**SKILL / JOB RECOMMENDER APPLICATION**

**TEAMID**: PNT2022TMID20598

TEAM LEADER        : PRADEEP RAJADURAI (49621915012)

TEAM MEMBER      : MUTHULAKSHMI (49621915041)

TEAM MEMBER      : VENKATRAMAN (49621915024)

TEAM MEMBER      : THATCHAINI (49621915026)

**1.INTRODUCTION**

**1.1 PROJECT OVERVIEW**

Finding a job in today’s market is a major challenge. Using job search websites is a frequent strategy for looking for a job. instead of spending the time to look through newspapers, business websites, and other conventional job advertisements. With only one click, a job search website may complete all of these tasks. A job search engine makes it easier for job searchers and employers to connect and share available positions. Many applicants want to apply to and work for these organizations since there are a rising number of financially sound, reliable, and promising technical companies/ start-ups on the web that are currently in high demand. They frequently miss out on these postings because there are a vast number of systems already in place that display millions of jobs that are typically completely irrelevant to the users. There are various options but few that have been streamlined. Job seekers frequently discover that they are unable to obtain the right occupation for themselves based on the actual abilities or interests of an individual. Using job search engines like LinkedIn, indeed, and others, Job seekers frequently conduct their job searches online. When using these websites, a job seeker often has two options: creating and/or updating a professional profile with information about their education, professional experience, professional skills, and other, and receiving tailored job recommendations based on this information,  or performing a search using keywords related to the job vacancy they are looking for. Sites that support to the former case are more widely used and have a simpler layout, but their recommendations are less reliable than those of the sites that use profile data. Personalized job recommendation sites have implemented a range of recommender system types, including content-based filtering, collaborative filtering, knowledge-based approaches, and hybrid approaches. Furthermore, the majority of these job recommender systems base their recommendations on the complete profile of job searchers in addition to taking into account additional data sources like social networking activity, web search history, etc. Despite the fact that a variety of data sources can be helpful to enhance job recommendations, prior studies revealed that the optimal person-job fit is only feasible when a job seeker's specific skills match the demands of a job offer. Numerous employment firms have developed methods for offering the job board in order to serve the continuous cycle of the recruiting process from the viewpoint of the job seeker. One searches for and applies for jobs that they believe are relevant to them. As there are numerous job boards, candidates typically use the one that offers the best services to them, such as generating a CV, building a job profile, and recommending new jobs to a job seeker. In their pursuit of new career chances that match their skills, job seekers have grown more tenacious and assertive. Companies who are focusing on these job seekers are having trouble determining their skill set and making tailored employment recommendations. With our skill recommender solution, either a skilled or a fresher user may sign up, search for jobs using the search bar, or speak with the chatbot directly to land their ideal position. Therefore, this system addresses the idea from the perspective of the data, placing more emphasis on the quality of the data than the number.

**1.2 PURPOSE**

Every system conceivable use recommender system, including those for books, movies, and other media. However, based on the application domain, many recommendation types may offer. In job recommendation systems, there are various job seekers, having different education levels and skills. Each job seeker anticipates receiving only those employment recommendations that are extremely pertinent to their individual background information. By using ratings and comparing applicants' talents to the needed ones, our recommendation system is created for positions. For both job suppliers and job searchers, the system may result in significant time savings in addressing their demands. The value of a job recommendation system that takes the user's skill set into account. The purpose of this recommendation application 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.  A system that not only suggests jobs but also identifies the skill sets required for such jobs might aid users in learning more about those skill sets. A system that is appropriate for job seekers is needed to deal with the issues of urbanization and employment trends in this constantly changing environment. This methodology is intended to assist recruiters in making skill-based hiring decisions. It is created in a way that will assist close the gap between the two of them, making it easy for both the recruiter and the job seeker to use.

**2.LITERATURE SURVEY**

**2.1 EXISTING PROBLEM**

The existing system had some potential for proposing fraudulent jobs as well as jobs that were irrelevant to job seekers, which saddened and disappointed them. We offered our application in an effort to correct the irrelevant job recommendations and direct job seekers to the most appropriate positions.

**2.2 REFERENCES**

[1] R. J. Mooney and L. Roy, "Content-Based Book Recommending Using Learning for Text Categorization," in Proceedings of DL ’00: Proceedings of the Fifth ACM Conference on Digital Libraries, New York, NY, pp. 13-20, 2000.

[2] Li-Ping Jing, Hou-Kuan Huang, Hong-Bo Shi, “Improved feature selection approach TFIDF in text mining”, International Conference on Machine Learning and Cybernetics, pp. 944-946, 2002, doi:10.1109/icmlc.2002.1174522.

[3] Shouning Qu ,Sujuan Wang,Yan Zou, ” Improvement of Text Feature Selection Method Based on TFIDF”, International Seminar on Future Information Technology and Management Engineering, pp. 79-81, 2008, doi:10.1109/fitme.2008.25.

[4] I. A. Braga, "Evaluation of stopwords removal on the statistical approach for automatic term extraction," Seventh Brazilian Symposium in Information and Human Language Technology, pp. 142-149, 2009.

[5] Nikolaos D. Almalis, Prof. George A. Tsihrintzis, Nikolaos Karagiannis, Aggeliki D. Strati, “A new content-www.ijcrt.org © 2022 IJCRT | Volume 10, Issue 8 August 2022 | ISSN: 2320-2882 IJCRT2208099 International Journal of Creative Research Thoughts (IJCRT) www.ijcrt.org a784 based job recommendation algorithm for job seeking and recruiting”, 6th International Conference on Information, Intelligence, Systems and Applications (IISA), pp. 1-7, 2015, doi:10.1109/iisa.2015.7388018.

[6] Mohammad Alodadi and Vandana P. Janeja, “ Similarity in Patient Support Forums Using TF-IDF and Cosine Similarity Metrics”, International Conference on Healthcare Informatics, pp. 521-522, 2015, doi:10.1109/ichi.2015.99.

[7] L. Zahrotun, “Comparison jaccard similarity, cosine similarity and combined both of the data clustering with shared nearest neighbor method,” Computer Engineering and Applications Journal. vol. 5. Pp. 11-18, 2016, doi:10.18495/comengapp.v5i1.160, 2016.

[8] Peng Yi, Cheng Yang ,Chen Li, Yingya Zhang, “A Job Recommendation Method Optimized by Position Descriptions and Resume Information”, IEEE Advanced Information Management, Communicates, Electronic and  Automation Control Conference (IMCEC), pp. 762 -764, March 2017, doi:10.1109/rteict.2017.8256590.

[9] Minh-Luan Tran, Anh-Tuyen Nguyen, Quoc-Dung Nguyen, Tin Huynh, “A comparison study for job recommendation”, International Conference on Information and Communications (ICIC), pp. 199-204, 2017, doi:10.1109/infoc.2017.8001667.

[10] Gokul P.P, Akhil BK, Shiva Kumar K.M, “Sentence similarity detection in Malayalam language using cosine similarity”, 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), pp. 221-225, 2017, doi:10.1109/rteict.2017.8256590.

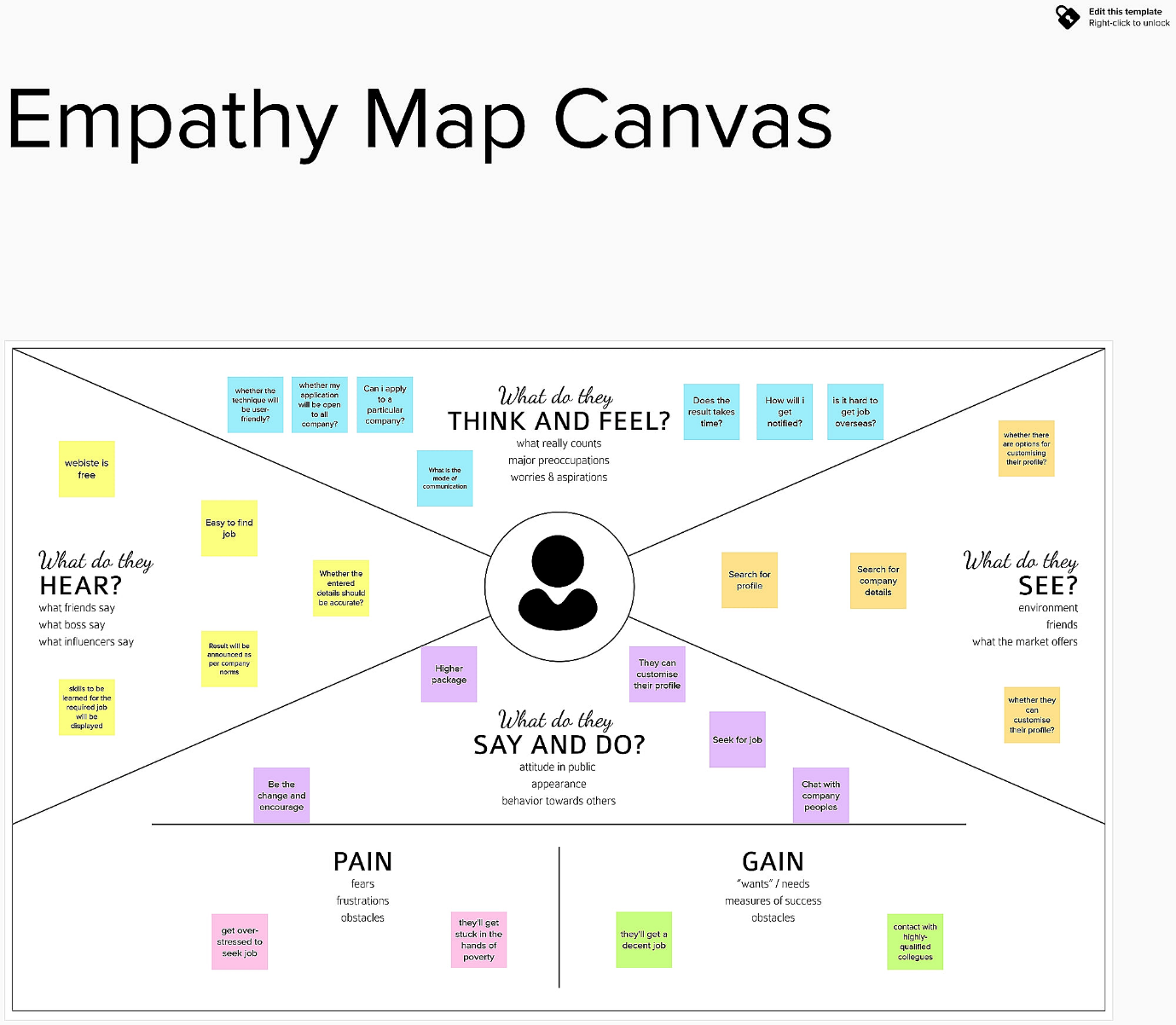
**2.3 PROBLEM STATEMENT DEFINITION**

Dealing with the enormous amount of recruiting information on the Internet, a job seeker always spends hours to find useful ones. Many times, people who lack industry knowledge are unclear about what exactly they need to learn in order to get a suitable job for them. We address the problem of recommending suitable jobs to people who are seeking for a new job. 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.

**3.IDEATION AND PROPOSED SOLUTION**

**3.1 EMPATHY MAP CANVAS**

Following is the Empathy Map Canvas of Skill/ Job Recommender System.



**3.2 IDEATION AND BRAINSTORMING**

Following are the Ideation and Brainstorm of Skill/ Job Recommender System.



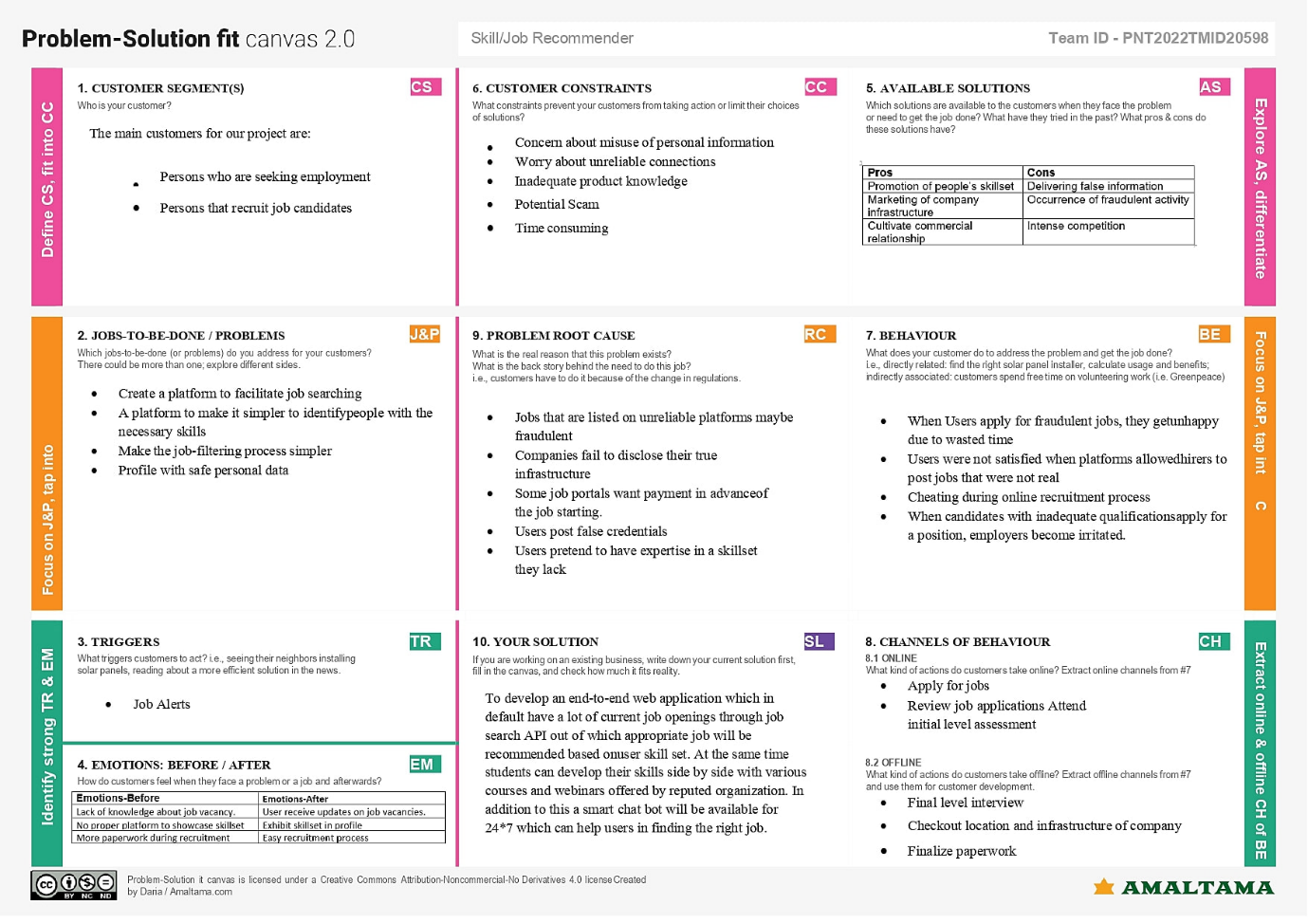
**3.3 PROPOSED SOLUTION**

Following is the Proposed Solutions of Skill/ Job Recommender System.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
|  | Problem Statement  (Problems to be solve) | 1. Having better skills but wondering which job will best suits you? 2. We are giving opportunity to job Seekers. 3. User can access large no of data. 4. 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. 5. 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. |
|  | Idea/ Solution description | 1. To focuses on fit for feature. 2. To provide user what company expect. 3. 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. 4. Put forward the proposal of a framework for job recommendation based on professional skills of job seekers. 5. Carried out an evaluation to quantify empirically the recommendation abilities of two state of the art methods, considering different configurations, within the proposed framework. 6. We thus present a general panorama of job recommendation task aiming to facilitate research and real-world application design regarding this important issue. |
|  | Novelty / Uniqueness | 1. We provide high Data Security. 2. We provide Mobile and computer both platforms. 3. The best position is suggested to any person according to her skills. While the position of known profiles is assumed to be correct, it should be noted that there are usually multiple advisable positions corresponding to a set of skills. A recommendation system should return a set of most likely positions and all of them can be equally valid. 4. The recommendation method we use is simply based on representing both positions and profiles as comparable vectors and seeking for each profile the positions with the most similar vectors. |
|  | Social Impact / Customer Satisfaction | 1. At last, we believe that two people with equal talent should have equal access to opportunity and we’re committed to making this vision reality through our project. 2. We are providing Friendly approach and employability. 3. Students will be benefited as they will get to know which job suits them based on their skills. |
|  | Business Model (Revenue Model) | 1. We are connecting you with other professionals also with companies and recruiters. Along with professionals, it also serves companies and even charges for providing certain premium services. 2. We can provide the application for job seekers in a subscription based and we can share the profiles with companies and generate the revenue by providing them best profiles |
|  | Scalability of the Solution | 1. Scalability is a custom training and organizational development firm dedicated to helping businesses scale. 2. Data can be scaled up and scaled down according to number of current job openings. |

**3.4 PROBLEM SOLUTION FIT**

Following is the Problem Solution Fit of Skill/ Job Recommender System.



**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 | User Registration | 1. Registration through Form 2. Registration through Gmail 3. Registration through LinkedIn |
| FR-2 | User Confirmation | 1. Confirmation via Email 2. Confirmation via OTP |
| 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 issue |
| FR-5 | Job registration | Copy of the company the user applied for with its registration/description details will be sent to the registered email id. |
| FR-6 | Logout |  |

**4.2 NON-FUNCTIONAL REQUIREMENTS**

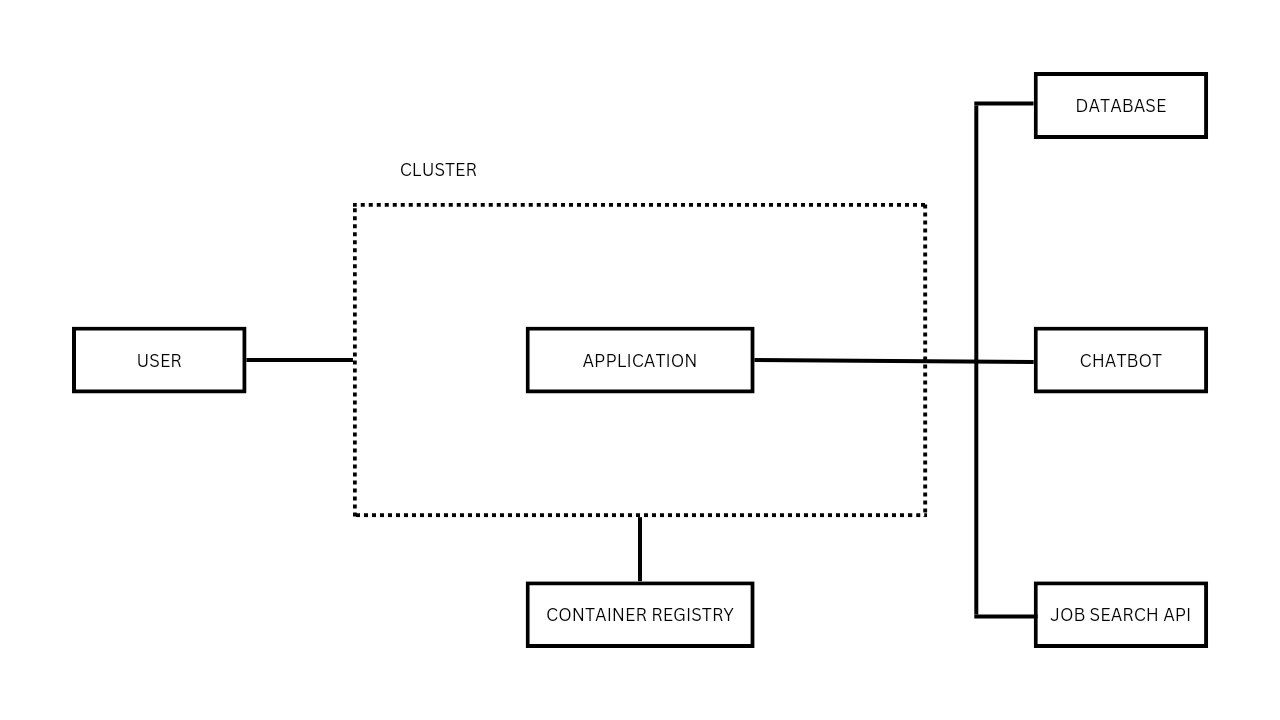
Following are the non-functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | 1. 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.) 2. Reduce information overload by generating personalized job suggestions. |
| NFR-2 | **Security** | 1. Using of SSL certificate (Python Flask to Cloud connect) will provide security to the project. 2. Database will be safely stored in DB2. |
| NFR-3 | **Reliability** | To make sure the webpage doesn’t                go down due to network traffic. |
| NFR-4 | **Performance** | 1. Focus on loading the webpage as quickly as possible irrespective of the number of user/integrator traffic. 2. Carry out an evaluation to quantify empirically the recommendation abilities of two state-of-the-art methods, considering different configurations, within the proposed framework |
| NFR-5 | **Availability** | 1. The scraper is set up to avoid duplicate job offers, thus all the job offers are unique. 2. To making the user reliable. This webpage will be available to all users (network connectivity is necessary) at any given point of time. 3. 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. |
| NFR-6 | **Scalability** | 1. Increasing the storage space of database can increase the number of users. 2. Add some features in future to make the webpage unique and attractive |

**5.PROJECT DESIGN**

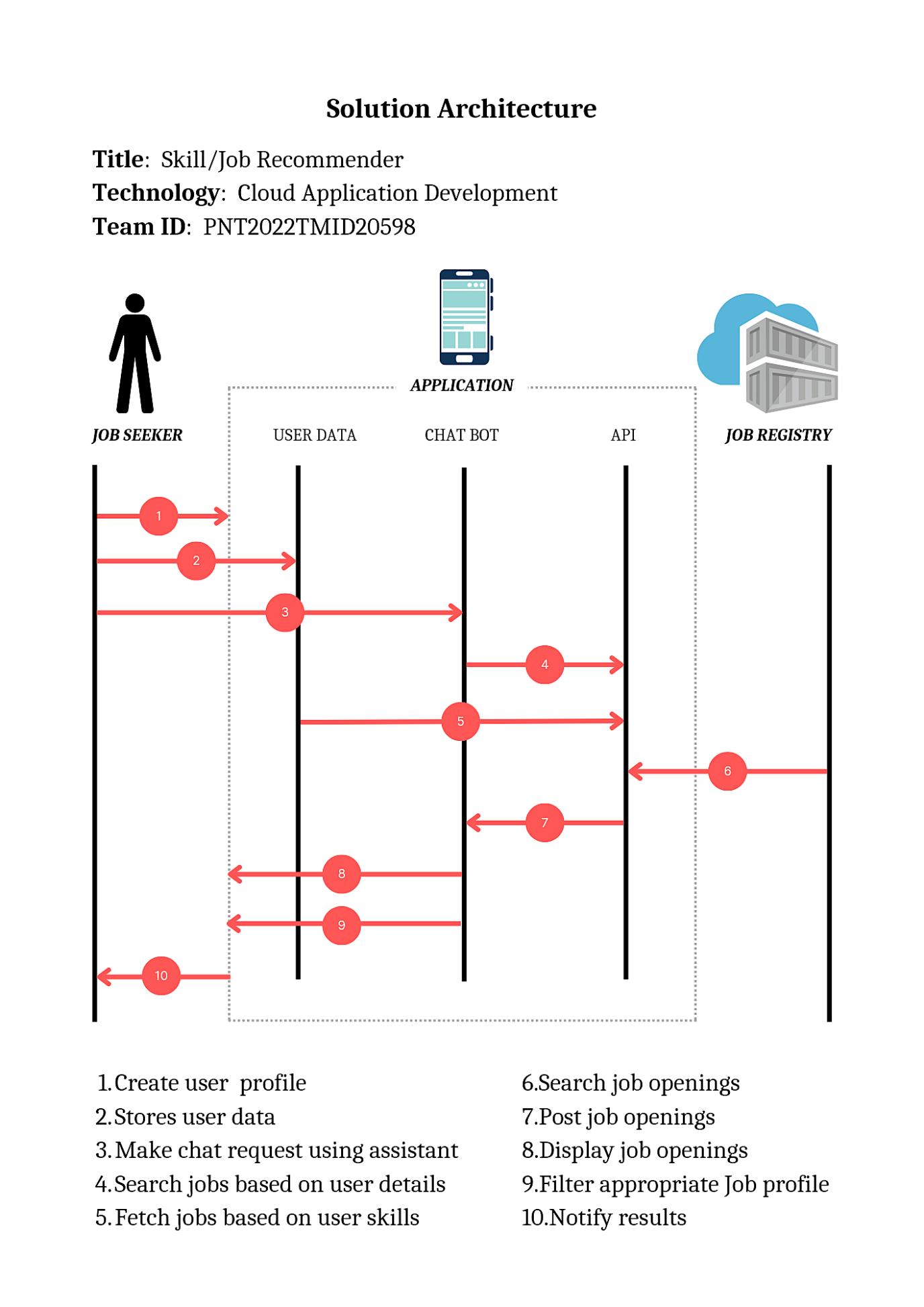
**5.1 DATA FLOW DIAGRAM**

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

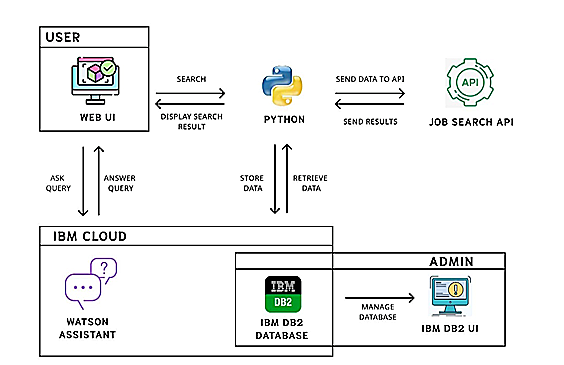


**5.2 SOLUTION AND TECHNICAL ARCHITECTURE**

To resolve this problem, 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.



**TECHNICAL ARCHITECTURE**



**5.2.1 COMPONENTS AND TECHNOLOGY**

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **Component** | **Description** | **Technology** |
| 1. | User Interface | How userinteracts with application e.g., Web UI, Mobile App,Chatbot etc. | HTML, CSS, JavaScript, Bootstrap |
| 2. | Application Logic-1 | Logic for a process in the application | Python |
| 3. | Application Logic-2 | Logic for a process in the application | IBM WatsonSTT service |
| 4. | Application Logic-3 | Logic for a process in the application | IBM WatsonAssistant |
| 5. | Database | Data Type,Configurations etc. | MySQL |
| 6. | Cloud Database | Database Serviceon Cloud | IBM DB2, IBM Cloudant etc. |
| 7. | File Storage | File storage requirements | IBM Block Storage or Other Storage Service or Local Filesystem |
| 8. | Infrastructure  (Server / Cloud) | Application Deployment on Local System/ Cloud LocalServer Configuration:Cloud ServerConfiguration: | Local, CloudFoundry, Kubernetes, etc. |

**5.2.2 APPLICATION CHARACTERISTICS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Characteristics** | **Description** | **Technology** |
| 1. | Open-Source Frameworks | List the open-source frameworks used | IBM cloud Kubernetes service |
| 2. | Security  Implementations | List allthe security / access controls implemented, use of firewalls etc. | e.g., SHA-256, Encryptions, IAM Controls, OWASPetc. |
| 3. | Scalable Architecture | Justify the scalability of architecture (3 – tier, Micro-services) | Technology used |
| 4. | Availability | Justify the availability of application (e.g.,use of loadbalancers, distributed servers etc.) | Technology used |
| 5. | Performance | Design consideration for the performance of the application (numberof requests per sec, useof  Cache, use of CDN’s)etc. | Technology used |

**5.3 USER STORIES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional**  **Requirement (Epic)** | **User Story Number** | **User Story/ Task** | **Acceptance criteria** | **Priority** | **Release** |
| Customer (Mobileuser) | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming  my password. | I can access my account / dashboard | High | Sprint-1 |
|  |  | USN-2 | As a user, I will receive confirmation email once I have registered for the application | I can receive confirmation email& click confirm | High | Sprint-1 |
|  |  | USN-3 | As a user, I can register for the application through LinkedIn | I canregister & accessthe  dashboard with Linkedin  Login | Low | Sprint-2 |
|  |  | USN-4 | As a user, I can register for the application through Gmail | I can register and access  the dashboard through  Gmail also | Medium | Sprint-1 |
|  | Login | USN-5 | As a user, I can log into the application by entering email& password | I can log on to the  application through email idand password | High | Sprint-1 |
|  | Dashboard | USN-6 | As a user, I can login and chat with the chatbot | Once I logged on the  application I can chat with thechatbot | High | Sprint-3 |
| Customer (Web user) | Registration | USN-7 | As a user, I can log on and register the application for the services being provided | I can access my account / dashboard | High | Sprint-1 |
|  |  | USN-8 | As a user, I will receive confirmation email onceI have registered for the application | I can receive confirmation email& click confirm | High | Sprint-1 |
|  | Login | USN-9 | As a user, I can log into the application by entering email& password | I can log on to the  application through email idand password | High | Sprint-1 |
| Customer care executive | Should Regularize the Send grid service | USN-10 | As a executive and service operator of the service they should make sure that service provided are properly send and received by  the user. |  | High | Sprint-2 |
|  | Should monitor the chatbot regularly whether  working or not | USN-11 | As a executive to provide a quality based service chatbot is important for assisting if anyassistance is needed for the user |  | High | Sprint-2 |

**6.PROJECT PLANNING AND SCHEDULING**

**6.1 SPRINT PLANNING AND ESTIMATION**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Total Story**  **Points** | **Duration** | **Sprint Start Date** | **Sprint**  **End-Date(Planned)** | **Story Points Completed**  **(as on planned date)** | **Sprint ReleaseDate(actual)** |
| S-1 | 20 | 6 Days | 24 Oct2022 | 29 Oct 2022 | 20 | 29 Oct2022 |
| S-2 | 20 | 6 Days | 31 Oct2022 | 05 Nov 2022 | 20 | 05 Nov 2022 |
| S-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 20 | 12 Nov 2022 |
| S-4 | 20 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 20 | 19 Nov 2022 |

**6.2 SPRINT DELIVERY SCHEDULE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional**  **Requirements (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priorit y** | **Team Members** |
| S-1 | User Panel | USN-1 | The user will access the website and view theproducts it provides after registering in. | 20 | High | Thatchaini S  Muthulakshmi A  Venkatraman S  Pradeep Rajadurai W |
| S-2 | Admin panel | USN-2 | The administrator's task is to look over the stock database andmonitor on everything  that people are buying. | 20 | High | Thatchaini S  Muthulakshmi A  Venkatraman S  Pradeep Rajadurai W |
| S-3 | Chat Bot | USN-3 | The user can directly talk to Chatbot regarding the products. Get the recommendations based on information provided  by the user. | 20 | High | Thatchaini S  Muthulakshmi A  Venkatraman S  Pradeep Rajadurai W |
| S-4 | final delivery | USN-4 | Container of applications usingdocker kubernetes and deployment the application.  Create the documentation andfinal submit the application | 20 | High | Thatchaini S  Muthulakshmi A  Venkatraman S  Pradeep Rajadurai W |

**7.CODING AND SOLUTION**

**7.1 FEATURE 1**

from flask import Flask, render\_template, request, redirect, url\_for, session

import ibm\_db

import re

app = Flask(\_\_name\_\_)

app.secret\_key = &#39;a&#39;

conn=ibm\_db.connect(&quot;DATABASE=bludb;HOSTNAME=824dfd4d-99de-440d-9991-

629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30119;SECURIT

Y=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=rzg70177;PWD=xHVRjmaJ8

pjbvjva&quot;,&#39;&#39;,&#39;&#39;)

@app.route(&#39;/&#39;)

def homer():

    return render\_template(&#39;homeCpy.html&#39;)

@app.route(&#39;/login&#39;,methods =[&#39;GET&#39;, &#39;POST&#39;])

def login():

    global userid

    msg = &#39;&#39;

    if request.method == &#39;POST&#39; :

        username = request.form[&#39;username&#39;]

        password = request.form[&#39;password&#39;]

        sql = &quot;SELECT \* FROM users WHERE username =? AND password=?&quot;

        stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(stmt,1,username)

        ibm\_db.bind\_param(stmt,2,password)

        ibm\_db.execute(stmt)

        account = ibm\_db.fetch\_assoc(stmt)

        print (account)

        if account:

            session[&#39;loggedin&#39;] = True

            session[&#39;id&#39;] = account[&#39;USERNAME&#39;]

            session[&#39;mail&#39;] = account[&quot;EMAIL&quot;]

            userid=  account[&#39;USERNAME&#39;]

            session[&#39;username&#39;] = account[&#39;USERNAME&#39;]

            msg = &#39;Logged in successfully !&#39;

            msg = &#39;Logged in successfully !&#39;

            return render\_template(&#39;dashboardCpy.html&#39;, msg = msg)

        else:

            msg = &#39;Incorrect username / password !&#39;

    return render\_template(&#39;login.html&#39;, msg = msg)

@app.route(&#39;/register&#39;, methods =[&#39;GET&#39;, &#39;POST&#39;])

def register():

    msg = &#39;&#39;

    if request.method == &#39;POST&#39; :

        username = request.form[&#39;username&#39;]

        email = request.form[&#39;email&#39;]

        password = request.form[&#39;password&#39;]

        sql = &quot;SELECT \* FROM users WHERE username =?&quot;

        stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(stmt,1,username)

        ibm\_db.execute(stmt)

        account = ibm\_db.fetch\_assoc(stmt)

        print(account)

        if account:

            msg = &#39;Account already exists !&#39;

        elif not re.match(r&#39;[^@]+@[^@]+\.[^@]+&#39;, email):

            msg = &#39;Invalid email address !&#39;

        elif not re.match(r&#39;[A-Za-z0-9]+&#39;, username):

            msg = &#39;name must contain only characters and numbers !&#39;

        else:

            insert\_sql = &quot;INSERT INTO  users VALUES (?, ?, ?)&quot;

            prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

            ibm\_db.bind\_param(prep\_stmt, 1, username)

            ibm\_db.bind\_param(prep\_stmt, 2, email)

            ibm\_db.bind\_param(prep\_stmt, 3, password)

            ibm\_db.execute(prep\_stmt)

            msg = &#39;You have successfully registered !&#39;

            return redirect(url\_for(&#39;login&#39;))

    elif request.method == &#39;POST&#39;:

        msg = &#39;Please fill out the form !&#39;

    return render\_template(&#39;register.html&#39;, msg = msg)

@app.route(&#39;/dashboard&#39;)

def dash():

    return render\_template(&#39;dashboardCpy.html&#39;)

@app.route(&#39;/logout&#39;)

def logout():

   session.pop(&#39;loggedin&#39;, None)

   session.pop(&#39;id&#39;, None)

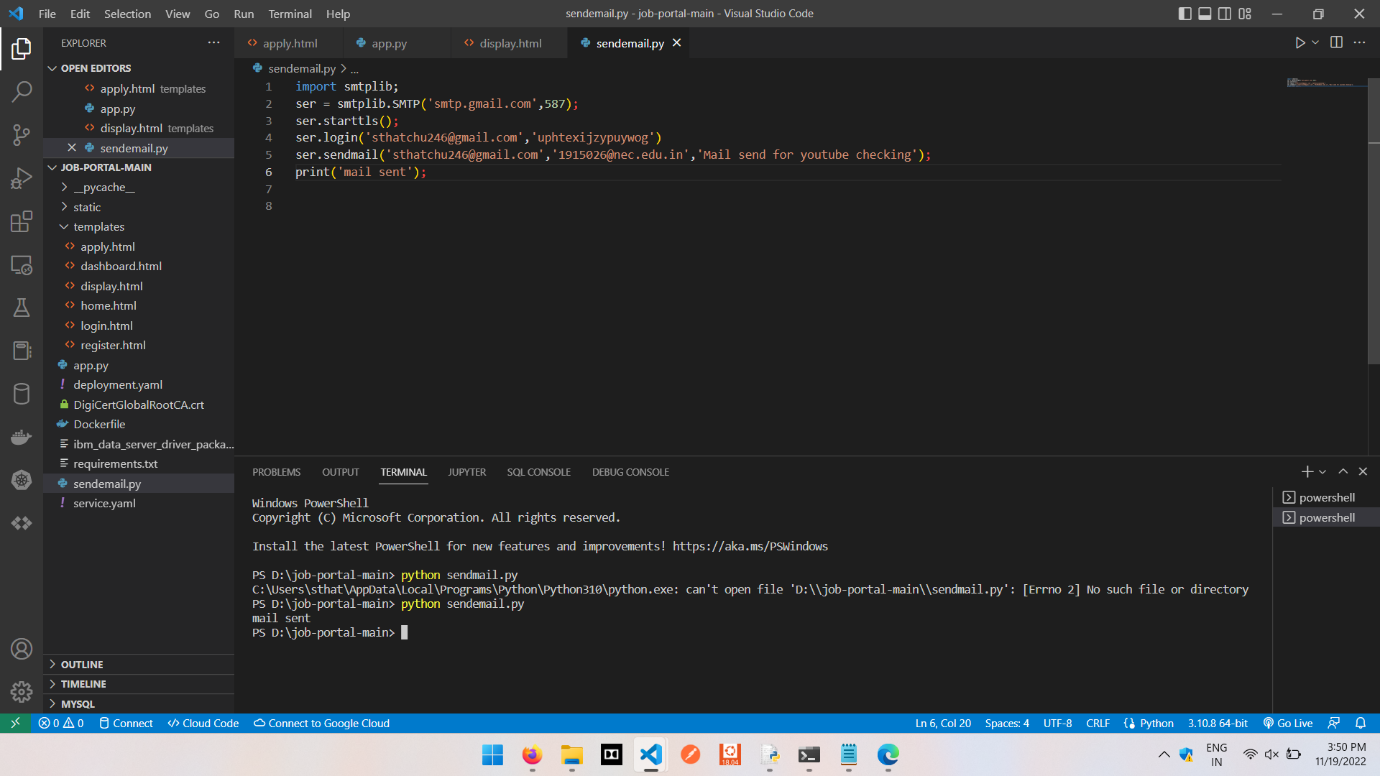
   session.pop(&#39;username&#39;, None)

   return render\_template(&#39;homeCpy.html&#39;)

if \_\_name\_\_ == &#39;\_\_main\_\_&#39;:

   app.run(host=&#39;0.0.0.0&#39;)

**7.2 FEATURE 2**



**8.TESTING**

**8.1 TEST CASES**

* User
* Existing user - check database
* New user - create new user
* Skills - Job recommendation related to skills input
* Skills - appropriate entry
* Job opening verificcation

**8.2 USER ACCEPTANCE TESTING**

User Acceptance Testing (UAT), which is performed on most UIT projects, sometimes called beta testing or end-user testing, is a phase of software development in which the software is tested in the "real world" by the intended audience or business representative. Gather the key Acceptance Criteria, Define the scope of QA involvement. Analyze product requirements and define key deliverables. Choose the time and form of end-user testing. Recruit users and form UAT team. Implement end-user testing tools and onboard testers. Create user acceptance environment and run training. Run the tests. Collect output information and analyze it.

**9.RESULTS**

**9.1 PERFORMANCE METRICS**

* Predictive Accuracy Metrics.
* Classification Accuracy Metrics.
* Rank Accuracy Metrics.
* Click-Through Rates.
* Adoption and Conversion.
* User Behavior and Engagement.
* The Customer Feedback metric.
* The Service Efficiency Metric.
* Quality, Consistency and Compliance.
* Employee Engagement.
* Customer satisfaction score.
* Average handle time.
* Mean average precision.
* Mean absolute error.

**10.ADVANTAGES AND DISADVANTAGES**

**10.1 ADVANTAGES**

* Bidirectional recommendation.
* Effective matching methods.
* Includes many attributes.
* Relational aspects are included.
* Qualitative and quantity representation (proficiency level for skills is    included).
* Use two levels in skills matching (constrains and preferences).

**10.2 DISADVANTAGES**

* Knowledge acquisition and Knowledge engineering problems.
* Tools and technologies skills excluded.
* Scalability, ramp-up, and data sparsity problems.

**11.CONCLUSION**

As a result, we draw the conclusion that a job recommendation system that analyses the job description and suggests a job based on the user's abilities and preferences qualifies as a good Recsys model for proposing open opportunities to people looking for new jobs.In light of this, we decided to model the recommender system utilising content-based filtering among the many threshold and filtering strategies.

**12.FUTURE SCOPE**

Future work in this area of employment recommendation systems has a wide range of potential applications, including:

* We may compare the results of different similarity measures to discover which one provides the most accurate response.
* We can evaluate their mean absolute error if we consider the recommender system's recommendations in comparison to actual preferences.
* By assigning them corresponding weights, we may take into account a vast variety of parameters for more accurate Content-Based recommendation.
* Hybrid recommendations  can be created by merging methods or by integrating the results of collaborative and content-based recommendations.
* Natural language processing can be used to to extract information from jobseekers resume and then recommending him the jobs.

**13.APPENDIX**

**SOURCE CODE:**

from flask import Flask, render\_template, request, redirect, url\_for, session

import ibm\_db

import re

app = Flask(\_\_name\_\_)

app.secret\_key = 'a'

conn=ibm\_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30119;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=rzg70177;PWD=xHVRjmaJ8pjbvjva",'','')

@app.route('/')

def homer():

    return render\_template('homeCpy.html')

@app.route('/login',methods =['GET', 'POST'])

def login():

    global userid

    msg = ''

    if request.method == 'POST' :

        username = request.form['username']

        password = request.form['password']

        sql = "SELECT \* FROM users WHERE username =? AND password=?"

        stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(stmt,1,username)

        ibm\_db.bind\_param(stmt,2,password)

        ibm\_db.execute(stmt)

        account = ibm\_db.fetch\_assoc(stmt)

        print (account)

        if account:

            session['loggedin'] = True

            session['id'] = account['USERNAME']

            session['mail'] = account["EMAIL"]

            userid=  account['USERNAME']

            session['username'] = account['USERNAME']

            msg = 'Logged in successfully !'

            msg = 'Logged in successfully !'

            return render\_template('dashboardCpy.html', msg = msg)

        else:

            msg = 'Incorrect username / password !'

    return render\_template('login.html', msg = msg)

@app.route('/register', methods =['GET', 'POST'])

def register():

    msg = ''

    if request.method == 'POST' :

        username = request.form['username']

        email = request.form['email']

        password = request.form['password']

        sql = "SELECT \* FROM users WHERE username =?"

        stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(stmt,1,username)

        ibm\_db.execute(stmt)

        account = ibm\_db.fetch\_assoc(stmt)

        print(account)

        if account:

            msg = 'Account already exists !'

        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

            msg = 'Invalid email address !'

        elif not re.match(r'[A-Za-z0-9]+', username):

            msg = 'name must contain only characters and numbers !'

        else:

            insert\_sql = "INSERT INTO  users VALUES (?, ?, ?)"

            prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

            ibm\_db.bind\_param(prep\_stmt, 1, username)

            ibm\_db.bind\_param(prep\_stmt, 2, email)

            ibm\_db.bind\_param(prep\_stmt, 3, password)

            ibm\_db.execute(prep\_stmt)

            msg = 'You have successfully registered !'

            return redirect(url\_for('login'))

    elif request.method == 'POST':

        msg = 'Please fill out the form !'

    return render\_template('register.html', msg = msg)

@app.route('/dashboard')

def dash():

    return render\_template('dashboardCpy.html')

@app.route('/apply',methods =['GET', 'POST'])

def apply():

    msg = ''

    if request.method == 'POST' and 'username' in request.form and 'skills' in request.form:

        username = request.form['username']

        email = request.form['email']

        qualification= request.form['qualification']

        skills = request.form['skills']

        jobs = request.form['s']

        sql = "SELECT \* FROM users WHERE username =?"

        stmt = ibm\_db.prepare(conn, sql)

        ibm\_db.bind\_param(stmt,1,username)

        ibm\_db.execute(stmt)

        account = ibm\_db.fetch\_assoc(stmt)

        print(account)

        if account:

            # msg = 'there is only  1job position! for you'

            insert\_sql = "INSERT INTO  job VALUES (?, ?, ?, ?, ?)"

            prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

            ibm\_db.bind\_param(prep\_stmt, 1, username)

            ibm\_db.bind\_param(prep\_stmt, 2, email)

            ibm\_db.bind\_param(prep\_stmt, 3, qualification)

            ibm\_db.bind\_param(prep\_stmt, 4, skills)

            ibm\_db.bind\_param(prep\_stmt, 5, jobs)

            ibm\_db.execute(prep\_stmt)

            msg = 'You have successfully applied for job !'

            return render\_template('dashboardCpy.html', msg = msg)

        # session['loggedin'] = True

        # TEXT = "Hello sandeep,a new appliaction for job position" +jobs+"is requested"

        #  #sendmail(TEXT,"sandeep@thesmartbridge.com")

        #  sendgridmail("sandeep@thesmartbridge.com",TEXT)

    elif request.method == 'POST':

        msg = 'Please fill out the form !'

    return render\_template('apply.html', msg = msg)

@app.route('/display')

def display():

    sql = "SELECT \* FROM job WHERE USERNAME = '"+session['id']+"'"

    stmt = ibm\_db.exec\_immediate(conn,sql)

    acnt = []

    # abc = ibm\_db.fetch\_row(stmt)

    # print(session['id'] + " abc : "+ abc)

        # print(abc)

    while ibm\_db.fetch\_row(stmt)!=False:

        account = dict()

        account["USERNAME"] = ibm\_db.result(stmt,"USERNAME")

        account["EMAIL"] = ibm\_db.result(stmt,"EMAIL")

        account["QUALIFICATION"] = ibm\_db.result(stmt,"QUALIFICATION")

        account["SKILLS"] = ibm\_db.result(stmt,"SKILLS")

        account["JOBS"] = ibm\_db.result(stmt,"JOBS")

        print(account)

        acnt.append(account)

        # abc = ibm\_db.fetch\_row(stmt)

    return render\_template('display.html',acnt = acnt)

@app.route('/logout')

def logout():

   session.pop('loggedin', None)

   session.pop('id', None)

   session.pop('username', None)

   return render\_template('homeCpy.html')

if \_\_name\_\_ == '\_\_main\_\_':

   app.run(host='0.0.0.0')

**GITHUB LINK:** https://github.com/IBM-EPBL/IBM-Project-33061-1660214230