

# PROJECT REPORT

## 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.

### 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.

## 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

### 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.

### 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>

## 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?".

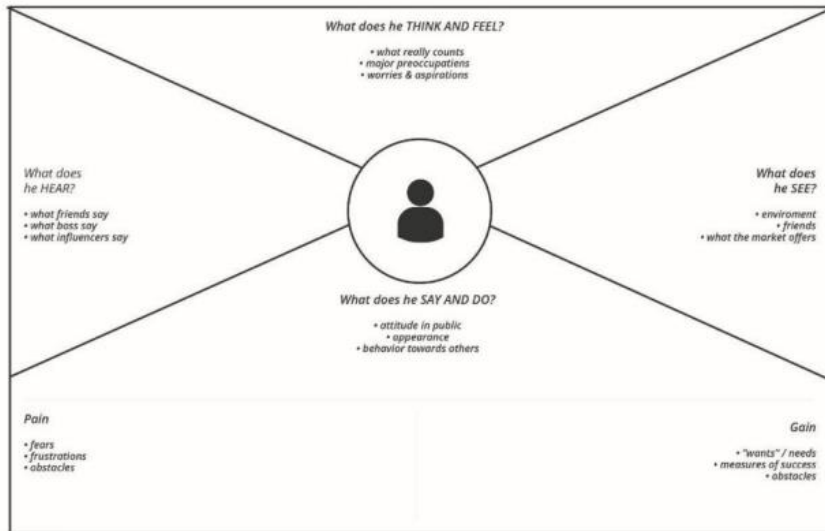
In current situation recruitment s done manually for lakhs of students in which many talented students may lose their opportunities due to different reasons since it is done manually, and company also need the highly talented people from the mass group for their growth. So we have build a cloud application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

## 3. IDEATION AND PROPOSED SOLUTION

### 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



## 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 wants a better way to get a better financial control, justify his skills, move to a relevant domain, learn new skills, challenge himself, career growth, and get a better lifestyle.

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 hit the pencil icon to start drawing!

Shyam Sunder S

Obstacle in searching job

User friendly environment

Optimised Search Engine

Communication between user and employer

Sham Kumar J

The suggestions results are Responsive

Skill based domain

Solving the queries

Geo expand results

Nishish Kumar R

Accurate job search results

Description of job details

Desired Salary

Connecting people society

Ashish

Identifying roles

Jobs for non-technical domain

Providing detailed information of user

Time management

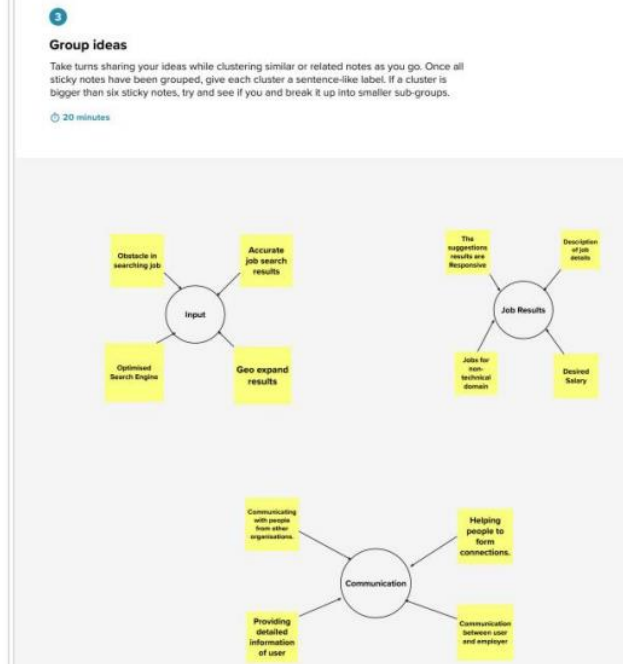
Himasha Reddy

Referral

Communicating with people from other organizations.

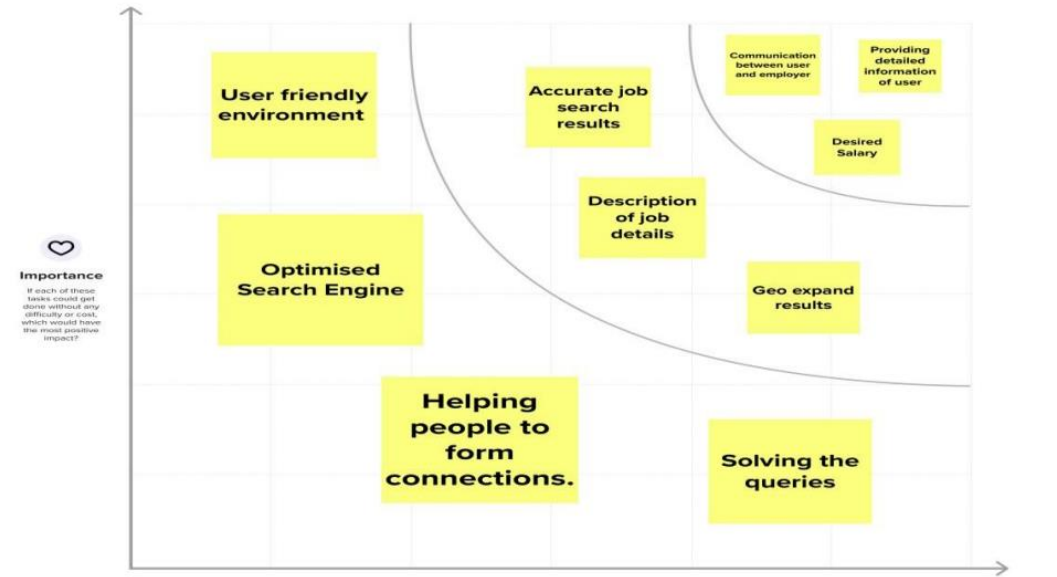
A place for all jobs

Helping people to form connections.



### STEP 3:

#### Idea Prioritization



#### 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

## PROBLEM SOLUTION FIT

|  |  |  |  |  |
|--|--|--|--|--|
| Define CS, Fit in CC                     | <b>1. CUSTOMER SEGMENT(S)</b><br><b>Who is your customer?</b> <b>CS</b><br><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><br><b>What constraint prevents your customer from taking action or limiting their choice of solution?</b> <b>CC</b><br><p>The problem of contacting the agent and all the problems and procedure in it.</p>   | <b>5. AVAILABLE SOLUTION</b><br><b>Which solutions are available to the customer when they face the problem.</b> <b>AS</b><br><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><br><b>Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; Explore different sides?</b> <b>J&amp;P</b><br><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><br><b>What is the real reason that the problem exists?</b> <b>RC</b><br><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><br><b>What does your customer do to address the problem and get the job done.</b> <b>BE</b><br><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><br><b>What triggers customers to act.</b> <b>ER</b><br><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><br>Our solution involves autonomous system which does the following: <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><br><b>ONLINE</b> <b>CH</b><br><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><br><b>How do customers feel when they face a problem or a job and afterwards.</b> <b>TM</b><br><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><br><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

### FUNCTIONAL REQUIREMENT

| Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task)  |
|-------------------------------|---|
| User Registration             | Registration through Form<br>Registration through Gmail   |
| User Confirmation             | Confirmation via Email<br>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<br>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.  |

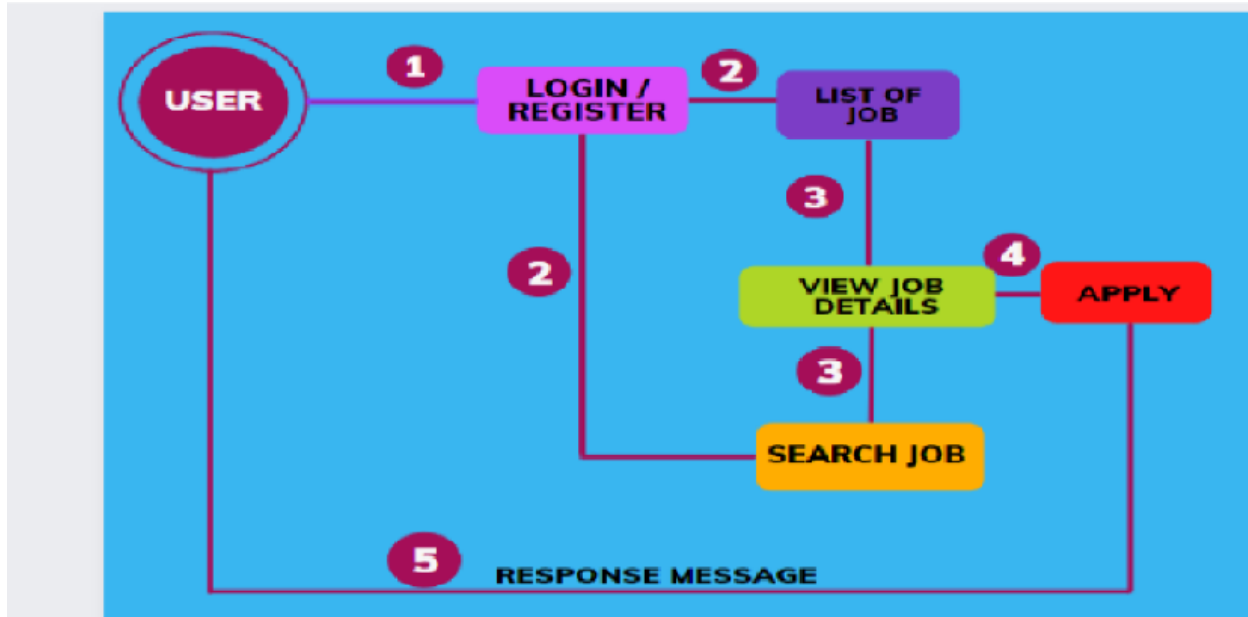
### NON FUNCTIONAL REQUIREMENTS

Non functional Requirements are :

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

## 5 PROJECT DESIGN

### DATAFLOW DIAGRAM



### TECHNICAL ARCHITECTURE

Solution architecture is a complex process – with many sub-processes – that bridges

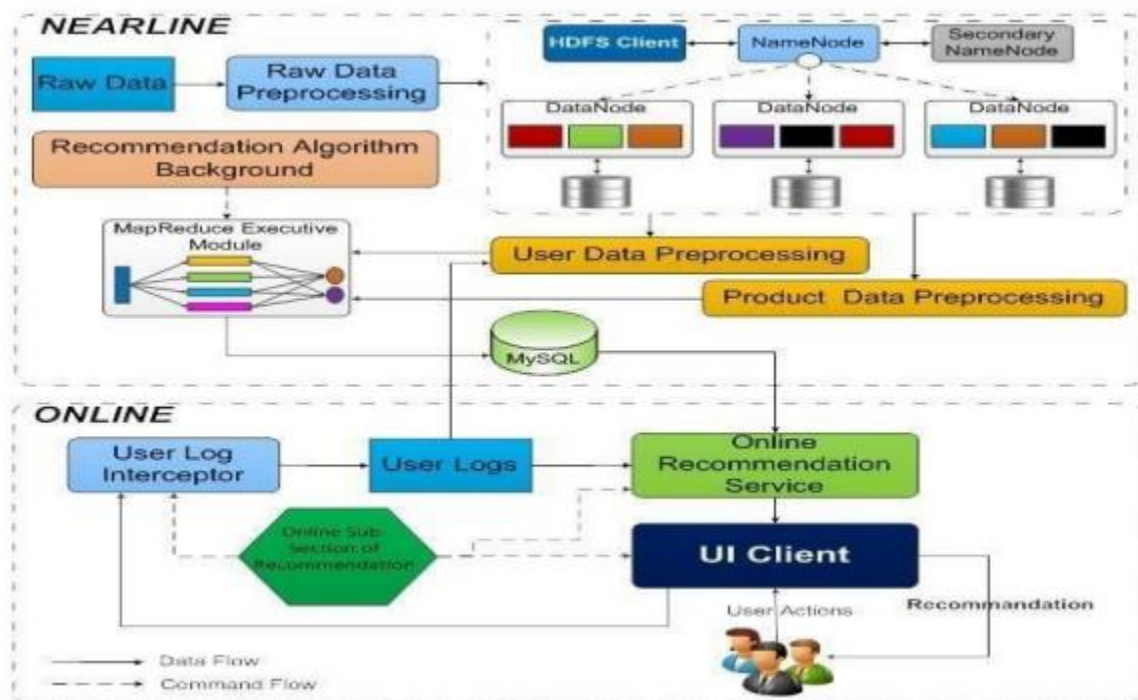
the gap between business problems and technology solutions. Its goals are to:

- 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

#### Architecture



## 6 PROJECT PLANNING AND SCHEDULING

### SPRINT PLANNING AND EXSTIMATION

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

## SPRINT DELIVERY SCHEDULE

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

## 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.

## 7.CODING & SOLUTIONING

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 = []

        cols = 4
        size = len(jobnames)
        print("$$$$$$$$$$$$$$$$$$$$$$$$4",skill1,skill2,skill3)

        for i in range(size):
            col = []

print("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
if jobdescription[i].lower() == skill1.lower() or jobdescription[i].lower() == skill2.lower() or
jobdescription[i].lower() == skill3.lower() :
```

[illegible]

### Feature 2:

This chat bot feature provides help tooltip for end users if any help needed for users

### Database Schema:

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

5QL

Tables

New table

Filter

Sort

Columns

| Name         | Schema   | Properties |
|--------------|----------|------------|
| ACCOUNT      | JDD83131 | ...        |
| ACCOUNTSKILL | JDD83131 | ...        |
| APPLIEDJOBS  | JDD83131 | ...        |
| CUSTOMER     | JDD83131 | ...        |
| JOBMARKET    | JDD83131 | ...        |
| STUDENTS     | JDD83131 | ...        |

Total: 6, selected: 0

Table definition

STUDENTS

Approximate 3 rows (32.0 KB)  
Updated on 2022-10-26 18:36:10

| Name    | Data type | Nullable | Length | Scale |
|---------|-----------|----------|--------|-------|
| NAME    | VARCHAR   | Y        | 255    | 0     |
| ADDRESS | VARCHAR   | Y        | 255    | 0     |
| CITY    | VARCHAR   | Y        | 255    | 0     |
| PIN     | VARCHAR   | Y        | 255    | 0     |

View data

IBM Db2 on Cloud

Load DataLoad HistoryTablesViewsIndexesAliasesMQTSSequencesApplication objects

Find schemas or tables

Refresh

Schemas

5QL

Tables

New table

Filter

Sort

Columns

| Name         | Schema   | Properties |
|--------------|----------|------------|
| ACCOUNT      | JDD83131 | ...        |
| ACCOUNTSKILL | JDD83131 | ...        |
| APPLIEDJOBS  | JDD83131 | ...        |
| CUSTOMER     | JDD83131 | ...        |
| JOBMARKET    | JDD83131 | ...        |
| STUDENTS     | JDD83131 | ...        |

Total: 6, selected: 0

Table definition

JOBMARKET

Approximate 6 rows (32.0 KB)  
Updated on 2022-10-30 09:25:21

| Name          | Data type | Nullable | Length | Scale |
|---------------|-----------|----------|--------|-------|
| JOBID         | INTEGER   | N        |        | 0     |
| JOBCOMPANY    | VARCHAR   | Y        | 255    | 0     |
| JOBIMAGE      | VARCHAR   | Y        | 500    | 0     |
| JOBSKILL      | VARCHAR   | Y        | 255    | 0     |
| COMPANY_EMAIL | VARCHAR   | Y        | 255    | 0     |

View data

IBM Db2 on Cloud

Load DataLoad HistoryTablesViewsIndexesAliasesMQTsSequencesApplication objects

Find schemas or tables

Refresh

SQL

Schemas

Tables

Views

Indexes

Aliases

MQTs

Sequences

Application objects

Tables

New table

| Name         | Schema   | Properties |
|--------------|----------|------------|
| ACCOUNT      | JDD83131 | ...        |
| ACCOUNTSKILL | JDD83131 | ...        |
| APPLIEDJOBS  | JDD83131 | ...        |
| CUSTOMER     | JDD83131 | ...        |
| JOBMARKET    | JDD83131 | ...        |
| STUDENTS     | JDD83131 | ...        |

Total: 6, selected: 0

Table definition

CUSTOMER

Approximate 0 rows (0 KB)  
Updated on 2022-10-19 04:42:20

| Name       | Data type | Nullable | Length | Scale |
|------------|-----------|----------|--------|-------|
| CUSTOMERID | INTEGER   | Y        |        | 0     |
| LASTNAME   | VARCHAR   | Y        | 255    | 0     |
| FIRSTNAME  | VARCHAR   | Y        | 255    | 0     |
| ADDRESS    | VARCHAR   | Y        | 255    | 0     |
| CITY       | VARCHAR   | Y        | 255    | 0     |

View data

IBM Db2 on Cloud

Load DataLoad HistoryTablesViewsIndexesAliasesMQTsSequencesApplication objects

Find schemas or tables

Refresh

SQL

Schemas

Tables

Views

Indexes

Aliases

MQTs

Sequences

Application objects

Tables

New table

| Name         | Schema   | Properties |
|--------------|----------|------------|
| ACCOUNT      | JDD83131 | ...        |
| ACCOUNTSKILL | JDD83131 | ...        |
| APPLIEDJOBS  | JDD83131 | ...        |
| CUSTOMER     | JDD83131 | ...        |
| JOBMARKET    | JDD83131 | ...        |
| STUDENTS     | JDD83131 | ...        |

Total: 6, selected: 0

Table definition

APPLIEDJOBS

Approximate 16 rows (32.0 KB)  
Updated on 2022-11-11 09:35:52

| Name     | Data type | Nullable | Length | Scale |
|----------|-----------|----------|--------|-------|
| USERNAME | VARCHAR   | N        | 255    | 0     |
| JOBID    | INTEGER   | N        |        | 0     |

View data

Donate | The Eclipse Foundation

IBM-Project-24324-1659941545/base

Service Details - IBM Cloud

IBM Db2 on Cloud

bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crm%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%...

IBM Db2 on Cloud

Load DataLoad HistoryTablesViewsIndexesAliasesMQTsSequencesApplication objects

Find schemas or tables

Refresh

Schemas

SQL

Tables

New table +

Filter

Sort

Columns

| <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

Table definition

ACCOUNTSKILL

Approximate 6 rows (32.0 KB)  
Updated on 2022-11-13 17:50:47

| Name     | Data type | Nullable | Length | Scale |   |
|----------|-----------|----------|--------|-------|---|
| USERNAME | VARCHAR   | N        | 255    | 0     | 🔗 |
| SKILL1   | VARCHAR   | Y        | 255    | 0     | 🔗 |
| SKILL2   | VARCHAR   | Y        | 255    | 0     | 🔗 |
| SKILL3   | VARCHAR   | Y        | 255    | 0     | 🔗 |

View data

Donate | The Eclipse Foundation

IBM-Project-24324-1659941545/base

Service Details - IBM Cloud

IBM Db2 on Cloud

bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crm%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%...

IBM Db2 on Cloud

Load DataLoad HistoryTablesViewsIndexesAliasesMQTsSequencesApplication objects

Find schemas or tables

Refresh

Schemas

SQL

Tables

New table +

Filter

Sort

Columns

| <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

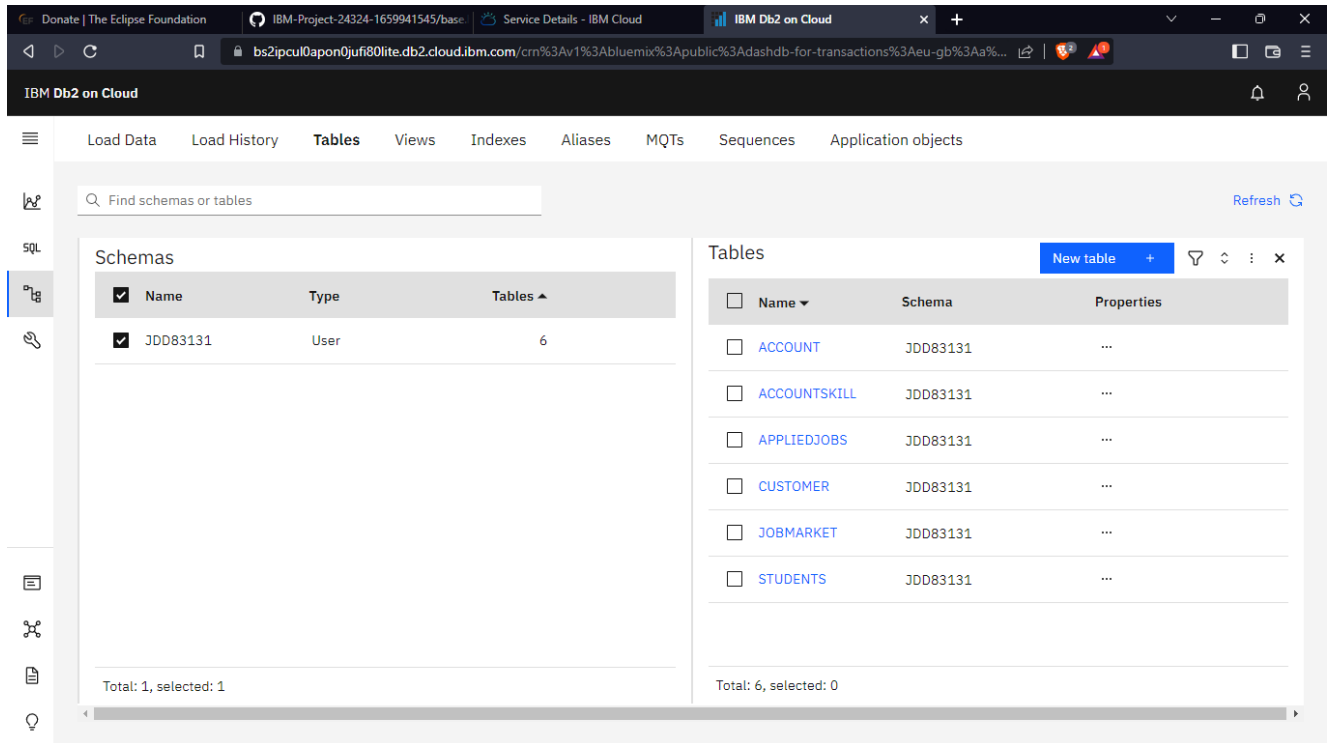
Table definition

ACCOUNT

Approximate 16 rows (32.0 KB)  
Updated on 2022-11-02 21:25:41

| Name      | Data type | Nullable | Length | Scale |   |
|-----------|-----------|----------|--------|-------|---|
| USERNAME  | VARCHAR   | N        | 255    | 0     | 🔗 |
| UPASSWORD | VARCHAR   | Y        | 255    | 0     | 🔗 |
| EMAILID   | VARCHAR   | Y        | 255    | 0     | 🔗 |
| LASTNAME  | VARCHAR   | Y        | 255    | 0     | 🔗 |
| FIRSTNAME | VARCHAR   | Y        | 255    | 0     | 🔗 |

View data



## 8.TESTING

### 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.

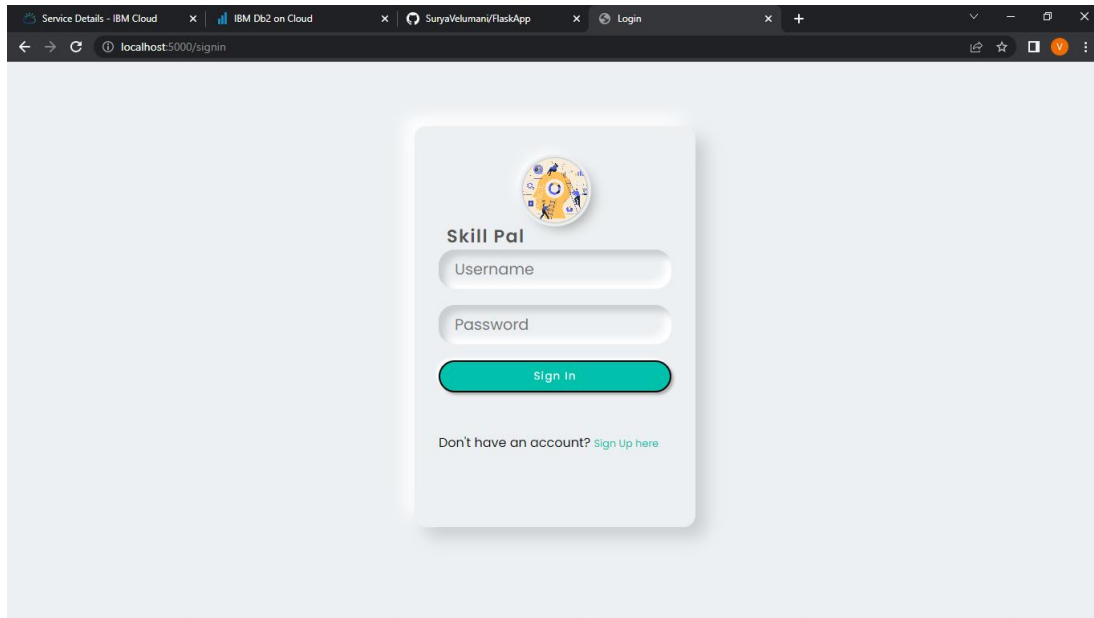
Testing was done in phase 1 and phase 2, where issues found in phase1 were fixed and then tested again in phase2.

### User Acceptance Testing:

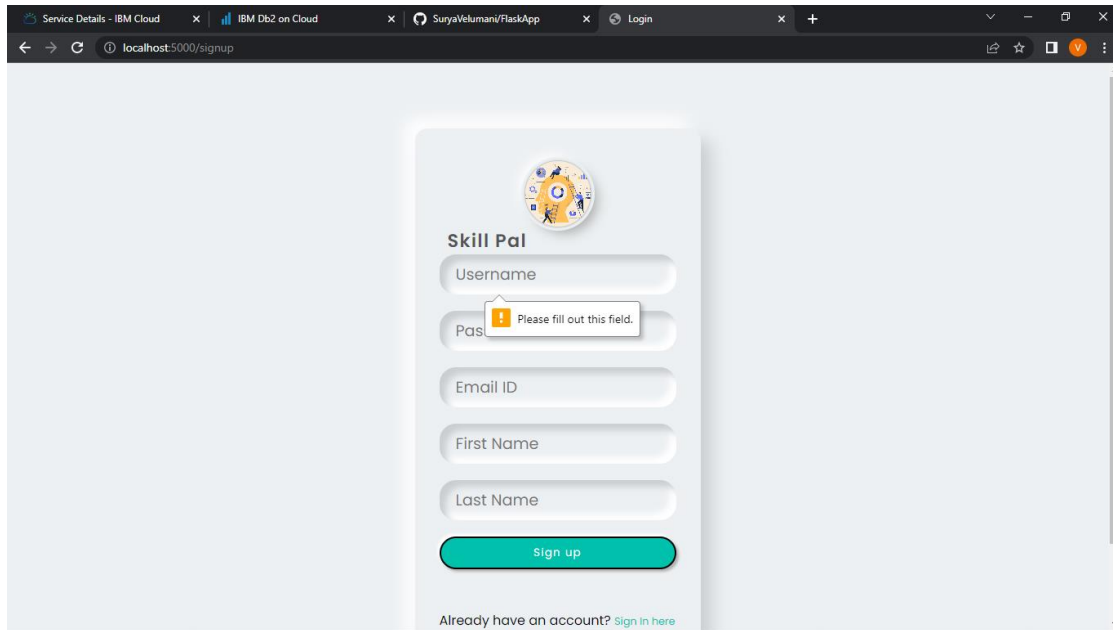
Real world testing was also done, by giving to remote users and asking them to use the application. Their difficulties were fixed and tested again until all the issues were fixed.

## 9.RESULTS

### Performance Metrics:




A screenshot of a web browser displaying the Skill Pal login page. The browser's address bar shows 'localhost:5000/signin'. The page features a central white card with a circular logo at the top. Below the logo, the text 'Skill Pal' is displayed. There are two input fields: 'Username' and 'Password'. A green 'Sign in' button is positioned below the password field. At the bottom of the card, there is a link that says 'Don't have an account? Sign Up here'.



A screenshot of a web browser displaying the Skill Pal signup page. The browser's address bar shows 'localhost:5000/signup'. The page features a central white card with a circular logo at the top. Below the logo, the text 'Skill Pal' is displayed. There are five input fields: 'Username', 'Password', 'Email ID', 'First Name', and 'Last Name'. A green 'Sign up' button is positioned below the 'Last Name' field. A tooltip message 'Please fill out this field.' is visible over the 'Password' field. At the bottom of the card, there is a link that says 'Already have an account? Sign in here'.

Service Details - IBM Cloud x IBM Db2 on Cloud x SuryaVelumani/FlaskApp x Login

localhost:5000/signup




### Skill Pal

Already have an account? [Sign in here](#)


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

localhost:5000/jobmarket


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




Zoho  
1  
java



Amazon  
2  
cloud



Google  
3  
Go lang



Please tell about yourself, and why you need this job :

Hi, I am interested in applying for your company.

6

Company :

Amazon

Company Email :

suryaveducation@gmail.com

Portfolio Link :

www.demoPortfolio.com

Preferred Location :

CBE

Submit form

Close

Hi there, need help ?

Waiting for integrations.au-syd.assistant.watson.appdomain.cloud...

Service Details - IBM Cloud x IBM Db2 on Cloud x SuryaVelumani/FlaskApp x SkillPal x Hello - 19euec156@skcet.ac x +

mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzGqRZZZtSsFqFvzdTfzFJsSVSG

Gmail Search in mail

Compose

Inbox 5

Starred

Snoozed

Sent

Drafts

More

Labels +

1 of 7,902

Hello External Inbox x

suryaveducation@gmail.com via sendgrid.net to me 3:00 PM (0 minutes ago)

Applicant Email : [suryaveducation@gmail.com](mailto:suryaveducation@gmail.com)

About Me :  
hi

Portfolio Link :


Preferred City :

Reply Forward

- Create an account using sign up and then sign in to your account.
- Use the Skill Enhancement tab to find courses and improve your skills.
- Add the skills that you have.
- Find all the jobs in job market.
- By clicking the filter icon on top right corner you can filter the jobs according to the skills you have.
- Click on Apply button to apply for the job. You will be navigated to a form page.
- Fill the text areas.
- Click submit form, so your application is successfully sent to the company.
- You can update your profile anytime you want by clicking the profile icon on top left in navbar.
- Use the Skill Pal assistant by clicking the message pop up in bottom right corner of the page.

Close


Hi there, need help ?



Zoho

5

java



Amazon

6

cloud

Close

Hi there, need help ?



Hi there, need help ?

Surya Velumani

Dashboard

Email API

Marketing

Design Library

Stats

Activity

Suppressions

Settings

Twilio SMS NEW

REPUTATION 83%

VIEW ACCOUNT USAGE

## Activity Feed

Search emails by:

To email address  
suryaveducation@gmail.com

Dates  
2022/11

| STATUS                   | MESSAGE                                |
|--------------------------|--|
| <span>●</span> Delivered | To: suryaveducation@gmail.com<br>Hello |
| <span>●</span> Delivered | To: suryaveducation@gmail.com<br>Hello |

### Email Information

Close

#### Details

|         |                           |
|---------|---------------------------|
| To      | suryaveducation@gmail.com |
| From    | suryaveducation@gmail.com |
| Subject | Hello                     |

[More Details](#)

#### Event History

✉ Received by SendGrid

✓ Processed 2022/11/11 5:35am UTC-04:00 >

✉ Received by gmail-smtp-in.l.google.com


✓ Delivered 2022/11/11 5:35am UTC-04:00 >




Service Details - IBM Cloud x IBM Db2 on Cloud x SuryaVelumani/FaskApp x SkillPal x Inbox (3) - suryaveducationE x +

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®

Service Details - IBM Cloud x IBM Db2 on Cloud x SuryaVelumani/FaskApp 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



Skill 1  
Java

Skill 2

Skill 3

Save

Close

Hi there, need help ?

## 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.

## 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 techniques 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 threshold to 15 MB
        file_threshold = 1024 * 1024 * 15

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

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

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

@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():
    usernameskill = session['username']
    sql = "SELECT * FROM ACCOUNTSKILL WHERE USERNAME = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,usernameskill)
    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,usernameskill)
        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("-----",usernameskill )
        sql = "INSERT INTO ACCOUNTSKILL VALUES (?, ?, ?, ?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,usernameskill)
        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)
    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 = []

cols = 4
size = len(jobnames)
print("$$$$$$$$$$$$$$$$$$$$$$$4",skill1,skill2,skill3)

for i in range(size):
    col = []

print("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@",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,firstName=firstNa
me,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,lastName=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 = []
        jobimages = []
        jobdescription = []

        print("nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn",len(jobids1))
        sql = "SELECT * FROM JOBMARKET WHERE JOBID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,jobids1[x])

```

