

# **PROJECT REPORT**

**Team ID : PNT2022TMID03200**

**Project Name : Skill / Job Recommender Application**

## **1.INTRODUCTION**

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.

### **1.1 PROJECT OVERVIEW**

There has been a sudden boom in the technical industry and an increase in the number of good startups. 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, IBM cloud services like db2, Watson assistant , cluster, kubernetes 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

### **1.2 PURPOSE**

With an increasing number of cash-rich, stable, and promising technical companies/startups on the web which are in much demand right now, many candidates want to apply and work for these companies. They tend to miss out on these postings because there is an ocean of existing systems that list millions of jobs which are generally not relevant at all to the users. There is an abundance of choices and not much streamlining. On the basis of the actual skills or interests of an individual, job seekers often find themselves unable to find the appropriate employment for themselves. This system, therefore, approaches the idea from a data point of view, emphasizing more on the quality of the data than the quantity.

## **2.LITERATURE SURVEY**

### **2.1 EXISTING PROBLEM :**

Existing system is not very efficient , it does not benefit the user in maximum way, so the proposed system uses ibm cloud services like db2, Watson virtual assistant , cluster , kubernetes and docker for containerization of the application.

#### **1.Title : Job Recommendation through Progression of Job Selection**

Authors : Amber Nigam, Aakash Roy, Hartaran Singh, Harsimran Waila

Description : The task of job recommendation has been invariably solved using either a filter-based technique or through recommender systems where categorical features associated with jobs and candidates are used to generate recommendations. Through this paper, we are introducing a novel machine learning model which uses the candidates' job preference over time to incorporate the dynamics associated with highly volatile job market. In addition to that, our approach comprises several other smaller recommendations that contribute to problems of a) generating serendipitous recommendations b) solving the cold-start problem for new jobs and new candidates. We have used skills as embedded features to derive latent competencies from them, thereby expanding the skills of jobs and candidate to achieve more coverage in the skill domain. Our model has been developed and deployed in a real-world job recommender system and the best performance of the click-through rate metric has been achieved through a blend of machine learning and non-machine learning recommendations. The best results have been achieved through Bidirectional Long Short Term Memory Networks (Bi-LSTM) with Attention for recommending jobs through machine learning that forms a major part of our recommendation.

#### **2. Title : A Recommender System For Open Educational Videos Based On Skill Requirements**

Authors: Mohammadreza Tavakoli, Sherzod Hakimov, Ralph Ewerth, Gábor Kismihók

Description : In this paper, it suggest a novel method to help learners find relevant open educational videos to master skills demanded on the labour market. They have built a prototype, which 1) applies text classification and text mining methods on job vacancy announcements to match jobs and their required skills; 2) predicts the quality of videos; and 3) creates an open educational video recommender system to suggest personalized learning content to learners. For the first evaluation of this prototype they focused on the area of data science related jobs. Our prototype was evaluated by in-depth, semi-structured interviews. 15 subject matter experts provided feedback to assess how our recommender prototype performs in terms of its objectives, logic, and contribution to learning. More than 250 videos were recommended, and 82.8% of these recommendations were treated as useful by the interviewees. Moreover, interviews revealed that our personalized video recommender system, has the potential to improve the learning experience.

#### **3.Title: Job Recommendation based in Job Seeker Skills: An Emperical study**

Author : Jorge Valverde – Rebazza, Ricardo Puma, Paul Bustios, Nathalia C.silva

Description : In the last years, job recommender systems have become popular since they successfully reduce information overload by generating personal-ized job suggestions. Although in the literature exists a variety of techniques and strategies used as part of job recommender systems, most of them fail to recommending job vacancies that fit properly to the job seekers profiles. Thus, the contributions of this work are threefold, we: i) made publicly available a new dataset formed by a set of job seekers profiles and a set of job vacancies collected from different job search engine sites; ii) put forward the proposal of a framework for job recommendation based on professional skills of job seekers; and iii) carried out an evaluation to quantify empirically the recommendation abilities of two state-of-the-art methods, considering different configurations, within the proposed framework. We thus present a general panorama of job recommendation task aiming to facilitate research and real-world application design regarding this important issue.

#### **4.Title : A Survey of job recommender systems**

Authors : Shaha T.Al-Otaibil , Mourad Ykhlet2

Description : The Internet-based recruiting platforms become a primary recruitment channel in most companies. While such platforms decrease the recruitment time and advertisement cost, they suffer from an inappropriateness of traditional information retrieval techniques like the Boolean search methods. Consequently, a vast amount of candidates missed the opportunity of recruiting. The recommender system technology aims to help users in finding items that match their personnel interests; it has a successful usage in e-commerce applications to deal with problems related to information overload efficiently. In order to improve the erecruiting functionality, many recommender system approaches have been proposed. This article will present a survey of e-recruiting process and existing recommendation approaches for building personalized recommender systems for candidates/job matching. References :  
[1][https://www.researchgate.net/publication/272802616\\_A\\_survey\\_of\\_job\\_recommender\\_systems](https://www.researchgate.net/publication/272802616_A_survey_of_job_recommender_systems)

[2][https://www.researchgate.net/publication/325697854\\_Job\\_Recommendation\\_based\\_on\\_Job\\_Seeker\\_Skills\\_An\\_Empirical\\_Study](https://www.researchgate.net/publication/325697854_Job_Recommendation_based_on_Job_Seeker_Skills_An_Empirical_Study)

[3] <https://ieeexplore.ieee.org/document/9073723/>

[4] <https://ieeexplore.ieee.org/document/9155881/>

## **2.2 REFERENCES**

Shaha T Al-Otaibi and Mourad Ykhlef. "A survey of job recommender systems". In: International Journal of the Physical Sciences 7.29 (2012), pp. 5127–5142. issn: 19921950. doi: 10.5897/IJPS12. 482

• N Deniz, A Noyan, and O G Ertosun. "Linking Person-job Fit to Job Stress: The Mediating Effect of Perceived Person-organization Fit". In: Procedia - Social and Behavioral Sciences 207 (2015),

pp. 369– 376.

- M Diaby, E Viennet, and T Launay. "Toward the next generation of recruitment tools: An online social network-based job recommender system". In: Proc. of the 2013 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining, ASONAM 2013 (2013), pp. 821–828. doi: 10.1145/2492517.2500266.
- M Diaby and E Viennet. "Taxonomy-based job recommender systems on Facebook and LinkedIn profiles". In: Proc. of Int. Conf. on Research Challenges in Information Science (2014), pp. 1–6. issn: 21511357. doi: 10.1109/RCIS.2014.6861048.
- M Kusner et al. "From word embeddings to document distances". In: Proc. of the 32nd Int. Conf. on Machine Learning, ICML'15. 2015, pp. 957–966.
- T Mikolov et al. "Distributed Representations of Words and Phrases and Their Compositionality". In: Proc. of the 26th Int. Conf. on Neural Information Processing Systems - Volume 2. NIPS'13. Lake Tahoe, Nevada, 2013, pp. 3111–3119. url: <http://dl.acm.org/citation.cfm?id=2999792>. 2999959.
- T Mikolov et al. "Efficient estimation of word representations in vector space". In: arXiv preprint arXiv:1301.3781 (2013).
- G Salton and C Buckley. "Term-weighting approaches in automatic text retrieval". In: Information Processing and Management 24.5 (1988), pp. 513–523. issn: 0306-4573. doi: [https://doi.org/10.1016/0306-4573\(88\)90021-0](https://doi.org/10.1016/0306-4573(88)90021-0). url: <http://www.sciencedirect.com/science/article/pii/030645738890021>

## 2.3 PROBLEM STATEMENT DEFINITION

"Can an efficient recommender system be modeled for the Job seekers which recommend Jobs with the user's skill set and job domain and also addresses the issue of cold start?".

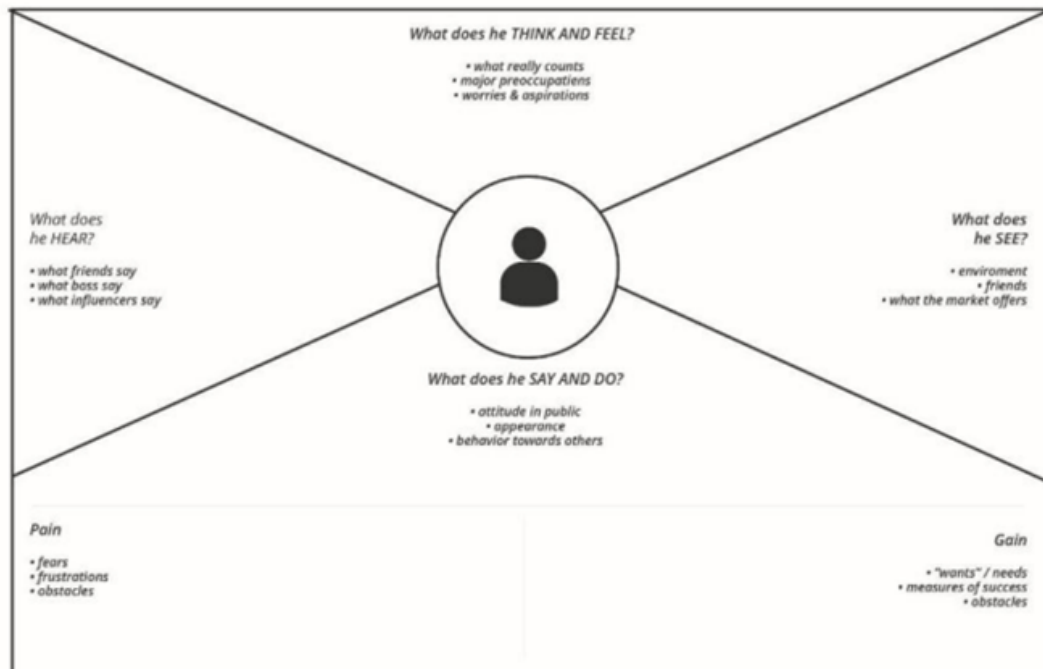
In current situation recruitment s done manually for lakhs of students in which many talented students may lose their opportunities due to different reasons since it is done manually, and company also need the highly talented people from the mass group for their growth. So we have build a cloud application to do this process in a efficient manner.

### 3. IDEATION AND PROPOSED SOLUTION

#### 3.1 EMPATHY MAP :

An empathy map is a collaborative visualization used to articulate what we know about a particular type of user. It externalizes knowledge about users in order to

- 1) Create a shared understanding of user needs, and
- 2) Aid in decision making



#### 3.2 IDEATION AND BRAINSTROMING


##### Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

## STEP 1:

### Team Gathering, Collaboration and Select the Problem Statement

Template



## Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

⌚ 10 minutes to prepare  
👥 1 hour to collaborate  
👤 2-8 people recommended

**Before you collaborate**

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

⌚ 10 minutes

**A Team gathering**

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

**B Set the goal**

Think about the problem you'll be focusing on solving in the brainstorming session.

**C Learn how to use the facilitation tools**

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

**1 Define your problem statement**

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

⌚ 5 minutes

**PROBLEM**

The user needs a better way to find a job. They need to be able to find a job that is relevant to their skills, experience, and interests. They need to be able to find a job that is challenging and offers growth opportunities.

**Key rules of brainstorming**

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

## STEP 2:

### Brainstorm, Idea Listing and Grouping

**2 Brainstorm**

Write down any ideas that come to mind that address your problem statement.

⌚ 10 minutes

**TIP**

You can select a sticky note and in the panel switch to sketch? (don't start drawing!)

**Shyam Sundar S**

- Obstacle in searching job
- User friendly environment
- Optimal Search Engine
- Communication between user and employer

**Sham Kumar J**

- The suggestions results are Responsive
- Solving the queries
- Skill based domain
- Geo expand results

**Nitish Kumar R**

- Accurate job search results
- Description of job details
- Desired Salary
- Connecting people socially

**Aashish**

- Identifying roles
- Jobs for non-technical domain
- Providing detailed information of user
- Time management

**Kimuteja Reddy**

- Referral
- Communicating with people from other organizations
- A place for all jobs
- Helping people to form connections

**3 Group Ideas**

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

⌚ 20 minutes

**Input**

- Obstacle in searching job
- Accurate job search results
- Optimal Search Engine
- Geo expand results

**Job Results**

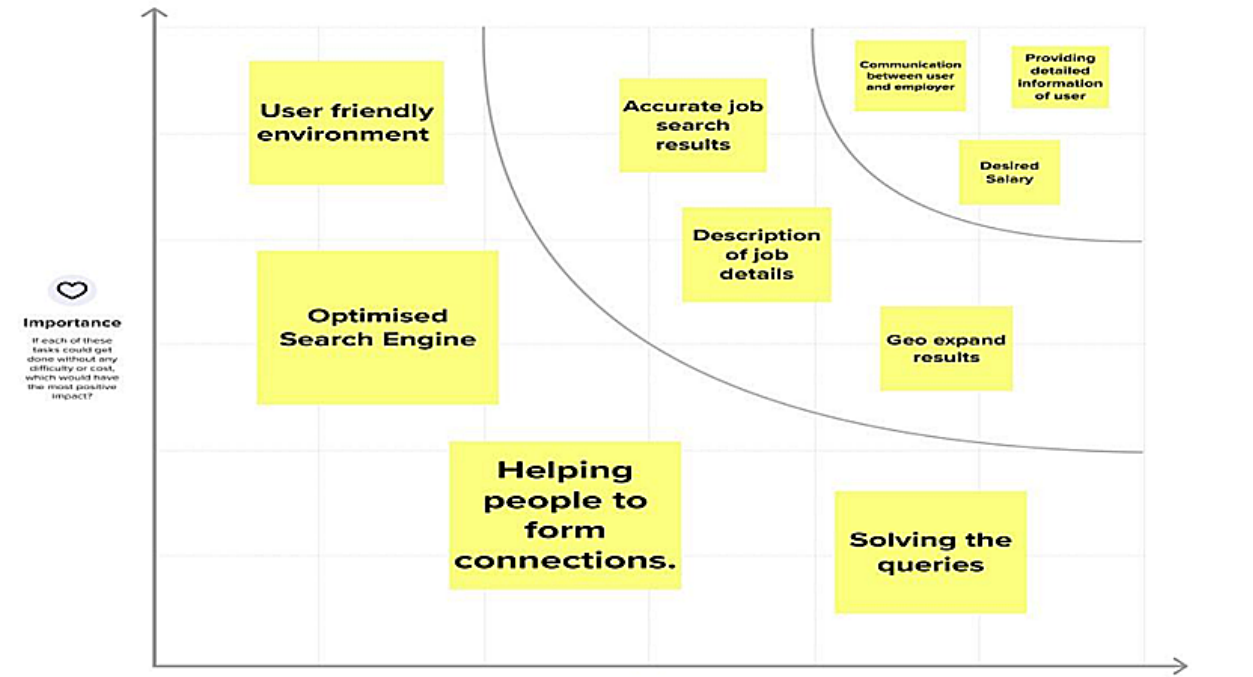
- The suggestions results are Responsive
- Decision of job details
- Jobs for non-technical domain
- Desired Salary

**Communication**

- Communicating with people from other organizations
- Helping people to form connections
- Communication between user and employer
- Providing detailed information of user

### STEP 3:

#### Idea Prioritization



### 3.3 PROPOSED SOLUTION

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

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

### 3.4 PROBLEM SOLUTION FIT

Define CS, fit in CC	<b>1. CUSTOMER SEGMENT(S)</b> Who is your customer? <span>CS</span> <p>Customers who are not able to solve their own Problem and in need for a possible solution from their agents/job providers.</p>	<b>6. CUSTOMER CONSTRAINT.</b> What constraint prevents your customer from taking action or limiting their choice of solution? <span>CC</span> <p>The problem of contacting the agent and all the problems and procedure in it.</p>	<b>5. AVAILABLE SOLUTION</b> Which solutions are available to the customer when they face the problem. <span>AS</span> <ul style="list-style-type: none"> <li>They can check FAQ's Session for fast support.</li> <li>If the problem is not listed, they can post the problem in new queries section.</li> <li>Which will be further assisted by the agent team.</li> </ul>	Explore AS, Differentiate
Focus on J&P, Tap into BE, Understand RC	<b>2. JOBS-TO-BE-DONE/PROBLEMS</b> Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; Explore different sides? <span>J&amp;P</span> <ul style="list-style-type: none"> <li>This Application Allows Customers to get recommended job according to their skillset</li> <li>They will be able post their resume and wait for the solution.</li> <li>They will also get solutions to their queries</li> <li>They can also access our FAQ's Section on our website.</li> </ul>	<b>9. PROBLEM ROOT CAUSE.</b> What is the real reason that the problem exists? <span>RC</span> <p>The only real reason that this problem exists is the lack of awareness and ratio of proven results which could create trust issues with their agent.</p>	<b>7. BEHAVIOR</b> What does your customer do to address the problem and get the job done. <span>BE</span> <ul style="list-style-type: none"> <li>They must first Post their resume and then wait for 2 hours.</li> <li>They can also use our chatbot to easily contact our Team.</li> <li>They can also refer the FAQ's session.</li> </ul>	Focus on J&P, Tap into BE, Understand RC
Identify string TR & ME	<b>3. TRIGGERS</b> What triggers customers to act. <span>ER</span> <ul style="list-style-type: none"> <li>Customers get to know the absolute recommendation to their need.</li> <li>Fast Response.</li> </ul>	<b>10. YOUR SOLUTION</b> Our solution involves autonomous system which does the following: <span>RC</span> <ul style="list-style-type: none"> <li>A personal Help desk which can be accessed through all the devices which are compatible with browser.</li> <li>Customers can post their queries in the new thread section.</li> <li>They can also access the FAQ's Section to see if the problem is already listed</li> <li>They can also view their results progress through their mails.</li> <li>They will get support from the team until the problem gets resolved.</li> </ul>	<b>8. CHANNELS of BEHAVIOR</b> <b>ONLINE</b> <span>CH</span> <ul style="list-style-type: none"> <li>For a new query they need an online connectivity to post and receive recommendation from our team.</li> <li>They can also use our chatbot 24/7 While they are in online.</li> </ul>	Identify string TR & ME
	<b>4. EMOTIONS: BEFORE/AFTER</b> How do customers feel when they face a problem or a job and afterwards. <span>TM</span> <ul style="list-style-type: none"> <li>Enables Customers to Trust to their agent about posting their personal informations.</li> <li>Feeling comfortable with the solution and the company's service.</li> </ul>		<b>OFFLINE</b> <ul style="list-style-type: none"> <li>They can Read the messages once it is received through the cloud app.</li> <li>They can access FAQ's while they are offline.</li> </ul>	



## 4. REQUIREMENT ANALYSIS

### 4.1 FUNCTIONAL REQUIREMENT :

Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
User Registration	Registration through Form Registration through Gmail
User Confirmation	Confirmation via Email Confirmation via OTP
Chat Bot	A Chat Bot will be there in website to solve user queries and problems related to applying a job, search for a job and much more.
User Login	Login through Form Login through Gmail
User Search	Exploration of Jobs based on job filters and skill recommendations.
User Profile	Updation of the user profile through the login credentials
User Acceptance	Confirmation of the Job.

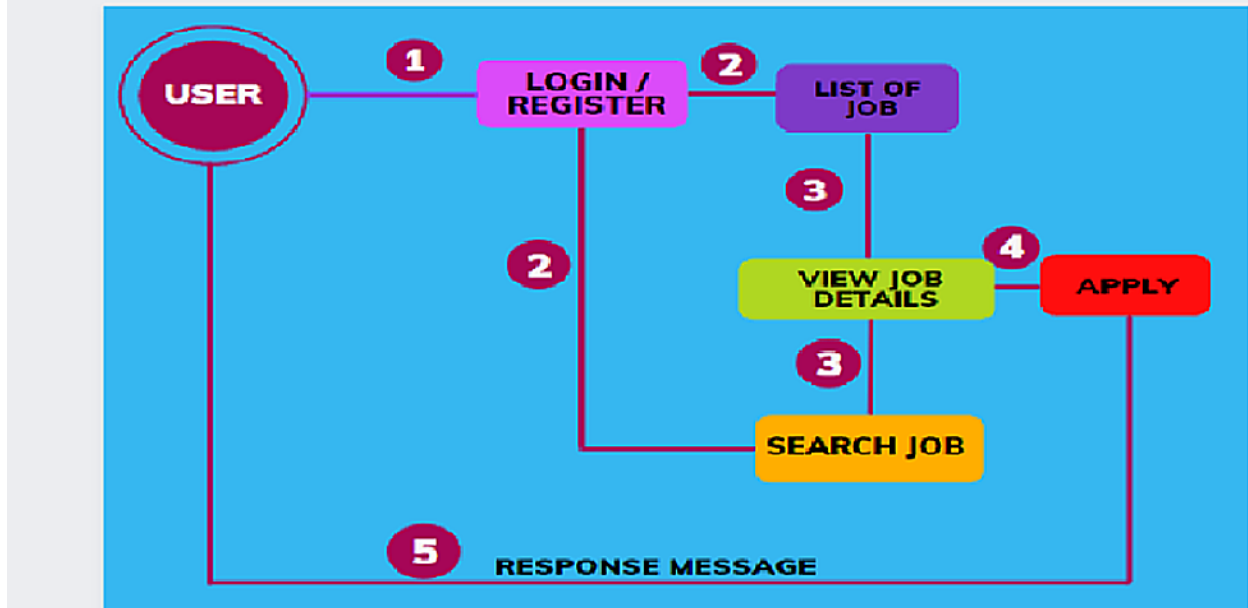
### 4.2 NON FUNCTIONAL REQUIREMENTS

Non functional Requirements are :

1. Usability
2. Security
3. Reliability
4. Performance
5. Availability
6. Scalability

## 5 PROJECT DESIGN

### 5.1 DATAFLOW DIAGRAM :



### 5.2 TECHNICAL ARCHITECTURE :

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed and delivered.
- Provide the best business require recommend by using the optimised and efficient algorithm
- Differentiate the fake job recommend by fake sites and be aware from the Scammers

[illegible]

I am  
a jobseeker

I'm trying to  
find a job

But  
I am unable to  
find the jobs that  
fits my skillset

Because  
There is no platform to  
find and apply for a job  
that recommends me  
based on my skills

Which makes me feel  
less confident  
and lose hope.

## 6 PROJECT PLANNING AND SCHEDULING

### 6.1 SPRINT PLANNING AND EXSTIMATION :

Title	Description
Information Gathering Literature Survey	Referring to the research publications & technical papers, etc.
Create Empathy Map	Preparing the List of Problem Statements and to capture user pain and gains.
Ideation	Prioritise a top ideas based on feasibility and Importance.
Proposed Solution	Solutions including feasibility, novelty, social impact, business model and scalability of solutions.
Problem Solution Fit	Solution fit document.
Solution Architecture	Solution Architecture.
Customer Journey	To Understand User Interactions and experiences with application.
Functional Requirement	Prepare functional Requirement.
Data flow Diagrams	Data flow diagram.
Technology Architecture	Technology Architecture diagram.
Milestone & sprint delivery plan	Activities are done & further plans.
Project Development Delivery of sprint 1,2,3 & 4	Develop and submit the developed code by testing it.

## 6.2 SPRINT DELIVERY SCHEDULE :

SPRINT	TASK	MEMBERS
SPRINT 1	Create Registration page , login page , Job search portal , job apply portal in flask	Surya V Srivishnu M
SPRINT 2	Connect application to ibm db2	Srivishnu M Surya V Sunil M Sudarsan Perumal V
SPRINT 3	Integrate ibm Watson assisstant	Surya V Srivishnu M
SPRINT 4	Containerize the app and Deploy the application in ibm cloud	Srivishnu M Surya V Sunil M Sudarsan Perumal V

## 6.3 REPORTS FROM JIRA :

Average Age Report.

Created vs Resolved Issues Report.

Pie Chart Report.

Recently Created Issues Report.

Resolution Time Report.

Single Level Group By Report.

Time Since Issues Report.

Time Tracking Report.

Welcome to

IBM

IBM

IBM-Project

Collaboration

Administrati

Skills and Joi

(3) WhatsApp

team-16686883124956.atlassian.net/jira/software/projects/SJR/boards/1/backlog

Jira Software

Your work

Projects

Filters

Dashboards

People

Apps

Create

Q Search

?

⚙

👤

Skills and Job recomm...

Software project

PLANNING

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Projects / Skills and Job recommender

Backlog

...

Q

👤👤👤👤

Epic

Insights

▼ SJR Sprint 1

Add dates

(1 issue)

000

Start sprint

...

SJR-3 Homepage(Login,Signup)

TO DO

👤

+ Create issue

▼ SJR Sprint 2

Add dates

(1 issue)

000

Start sprint

...

SJR-4 Application Features

TO DO

👤

+ Create issue

▼ SJR Sprint 3

Add dates

(1 issue)

000

Start sprint

...

SJR-5 Additional Features

TO DO

👤

+ Create issue

## 7.CODING & SOLUTIONING

### 7.1 Feature 1:

#### App Market

This is one of the feature of our application Skill Pal which provides companies job details for end users

```
@app.route('/jobmarket')
def jobmarket():
    jobids = []
    jobnames = []
    jobimages = []
    jobdescription = []

    sql = "SELECT * FROM JOBMARKET"
    stmt = ibm_db.prepare(conn, sql)
    username = session['username']
    print(username)
    #ibm_db.bind_param(stmt,1,username)
    ibm_db.execute(stmt)
    joblist = ibm_db.fetch_tuple(stmt)
    print(joblist)
    while joblist != False:
        jobids.append(joblist[0])
        jobnames.append(joblist[1])
        jobimages.append(joblist[2])
        jobdescription.append(joblist[3])
        joblist = ibm_db.fetch_tuple(stmt)

    jobinformation = []

    cols = 4
    size = len(jobnames)
    for i in range(size):
        col = []
        col.append(jobids[i])
        col.append(jobnames[i])
        col.append(jobimages[i])
        col.append(jobdescription[i])
        jobinformation.append(col)
    print(jobinformation)
```

```
return render_template('jobmarket.html', jobinformation = jobinformation)
```

```
@app.route('/filterjobs')
```

```
def filterjobs():
```

```
    skill1 = ""
```

```
    skill2 = ""
```

```
    skill3 = ""
```

```
    user = session['username']
```

```
    sql = "SELECT * FROM ACCOUNTSKILL WHERE USERNAME = ?"
```

```
    stmt = ibm_db.prepare(conn, sql)
```

```
    ibm_db.bind_param(stmt,1,user)
```

```
    ibm_db.execute(stmt)
```

```
    skillres = ibm_db.fetch_assoc(stmt)
```

```
    if skillres:
```

```
        skill1 = skillres['SKILL1']
```

```
        skill2 = skillres['SKILL2']
```

```
        skill3 = skillres['SKILL3']
```

```
        print(skillres)
```

```
        jobids = []
```

```
        jobnames = []
```

```
        jobimages = []
```

```
        jobdescription = []
```

```
    sql = "SELECT * FROM JOBMARKET"
```

```
    stmt = ibm_db.prepare(conn, sql)
```

```
    username = session['username']
```

```
    print(username)
```

```
    #ibm_db.bind_param(stmt,1,username)
```

```
    ibm_db.execute(stmt)
```

```
    joblist = ibm_db.fetch_tuple(stmt)
```

```
    print(joblist)
```

```
    while joblist != False:
```

```
        jobids.append(joblist[0])
```

```
        jobnames.append(joblist[1])
```

```
        jobimages.append(joblist[2])
```

```
        jobdescription.append(joblist[3])
```

```
        joblist = ibm_db.fetch_tuple(stmt)
```

```
    jobinformation = []
```





### **7.3 Database :**

We use IBM DB2 for our database, below are the tables we used with the parameters given.



IBM Db2 on Cloud

Load DataLoad HistoryTablesViewsIndexesAliasesMQTsSequencesApplication objects

Find schemas or tables

Refresh

Schemas

<input checked="" type="checkbox"/>	Name	Type	Tables
<input checked="" type="checkbox"/>	JDD83131	User	6

Total: 1, selected: 1

Tables

New table +

<input type="checkbox"/>	Name	Schema	Properties
<input type="checkbox"/>	ACCOUNT	JDD83131	...
<input type="checkbox"/>	ACCOUNTSKILL	JDD83131	...
<input type="checkbox"/>	APPLIEDJOBS	JDD83131	...
<input type="checkbox"/>	CUSTOMER	JDD83131	...
<input type="checkbox"/>	JOBMARKET	JDD83131	...
<input type="checkbox"/>	STUDENTS	JDD83131	...

Total: 6, selected: 0

## **8.TESTING**

### **8.1 Test Cases:**

We tested for various validations. Tested all the features with using all the functionalities. Tested the data base storage and retrieval feature too.

Testing was done in phase 1 and phase 2, where issues found in phase1 were fixed and then tested again in phase2.

### **8.2 User Acceptance Testing:**

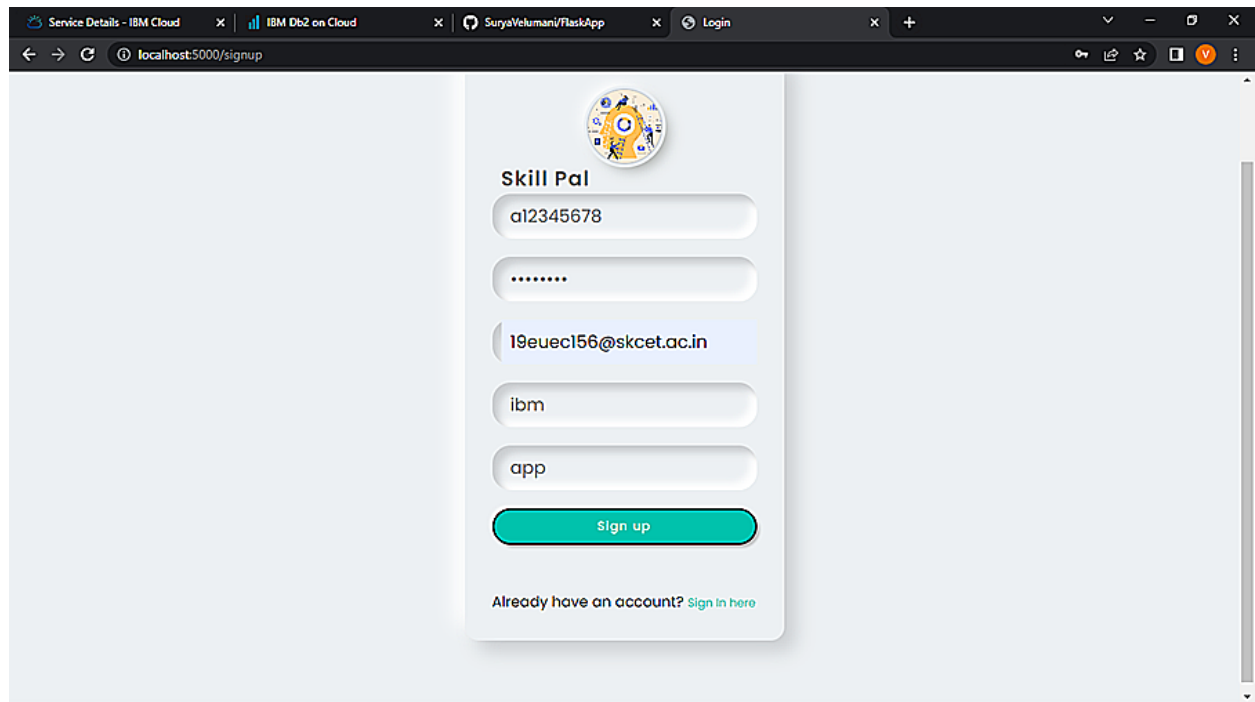
Real world testing was also done, by giving to remote users and asking them to use the application. Their difficulties were fixed and tested again until all the issues were fixed.

## 9.RESULTS

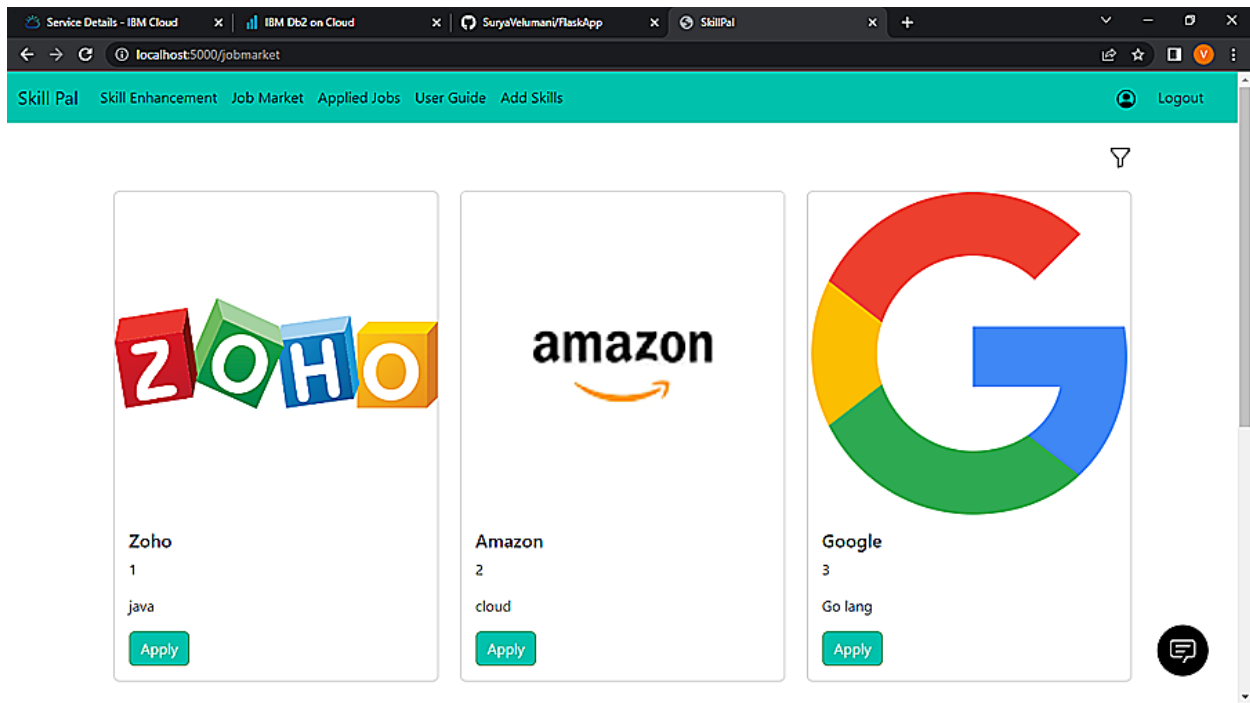
### 9.1 Perfomance Metrics:

Sign In Page :

Sign Up page with validations :



Job Market Page (Filter Option Available) :



## Job Application :

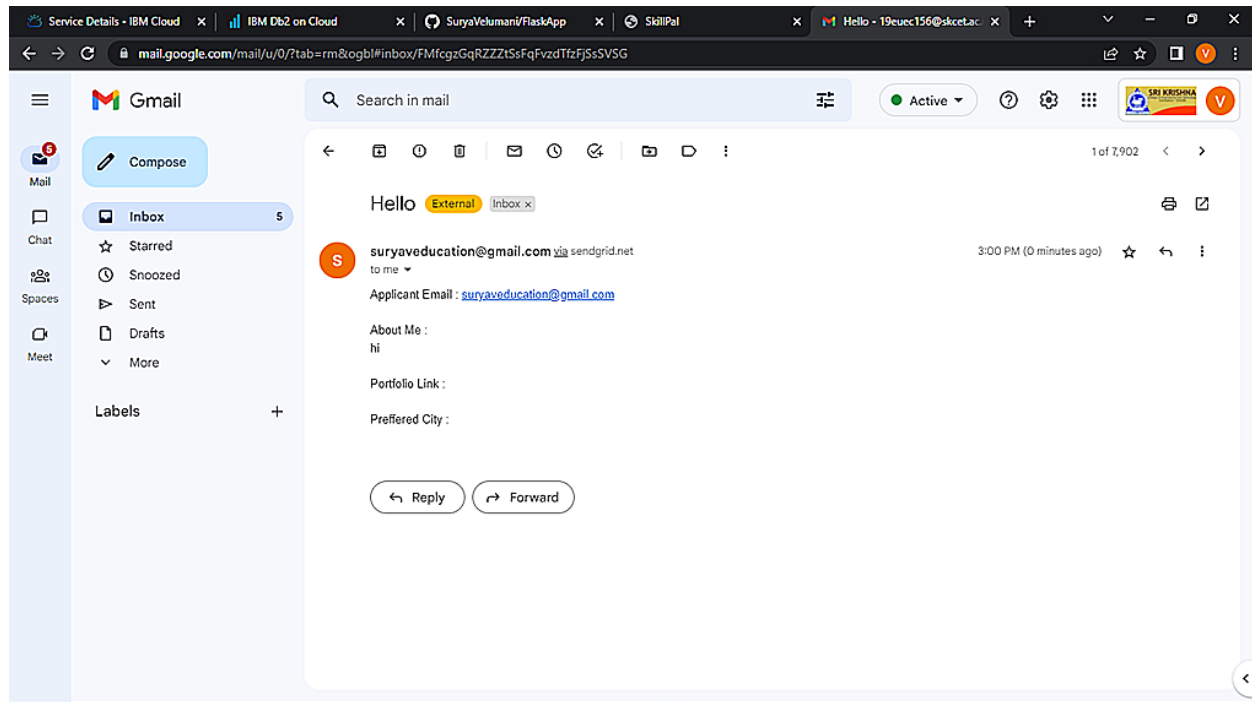
The screenshot shows the 'Job Application' form on the SkillPal website. The browser tabs and header are the same as in the previous screenshot. The address bar shows 'localhost:5000/jobapplied/5'. The form contains the following fields:

- Please tell about yourself, and why you need this job :** A text area containing the text: 'Hi, I am highly skilled in java, so I am interested in applying for this job.'
- 5** A text input field.
- Company :** A text input field containing 'Zoho'.
- Company Email :** A text input field containing '19euec156@skcet.ac.in'.
- Portfolio Link :** A text input field containing 'www.ram.com'.
- Preferred Location :** A text input field containing 'CBE'.
- Submit form** A green button.

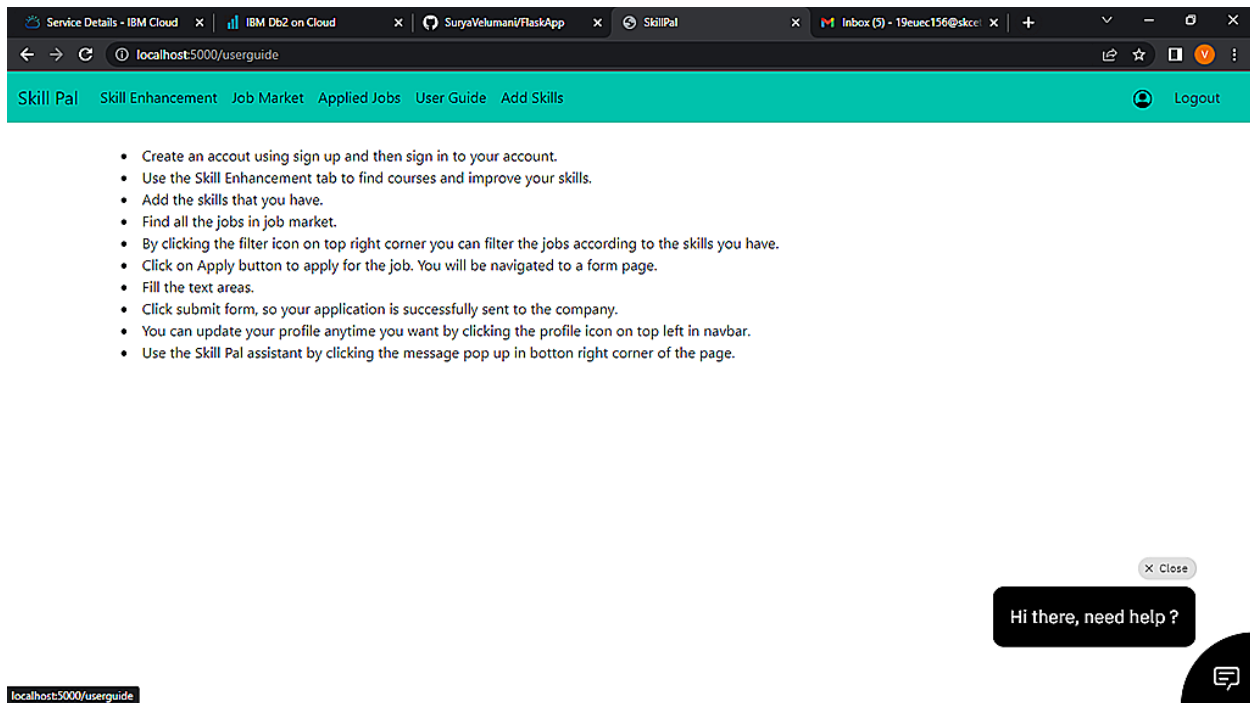
A chat bubble in the bottom right corner says 'Hi there, need help ?' with a 'Close' button.



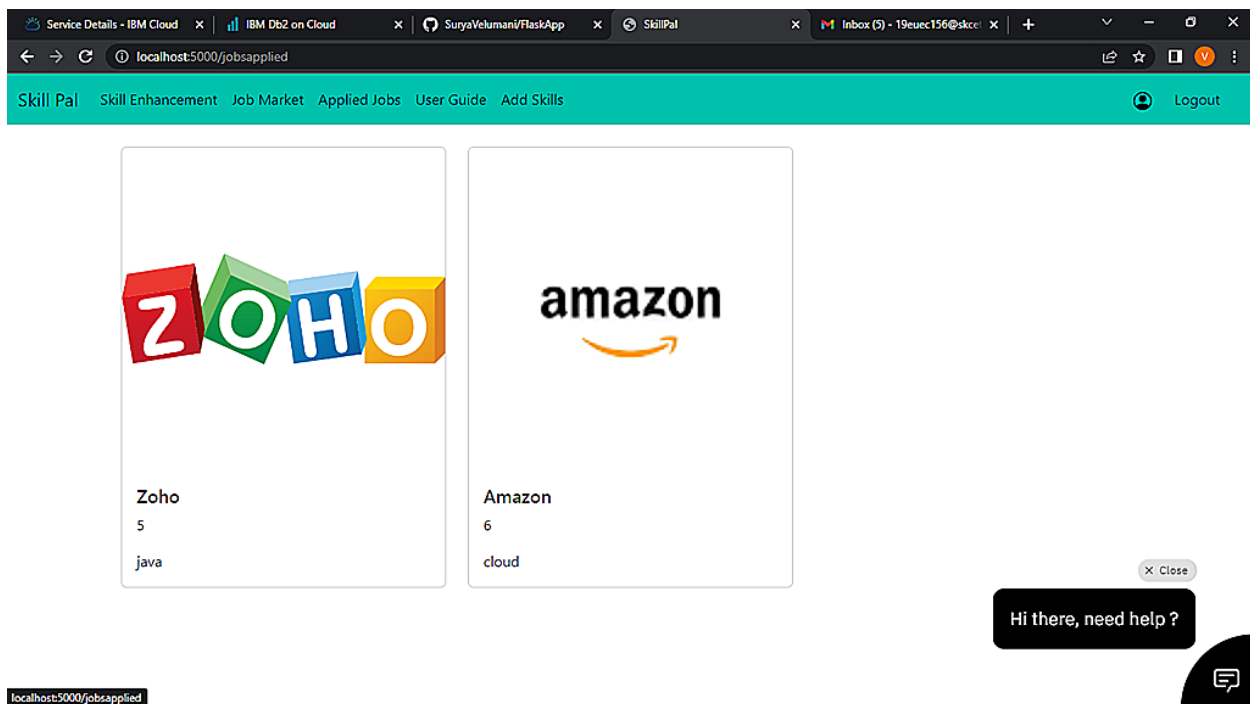
## Email On Registration :



## User Guide :



## Applied Jobs :



## Send Grid :



## Edit Profile :

Service Details - IBM Cloud x IBM Db2 on Cloud x SuryaVelumani/FlaskApp x SkillPal x SendGrid x

localhost:5000/profile

Skill Pal Skill Enhancement Job Market Applied Jobs User Guide Add Skills Logout

User Name :  
c12345678

Password :  
\*\*\*\*\*

Email Id :  
suryaveducation@gmail.com

First Name :  
Surya

Last Name :  
V

Save

Chatbot icon

## Chatbot :

Service Details - IBM Cloud x IBM Db2 on Cloud x SuryaVelumani/FlaskApp x SkillPal x Inbox (3) - suryaveducation@gmail.com

localhost:5000/jobsapplied

Skill Pal Skill Enhancement Job Market Applied Jobs User Guide Add Skills Logout

Zoho  
5  
java

Amazon  
6  
cloud

Skill Pal

Hi

May I know your username first.

IBM

How can I help you ?

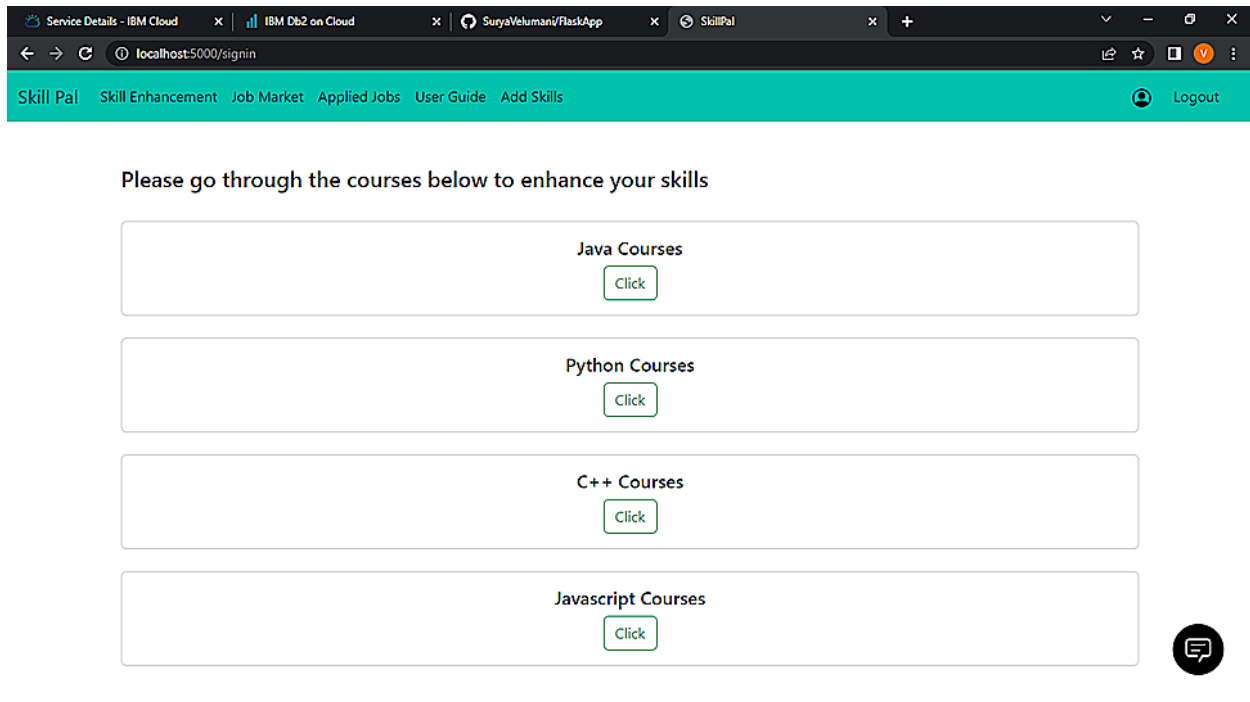
Search For Job Look For Jobs You Applied

Profile Issues Other Issues

Type something...

Built with IBM Watson®

## Skill Enhancement :



The screenshot shows a web browser with multiple tabs. The active tab is 'SkillPal' at 'localhost:5000/signin'. The page has a teal header with navigation links: 'Skill Pal', 'Skill Enhancement', 'Job Market', 'Applied Jobs', 'User Guide', and 'Add Skills'. A 'Logout' button is in the top right. The main content area has the heading 'Please go through the courses below to enhance your skills'. Below this are four white boxes, each with a title and a 'Click' button: 'Java Courses', 'Python Courses', 'C++ Courses', and 'Javascript Courses'. A chat icon is in the bottom right corner.

Service Details - IBM Cloud x IBM Db2 on Cloud x SuryaVelumani/FlaskApp x SkillPal x +

localhost:5000/signin

Skill Pal Skill Enhancement Job Market Applied Jobs User Guide Add Skills Logout

Please go through the courses below to enhance your skills

Java Courses  
Click

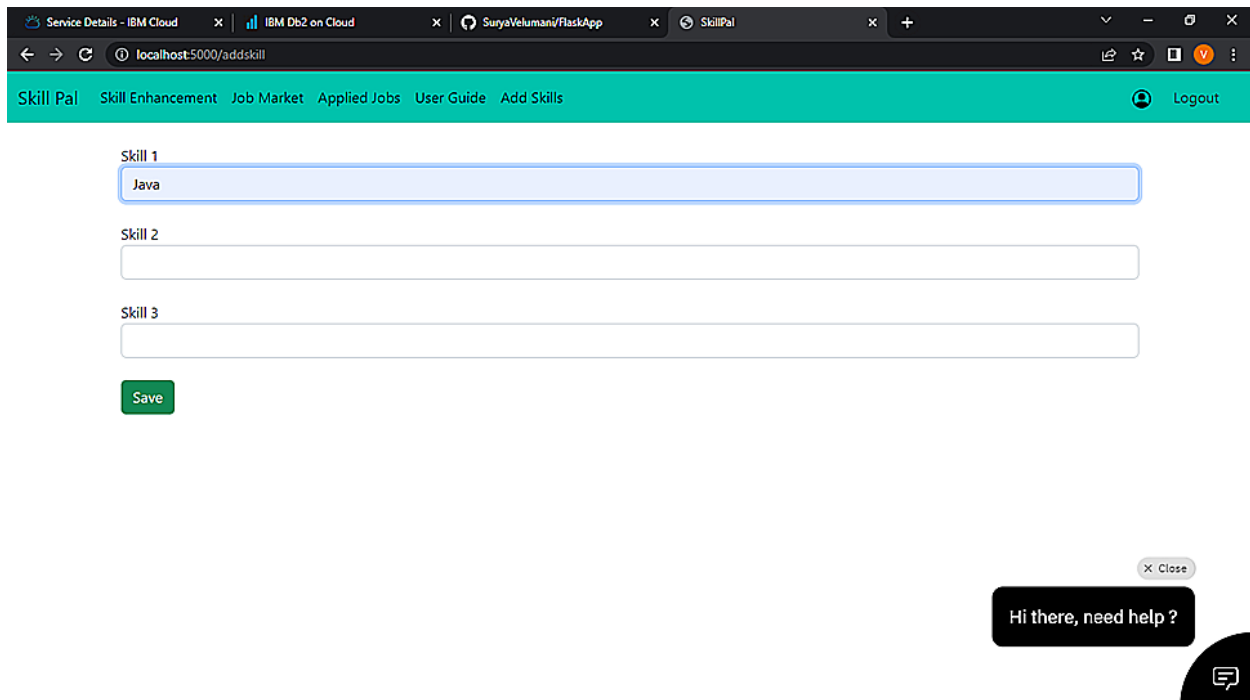
Python Courses  
Click

C++ Courses  
Click

Javascript Courses  
Click

Chat icon

## Add Skills :



The screenshot shows the 'SkillPal' web application at 'localhost:5000/addskill'. The header is the same as the previous page. The main content area has three input fields labeled 'Skill 1', 'Skill 2', and 'Skill 3'. The first field contains the text 'Java'. Below the fields is a green 'Save' button. A chat icon is in the bottom right corner. A dark chat bubble with the text 'Hi there, need help ?' and a 'Close' button is also visible.

Service Details - IBM Cloud x IBM Db2 on Cloud x SuryaVelumani/FlaskApp x SkillPal x +

localhost:5000/addskill

Skill Pal Skill Enhancement Job Market Applied Jobs User Guide Add Skills Logout

Skill 1  
Java

Skill 2

Skill 3

Save

Close

Hi there, need help ?

Chat icon

## **10. ADVANTAGE AND DISADVANTAGE**

### **ADVANTAGE :**

It helps candidates to search the job which perfectly suites them and make them aware of all the job openings.

It help recruiters of the company to choose the right candidates for their organisations with appropriate skills.

Since it is cloud application , it does require any installation of softwares and is portable.

### **DISADVANTAGE:**

It is costly.

Uninterrupted internet connection is required for smooth functioning of application.

## **11. CONCLUSION**

We have used ibm cloud services like db2, cloud registry , kubernetes , Watson assistant to create this application , which will be very usefull for candidates who are searching for job and as well as for the company to select the right candidate for their organization. This app will help people find jobs that could match their skills.

## **12. FUTURE SCOPE**

Future directions of our work will focus on performing a more exhaustive evaluation considering a greater amount of methods and data as well as a comprehensive evaluation of the impact of each professional skill of a job seeker on the received job recommendation. We can use machine learning technicques to recommend data in a efficient way.

## 13.APPENDIX

Source Code:

```
from turtle import st
from flask import Flask, render_template, request, redirect, url_for, session

import ibm_db
conn =
from flask_mail import Mail, Message

import ibm_boto3
from ibm_botocore.client import Config, ClientError

COS_ENDPOINT=
COS_API_KEY_ID=
COS_INSTANCE_CRN=

# Create resource https://s3.ap.cloud-object-storage.appdomain.cloud
cos = ibm_boto3.resource("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_INSTANCE_CRN,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)

app = Flask(_name_)

def multi_part_upload(bucket_name, item_name, file_path):
    try:
        print("Starting file transfer for {0} to bucket: {1}\n".format(item_name, bucket_name))
        # set 5 MB chunks
        part_size = 1024 * 1024 * 5

        # set threadhold to 15 MB
        file_threshold = 1024 * 1024 * 15
```

```

# set the transfer threshold and chunk size
transfer_config = ibm_boto3.s3.transfer.TransferConfig(
    multipart_threshold=file_threshold,
    multipart_chunksize=part_size
)

# the upload_fileobj method will automatically execute a multi-part upload
# in 5 MB chunks for all files over 15 MB
with open(file_path, "rb") as file_data:
    cos.Object(bucket_name, item_name).upload_fileobj(
        Fileobj=file_data,
        Config=transfer_config
    )

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

@app.route('/uploadResume', methods = ['GET', 'POST'])
def upload():
    if request.method == 'POST':
        bucket='sv-demoibm1'
        name_file = session['username']
        name_file += '.png'
        filenameis = request.files['file']
        filepath = request.form['filepath']
        f = filepath
        f = f+filenameis.filename
        print("-----",f)
        multi_part_upload(bucket,name_file,f)
        return redirect(url_for('dashboard'))

    if request.method == 'GET':
        return render_template('upload.html')

mail = Mail(app) # instantiate the mail class

```



```

app.config['MAIL_SERVER']='smtp.sendgrid.net'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)

@app.route('/')
def home():
    return redirect(url_for('signin'))

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

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

@app.route('/addskill')
def addskill():
    skill1 = ""
    skill2 = ""
    skill3 = ""
    user = session['username']
    sql = "SELECT * FROM ACCOUNTSKILL WHERE USERNAME = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,user)
    ibm_db.execute(stmt)
    skillres = ibm_db.fetch_assoc(stmt)
    if skillres:
        skill1 = skillres['SKILL1']
        skill2 = skillres['SKILL2']
        skill3 = skillres['SKILL3']
        print(skillres)
        return render_template('addSkill.html', skill1=skill1,skill2=skill2,skill3=skill3)
    else :
        return render_template('addSkill.html', skill1=skill1,skill2=skill2,skill3=skill3)

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

```

```

username = session['username']
sql = "SELECT * FROM ACCOUNTSKILL WHERE USERNAME = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,username)
ibm_db.execute(stmt)
skillres = ibm_db.fetch_assoc(stmt)
if skillres:
    msg = ""
    skill11 = request.form['skill1']
    skill21 = request.form['skill2']
    skill31 = request.form['skill3']
    print(skill11,"---",skill21,"--",skill31)
    sql = "UPDATE ACCOUNTSKILL SET SKILL1 = ?, SKILL2 = ?, SKILL3 = ? WHERE USERNAME
= ?;"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,skill11)
    ibm_db.bind_param(stmt,2,skill21)
    ibm_db.bind_param(stmt,3,skill31)
    ibm_db.bind_param(stmt,4,username)
    print(":",sql)
    ibm_db.execute(stmt)
    msg = "Saved Successfully !"
    return render_template('addSkill.html',msg = msg, skill1=skill11,skill2=skill21,skill3=skill31)
else :
    msg = ""
    skill12 = request.form['skill1']
    skill22 = request.form['skill2']
    skill32 = request.form['skill3']
    print("-----",username)
    sql = "INSERT INTO ACCOUNTSKILL VALUES (?,?,?)"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,username)
    ibm_db.bind_param(stmt,2,skill12)
    ibm_db.bind_param(stmt,3,skill22)
    ibm_db.bind_param(stmt,4,skill32)
    print(":",sql)
    ibm_db.execute(stmt)
    msg = "Saved Successfully !"
    return render_template('addSkill.html',msg = msg, skill1=skill12,skill2=skill22,skill3=skill32)

```

```
@app.route('/jobmarket')
```

```

def jobmarket():
    jobids = []
    jobnames = []
    jobimages = []
    jobdescription = []

    sql = "SELECT * FROM JOBMARKET"
    stmt = ibm_db.prepare(conn, sql)
    username = session['username']
    print(username)
    #ibm_db.bind_param(stmt,1,username)
    ibm_db.execute(stmt)
    joblist = ibm_db.fetch_tuple(stmt)
    print(joblist)
    while joblist != False:
        jobids.append(joblist[0])
        jobnames.append(joblist[1])
        jobimages.append(joblist[2])
        jobdescription.append(joblist[3])
        joblist = ibm_db.fetch_tuple(stmt)

    jobinformation = []

    cols = 4
    size = len(jobnames)
    for i in range(size):
        col = []
        col.append(jobids[i])
        col.append(jobnames[i])
        col.append(jobimages[i])
        col.append(jobdescription[i])
        jobinformation.append(col)
    print(jobinformation)

    return render_template('jobmarket.html', jobinformation = jobinformation)

@app.route('/filterjobs')
def filterjobs():
    skill1 = ""
    skill2 = ""
    skill3 = ""

```

```
user = session['username']  
sql = "SELECT * FROM ACCOUNTSKILL WHERE USERNAME = ?"  
stmt = ibm_db.prepare(conn, sql)  
ibm_db.bind_param(stmt,1,user)  
ibm_db.execute(stmt)  
skillsres = ibm_db.fetch_assoc(stmt)  
if skillsres:  
    skill1 = skillsres['SKILL1']  
    skill2 = skillsres['SKILL2']  
    skill3 = skillsres['SKILL3']  
    print(skillsres)  
    jobids = []  
    jobnames = []  
    jobimages = []  
    jobdescription = []  
  
    sql = "SELECT * FROM JOBMARKET"  
    stmt = ibm_db.prepare(conn, sql)  
    username = session['username']  
    print(username)  
    #ibm_db.bind_param(stmt,1,username)  
    ibm_db.execute(stmt)  
    joblist = ibm_db.fetch_tuple(stmt)  
    print(joblist)  
    while joblist != False:  
        jobids.append(joblist[0])  
        jobnames.append(joblist[1])  
        jobimages.append(joblist[2])  
        jobdescription.append(joblist[3])  
        joblist = ibm_db.fetch_tuple(stmt)  
  
    jobinformation = []  
  
    cols = 4  
    size = len(jobnames)  
    print("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$4",skill1,skill2,skill3)  
  
    for i in range(size):  
        col = []
```

```

@@@@@@@@@@@@@@@@,jobdescription[i])
    if jobdescription[i].lower() == skill1.lower() or jobdescription[i].lower() == skill2.lower() or
jobdescription[i].lower() == skill3.lower() :
        col.append(jobids[i])
        col.append(jobnames[i])
        col.append(jobimages[i])
        col.append(jobdescription[i])
        jobinformation.append(col)

```

```

print("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@",jobinformation)

```

```

    return render_template('jobmarket.html', jobinformation = jobinformation)

```

```

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

```

```

def signin():

```

```

    msg = "

```

```

    if request.method == 'POST':

```

```

        username = request.form['username']

```

```

        password = request.form['password']

```

```

        sql = "SELECT * FROM ACCOUNT 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)

```

```

    if account:

```

```

        passCheck = "SELECT UPASSWORD FROM ACCOUNT WHERE username =?"

```

```

        stmt = ibm_db.prepare(conn, passCheck)

```

```

        ibm_db.bind_param(stmt,1,username)

```

```

        ibm_db.execute(stmt)

```

```

        result = ibm_db.fetch_assoc(stmt)

```

```

        passWordInDb = result["UPASSWORD"]

```

```

        if passWordInDb == password:

```

```

            session['loggedin'] = True

```

```

            #session['id'] = account['UID']

```

```

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

```

```

            msg = 'Logged in successfully !'

```

```

            return render_template('dashboard.html', msg = msg)

```

```

        else:

```

```

            msg = 'Incorrect username / password !'

```

```

else:
    msg = 'Incorrect username / password !'
    """ if account:
        session['loggedin'] = True
        session['id'] = account['id']
        session['username'] = account['username']
        msg = 'Logged in successfully !'
        return render_template('index.html', msg = msg) """

return render_template('signin.html', msg = msg)

def applyJob():
    print("-----Function Called")

@app.route('/profile', methods =['GET', 'POST'])
def profile():
    user = session['username']
    sql = "SELECT * FROM ACCOUNT WHERE USERNAME = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,user)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    usernameInUser = account['USERNAME']
    userPassword = account['UPASSWORD']
    userEmail = account['EMAILID']
    firstName = account['FIRSTNAME']
    lastName = account['LASTNAME']
    print(account)

    return render_template('profile.html',
usernameInUser=usernameInUser,userPassword=userPassword,userEmail=userEmail,firstNam
e=firstName,lastName=lastName)

@app.route('/editProfile', methods =['GET', 'POST'])
def editProfile():
    if request.method == 'POST':
        msg = ""
        username = request.form['usernameInUser']
        password = request.form['userPassword']
        email = request.form['userEmail']

```

```

    fname = request.form['firstName']
    lname = request.form['lastName']
    sql = "UPDATE ACCOUNT SET UPASSWORD = ?, EMAILID = ?, FIRSTNAME = ?, LASTNAME =
? WHERE USERNAME = ?;"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,password)
    ibm_db.bind_param(stmt,2,email)
    ibm_db.bind_param(stmt,3,fname)
    ibm_db.bind_param(stmt,4,lname)
    ibm_db.bind_param(stmt,5,username)
    print("::::::::::::::::::::::::::",sql)
    ibm_db.execute(stmt)
    msg = "Saved Successfully !"

    return render_template('profile.html', msg = msg,
usernameInUser=username,userPassword=password,userEmail=email,firstName=fname,lastNa
me=lname)

```

```

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('username', None)
    return redirect(url_for('signin'))

```

```

@app.route('/signup', methods =['GET', 'POST'])
def signup():
    msg = "
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        email = request.form['email']
        fname = request.form['fname']
        lname = request.form['lname']
        sql = "SELECT * FROM ACCOUNT 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)

        if account:
            msg = 'Account already exists !'
        else:

```

```

insert_sql = "INSERT INTO ACCOUNT VALUES (?, ?, ?, ?, ?)"
prep_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prepare_stmt, 1, username)
ibm_db.bind_param(prepare_stmt, 2, password)
ibm_db.bind_param(prepare_stmt, 3, email)
ibm_db.bind_param(prepare_stmt, 4, lname)
ibm_db.bind_param(prepare_stmt, 5, fname)
ibm_db.execute(prepare_stmt)
msg = 'Data inserted successfully'
return render_template('signup.html', msg = msg)

```

```

@app.route('/jobapplied/<int:jobid>')
def jobappliedFunction(jobid):
    jobid = jobid
    sql = "SELECT JOBCOMPANY FROM JOBMARKET WHERE JOBID =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,jobid)
    ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)
    jobname = result['JOBCOMPANY']
    sql = "SELECT COMPANY_EMAIL FROM JOBMARKET WHERE JOBID =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,jobid)
    ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)
    jobemail = result['COMPANY_EMAIL']
    print("-----JOB APPLIED-----",jobid)
    return render_template('fillapplication.html',jobid = jobid, jobname = jobname, jobemail =
jobemail)

```

```

@app.route('/appliedjob', methods =['GET', 'POST'])
def appliedjob():
    username = session['username']
    passCheck = "SELECT EMAILID FROM ACCOUNT WHERE username =?"
    stmt = ibm_db.prepare(conn, passCheck)
    ibm_db.bind_param(stmt,1,username)
    ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)
    fromEmail = result["EMAILID"]

```



```

msgcontent = request.form['reasoncontent']
emailJob = request.form['jobEmailForm']
portfolioLink = request.form['portfolio']
city = request.form['citypreffered']
appliedJobId = request.form['appliedJobId']
print("-----",appliedJobId)
insert_sql = "INSERT INTO APPLIEDJOBS VALUES (?,?)"
prep_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prepare_stmt, 1, username)
ibm_db.bind_param(prepare_stmt, 2, int(appliedJobId))
ibm_db.execute(prepare_stmt)

msg = Message('Hello',sender = fromEmail,recipients = [emailJob])
msg.body = "Applicant Email : " + fromEmail + "\n" + "\nAbout Me : \n" + msgcontent + "\n" +
"\nPortfolio Link : " + portfolioLink + "\n" + "\nPreffered City : " + city
mail.send(msg)
return redirect(url_for('jobsapplied'))

```

```

@app.route('/jobsapplied')
def jobsapplied():
    jobids1 = []
    jobinformation = []

    sql = "SELECT * FROM APPLIEDJOBS WHERE USERNAME = ?"
    stmt = ibm_db.prepare(conn, sql)
    username = session['username']
    print(username)
    ibm_db.bind_param(stmt,1,username)
    ibm_db.execute(stmt)
    joblist = ibm_db.fetch_tuple(stmt)
    print(joblist)
    while joblist != False:
        print("-----",joblist)
        jobids1.append(joblist[1])
        joblist = ibm_db.fetch_tuple(stmt)

    print(jobids1)
    for x in range(len(jobids1)):
        jobids = []
        jobnames = []

```

