
        ibm_db.bind_param(prep_stmt, 6, postedby)
        ibm_db.bind_param(prep_stmt, 7, skills)
        ibm_db.bind_param(prep_stmt, 8, qualification)
        ibm_db.execute(prep_stmt)
        return redirect(url_for('viewpostedvacancy'))
```

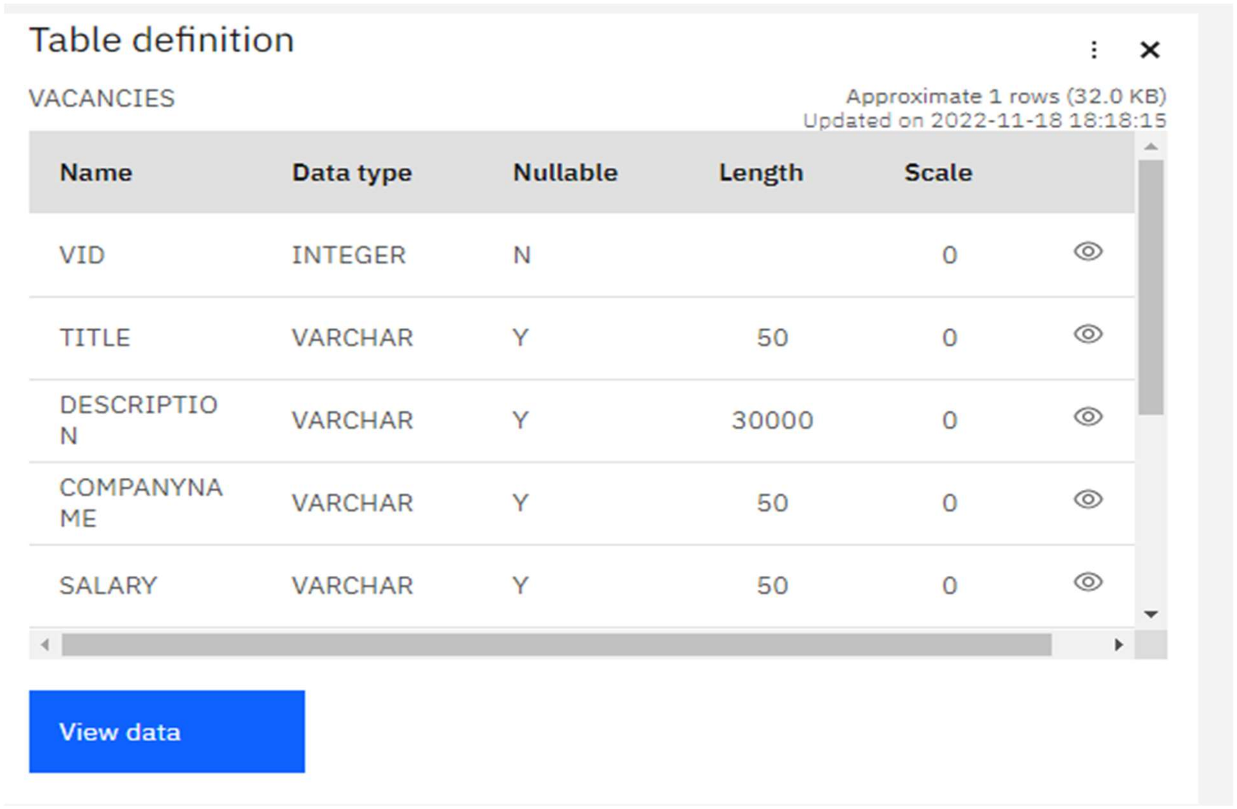


Table definition

VACANCIES

Approximate 1 rows (32.0 KB)
Updated on 2022-11-18 18:18:15

Name	Data type	Nullable	Length	Scale
VID	INTEGER	N		0
TITLE	VARCHAR	Y	50	0
DESCRIPTION	VARCHAR	Y	30000	0
COMPANYNAME	VARCHAR	Y	50	0
SALARY	VARCHAR	Y	50	0

Table 4-

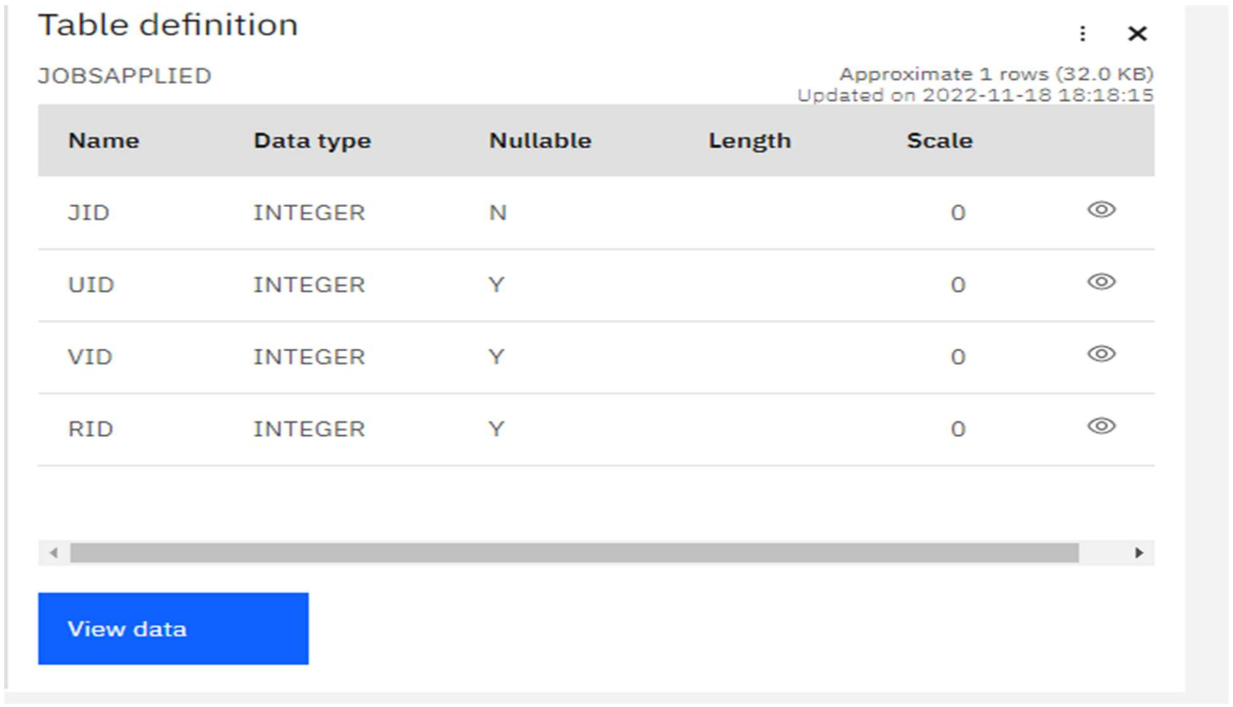


Table definition

JOBSAPPLIED

Approximate 1 rows (32.0 KB)
Updated on 2022-11-18 18:18:15

Name	Data type	Nullable	Length	Scale
JID	INTEGER	N		0
UID	INTEGER	Y		0
VID	INTEGER	Y		0
RID	INTEGER	Y		0

8.TESTING

1. Test Cases

Test Case ID	Purpose	TestCases	Result
TC1	Authentication	Password with length less than 10 characters	Pass
TC2	Authentication	password with length less than 40 characters	Pass
TC3	Authentication	password with length more than 50 characters	Fail
TC4	Authentication	User name left blank space	Fail
TC5	Authentication	Password field left blank space	Fail
TC6	Authentication	User id left blank space	Fail
TC7	Authentication	User id with length more than 50 characters	Fail
TC8	Authentication	User id with length 10 characters	Pass
TC9	Authentication	Phone number with alphabets	Fail
TC10	Authentication	Phone number with character length more than 10	Fail
TC11	Authentication	Phone number with character length 10	Pass
TC12	Authentication	Not verified website	Fail
TC13	Authentication	Verified website	Pass

2. User Acceptance Testing

Test Case ID	Test Case Description
TC_001	Verify if user is able to signup the platform
TC_002	Verify if user is able to create a profile
TC_003	verify if user is able to connect with people
TC_004	verify if user is able to search for job vacancies
TC_005	verify if user is able to use the chatbot
TC_006	verify if user is able to search job based on their location
TC_007	verify if user is able to apply for jobs
TC_008	verify if user is able to receive notification
TC_009	verify if user is able to find internship recommendation

9. RESULTS

1. Performance Metrics

- Profile of a user holds the information regarding the preferences of user on skill and domain specification.
- Profile of items holds the information regarding the skills and job domain that is required for that job.
- Companies that are targeting these job seekers are finding it challenging to identify the job seeker's skill and provide personalized job recommendations
- This system gives the ability to showcase users profile,expertise, recommendations and connections.
- It is a measurement tool utilized in company pages and individual profiles to track and measure their influence via specific goals.

- Customized notifications, timely and relevant updates about your connections and accounts.
- Mute or unmute option is available in order to avoid disturbance or other updates.
- Without authentication the user cannot login the application to prevent spam or fake messages.
- Reduces traffic in the server when logging in or active of too many people.

Merges with several companies and speed performance.

10. ADVANTAGES & DISADVANTAGES

a. Advantages

◊ **Use this platform as a research tool:**

Having an account in job recommendation application also means that you can use the site to research companies and job vacancies based on the user's qualification and interests.

◊ **Initiate connections:**

Users can start connecting with their school and college friends and eventually shift to their colleagues and alumnus from your university is good connections to have. They can share the opportunities to their friends.

◊ **Getting job alerts:**

When the user successfully update their profile they can get job alerts based on their qualifications, and skill.

◊ **Stay connected and get employed:**

The primary aim of this platform is linking employers with potential employees. So this is a platform for students and graduates to post their resume.

◊ **Deadline notification:**

The users will be notified about the deadline through the registered users email id.

- ◊ **Finding potential:**

This platform helps recruiters to look for candidates whose qualities match with their needs.

- ◊ **Query clarification:**

The users can post their queries in the chatbot and their doubts will be clarified.

b.Disadvantages

- ◊ **Potential spam:**

Users can receive spam messages from people they don't know, as well as a friend and connection requests from fake profiles.

- ◊ **Risk of identity theft:**

The users upload loads of personal information on their profile for recruiters to see. Hence, the users might stand a risk of losing important information to the public, resulting in identity theft.

- ◊ **Incomplete profile :**

Putting up an attractive profile that is appealing to the employers and prospective recruiters. The people however, find it hard to fill out the profile details completely due to one reason or the other.

11. CONCLUSION

We conclude this system with analysis of job description to recommend a job based on user's skills and preferences presents itself in recommending open position to the job seekers when looking for a new positions. Being an active member is always important for any discourse community one is involved in.Keeping profiles updates up-to-date allows one's connections to be informed on their recent awards or recognition within the community.Asking and answering questions shows interest and provides credibility for the member.

12. FUTURE SCOPE

Job recommender system provides opportunities for job seekers by suggesting relevant jobs to them. The main goal of this system is to provide the users with variety of job, companies offering based on their skills and fast responses. Management of their informations and skills can be stored in the database with high privacy and security. spam messages later can be reported to the related authority. Developing the scalability and reliability in this platform can be improved.

13. APPENDIX

Source Code:

```
from datetime import date

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

Skill / Job Recommender Application

Team ID: PNT2022TMID34860

```
import sendgrid
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import *
import sqlite3 as sql

conn = ibm_db.connect(
    "DATABASE=bludb;HOSTNAME=19af6446-6171-4641-8aba-
9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30699;SECURITY=SSL
;SslServerCertificate=DigiCertGlobalRootCA.crt;UID=gvc94094;PWD=FoHH6J0p5r8hxWSP;",
    "", "")

app = Flask(__name__,template_folder='templates', static_url_path='/static')
app.secret_key = 'jobskill111'


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


@app.route("/about")
def blog():
    return render_template('about.html')


def sendmail(toemail,name):
    message = Mail(
        from_email='jobrecommendator@gmail.com',
        to_emails=toemail,
        subject='Welcome to Job Portal - '+name,
        html_content='<strong>Hi '+name+', Welcome to Job Portal. Hope you will
find a suitable job . </strong>')
    try:
        sg =
SendGridAPIClient('SG.3VyaqJQ5TfSlopyMENco_g.baZUE6LnLDAqQLaF7gLqJbjpTzQwzrtlq2C3HT
MKo9k')
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e.message)



@app.route("/dashboard")
def dashboard():
    email = session['email']
    vacancies = []
    sql = "SELECT * FROM vacancies ORDER BY VID DESC LIMIT 2"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    vacancy = ibm_db.fetch_assoc(stmt)
    while vacancy != False:
```

z

```
        vacancies.append(vacancy)
        vacancy = ibm_db.fetch_assoc(stmt)
    return render_template("dashboard.html",data=email,vacancies=vacancies)



@app.route("/signup")
def signup():
    return render_template("signup.html")



@app.route("/confirm", methods=['POST', 'GET'])
def confirm():
    if request.method=='POST':
        email = request.form.get('email')
        password = request.form.get('password')
        fullname = request.form.get('fullname')
        gender = request.form.get('gender')
        dob = request.form.get('dob')
        address = request.form.get('address')
        city = request.form.get('city')
        state = request.form.get('state')
        pincode = request.form.get('pincode')
        skills = request.form.get('skills')
        qualification = request.form.get('qualification')
        sslc = request.form.get('sslc')
        hsc = request.form.get('hsc')

        sql = "SELECT * FROM users WHERE email =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
            return render_template('confirm.html', data="You are already a member,
please login using your details")
        else:
            insert_sql = "INSERT INTO
users(EMAIL,PASSWORD,FULLNAME,GENDER,DOB,ADDRESS,CITY,STATE,PINCODE,SKILLS,QUALIFIC
ATION,SSLC,HSC) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, email)
            ibm_db.bind_param(prep_stmt, 2, password)
            ibm_db.bind_param(prep_stmt, 3, fullname)
            ibm_db.bind_param(prep_stmt, 4, gender)
            ibm_db.bind_param(prep_stmt, 5, dob)
            ibm_db.bind_param(prep_stmt, 6, address)
            ibm_db.bind_param(prep_stmt, 7, city)
            ibm_db.bind_param(prep_stmt, 8, state)
            ibm_db.bind_param(prep_stmt, 9, pincode)
            ibm_db.bind_param(prep_stmt, 10, skills)
            ibm_db.bind_param(prep_stmt, 11, qualification)
            ibm_db.bind_param(prep_stmt, 12, sslc)
            ibm_db.bind_param(prep_stmt, 13, hsc)
            ibm_db.execute(prep_stmt)
     #       sendmail(email,fullname)
            return render_template("confirm.html", data="Thank you "+fullname+" for 
joining with us. Please Login to Continue !")



@app.route("/signin")
def signin():
    return render_template("signin.html")


@app.route("/enter", methods=['POST', 'GET'])
def enter():
    if request.method=='POST':
        email = request.form.get('email')
        password = request.form.get('password')

        sql = "SELECT * FROM users WHERE email =? and  password =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if account:
            session['email'] = email
            session['uid'] = account['UID']
            return redirect(url_for('dashboard'))
        else:
            return render_template("confirm.html", data="Invalid email or 
password")


@app.route("/listvacancies")
def listvacancies():
    vacancies = []
    sql = "SELECT * FROM vacancies ORDER BY VID DESC"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.execute(stmt)
    vacancy = ibm_db.fetch_assoc(stmt)
    while vacancy != False:
        vacancy['isapplied'] = isapplied(vacancy['VID'])
        vacancies.append(vacancy)
        vacancy = ibm_db.fetch_assoc(stmt)
    return render_template("listvacancies.html", data=vacancies, text="Posted 
Vacancies")


def isapplied(vid):
    sql = "SELECT * FROM JOBSAPPLIED WHERE vid =? and  uid =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, vid)
    ibm_db.bind_param(stmt, 2, session['uid'])
    ibm_db.execute(stmt)
```

```python
    account = ibm_db.fetch_assoc(stmt)
    if account:
        return True
    else:
        return False


@app.route("/searchvacancies", methods=['POST', 'GET'])
def searchvacancies():
    if request.method=='GET':
        vacancies = []
        term = request.args.get('term')
        searchterm = "%"+term+"%";
        sql = "SELECT * FROM vacancies WHERE LCASE(skills) LIKE ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, searchterm)
        ibm_db.execute(stmt)
        vacancy = ibm_db.fetch_assoc(stmt)
        if vacancy:
            while vacancy != False:
                vacancy['isapplied'] = isapplied(vacancy['VID'])
                vacancies.append(vacancy)
             #   print(isapplied(vacancy['VID']))
                vacancy = ibm_db.fetch_assoc(stmt)
            return render_template("listvacancies.html", data=vacancies,
text="Search for Vacancies - "+term)
        else:
            return render_template("listvacancies.html", text="No results - "+term)



@app.route("/applyjob", methods=['POST', 'GET'])
def applyjob():
    if request.method=='GET':
        vid = request.args.get('vid')
        rid = request.args.get('rid')
        insert_sql = "INSERT INTO jobsapplied(UID,VID,RID) VALUES (?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, session['uid'])
        ibm_db.bind_param(prep_stmt, 2, vid)
        ibm_db.bind_param(prep_stmt, 3, rid)
        ibm_db.execute(prep_stmt)
        return redirect(url_for('listvacancies'))




@app.route("/profile")
def profile():
    profiledetails = []
    sql = "SELECT * FROM users WHERE uid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session['uid'])
    ibm_db.execute(stmt)
    profile = ibm_db.fetch_assoc(stmt)
    if profile:
        while profile != False:
            profiledetails.append(profile)
            profile = ibm_db.fetch_assoc(stmt)
        return render_template("profile.html", data=profiledetails)
    else:
        return render_template('profile.html')


@app.route("/editprofile")
def editprofile():
    profiledetails = []
    sql = "SELECT * FROM users WHERE uid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session['uid'])
    ibm_db.execute(stmt)
    profile = ibm_db.fetch_assoc(stmt)
    if profile:
        while profile != False:
            profiledetails.append(profile)
            profile = ibm_db.fetch_assoc(stmt)
        return render_template("editprofile.html", data=profiledetails)
    else:
        return render_template('editprofile.html')



@app.route("/updateprofile", methods=['POST', 'GET'])
def updateprofile():
    if request.method=='POST':
        fullname = request.form.get('fullname')
        address = request.form.get('address')
        city = request.form.get('city')
        state = request.form.get('state')
        skills = request.form.get('skills')
        qualification = request.form.get('qualification')
        sslc = request.form.get('sslc')
        hsc = request.form.get('hsc')

        sql = "UPDATE users SET FULLNAME = ?, ADDRESS = ? , CITY = ? , STATE = ? ,
SKILLS = ? , QUALIFICATION = ? , SSLC = ? , HSC = ? WHERE uid = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, fullname)
        ibm_db.bind_param(stmt, 2, address)
        ibm_db.bind_param(stmt, 3, city)
        ibm_db.bind_param(stmt, 4, state)
        ibm_db.bind_param(stmt, 5, skills)
        ibm_db.bind_param(stmt, 6, qualification)
        ibm_db.bind_param(stmt, 7, sslc)
        ibm_db.bind_param(stmt, 8, hsc)
        ibm_db.bind_param(stmt, 9, session['uid'])
        ibm_db.execute(stmt)
        return redirect(url_for('profile'))
```

z

```
@app.route("/logout")
def logout():
    session.clear()
    return redirect(url_for('index'))


@app.route("/recruiter/recruitersignup")
def recruitersignup():
    return render_template("recruitersignup.html")


@app.route("/recruiter/recruiterconfirm", methods=['POST', 'GET'])
def recruiterconfirm():
    if request.method=='POST':
        email = request.form.get('email')
        password = request.form.get('password')
        fullname = request.form.get('fullname')
        companyname = request.form.get('company')
        website = request.form.get('website')

        sql = "SELECT * FROM recruiter WHERE email =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
            return render_template('confirm.html', data="You are already a member,
please login using your details")
        else:
            insert_sql = "INSERT INTO
recruiter(FULLNAME,PASSWORD,EMAIL,COMPANYNAME,WEBSITE) VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, fullname)
            ibm_db.bind_param(prep_stmt, 2, password)
            ibm_db.bind_param(prep_stmt, 3, email)
            ibm_db.bind_param(prep_stmt, 4, companyname)
            ibm_db.bind_param(prep_stmt, 5, website)
            ibm_db.execute(prep_stmt)
            return render_template("recruiterconfirm.html", data="Thank you
"+fullname+" for joining with us. Please Login to Continue !")



@app.route("/recruiter/recruitersignin")
def recruitersignin():
    return render_template("recruitersignin.html")


@app.route("/recruiter/enterrecruiter", methods=['POST', 'GET'])
def enterrecruiter():
    if request.method=='POST':
        email = request.form.get('email')
        password = request.form.get('password')
        sql = "SELECT * FROM recruiter WHERE email =? and  password =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if account:
            session['recruiteremail'] = email
            session['rid'] = account['RID']
            return redirect(url_for('recruiterdashboard'))
        else:
            return render_template("recruiterconfirm.html", data="Invalid email or
password")


@app.route("/recruiter/recruiterdashboard")
def recruiterdashboard():
    recruiteremail = session['recruiteremail']
    return render_template("recruiterdashboard.html",data=recruiteremail)


@app.route("/recruiter/addvacancy")
def addvacancy():
    return render_template("addvacancy.html")


@app.route("/recruiter/vacancyconfirm", methods=['POST', 'GET'])
def vacancyconfirm():
    if request.method=='POST':
        title = request.form.get('title')
        description = request.form.get('description')
        salary = request.form.get('salary')
        company = request.form.get('company')
        skills = request.form.get('skills')
        qualification = request.form.get('qualification')
        dateposted = date.today()
        postedby = session['rid']
        insert_sql = "INSERT INTO
vacancies(TITLE,DESCRIPTION,COMPANYNAME,SALARY,DATEPOSTED,POSTEDBY,SKILLS,QUALIFICA
TION) VALUES (?,?,?,?,?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, title)
        ibm_db.bind_param(prep_stmt, 2, description)
        ibm_db.bind_param(prep_stmt, 3, company)
        ibm_db.bind_param(prep_stmt, 4, salary)
        ibm_db.bind_param(prep_stmt, 5, dateposted)
        ibm_db.bind_param(prep_stmt, 6, postedby)
        ibm_db.bind_param(prep_stmt, 7, skills)
        ibm_db.bind_param(prep_stmt, 8, qualification)
        ibm_db.execute(prep_stmt)
        return redirect(url_for('viewpostedvacancy'))


@app.route("/recruiter/viewpostedvacancy")
def viewpostedvacancy():
    vacancies = []
    sql = "SELECT * FROM vacancies WHERE postedby =? ORDER BY VID DESC"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, session['rid'])
    ibm_db.execute(stmt)
    vacancy = ibm_db.fetch_assoc(stmt)
    while vacancy != False:
        vacancies.append(vacancy)
        vacancy = ibm_db.fetch_assoc(stmt)
    return render_template("viewpostedvacancy.html", data=vacancies)


@app.route("/recruiter/viewcandidates", methods=['POST', 'GET'])
def viewcandidates():
    if request.method == 'GET':
        vid = request.args.get('vid')
        profiledetails = []
        userdata = []
        sql = "SELECT * FROM JOBSAPPLIED WHERE VID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, vid)
        ibm_db.execute(stmt)
        profile = ibm_db.fetch_assoc(stmt)
        if profile:
            while profile != False:
                userdata.append(viewcandidateslist(profile['UID']))
                profiledetails.append(profile)
                profile = ibm_db.fetch_assoc(stmt)
            return render_template("viewcandidates.html", data=userdata)
        else:
            return render_template("viewcandidates.html")



def viewcandidateslist(uid):
    viewcandidates = []
    sql = "SELECT * FROM USERS WHERE UID = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, uid)
    ibm_db.execute(stmt)
    profile = ibm_db.fetch_assoc(stmt)
    if profile:
        while profile != False:
            viewcandidates.append(profile)
            return profile
            profile = ibm_db.fetch_assoc(stmt)
            return profile



@app.route("/recruiter/logout")
def recruiterlogout():
    session.clear()
    return redirect(url_for('index'))
```

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

Source code link:
https://drive.google.com/drive/folders/1447_r08j2BynqT8a46bk9AUCqczvV0r2?usp=share_link

GitHub and project Demo Link:

DemoLink:https://drive.google.com/file/d/1zjHR-Gt3D0BCZpDpi0UyAwdH4WYzdTUE/view?usp=share_link

Git hub Link: https://github.com/IBM-EPBL/IBM-Project-30731-1660159126