

# 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: 10.5897/1JPS12. 482

e 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?".

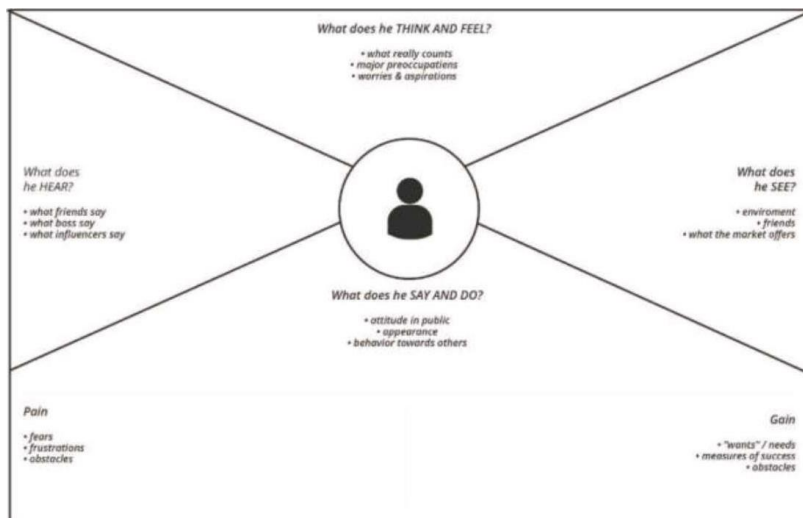
In current situation recruitment s done manually for lakhs of students in which many talented students may lose their opportunities due to different reasons since it is done manually, and company also need the highly talented people from the mass group for their growth. So we have build a cloud application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



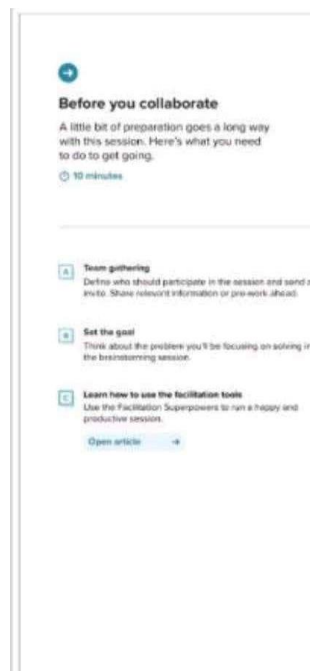
## IDEATION AND BRAINSTROMING

### 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



STEP 2:

Brainstorm, Idea Listing and Grouping

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 (switch to sketch) icon to start drawing!

Effective in searching job	User friendly environment
Optimized Search Results	Personalization between user and employer

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

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

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

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

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/new job to gain a better financial condition, 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.

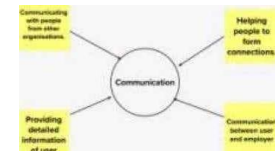
Listen to others.

Go for volume.

If possible, be visual.

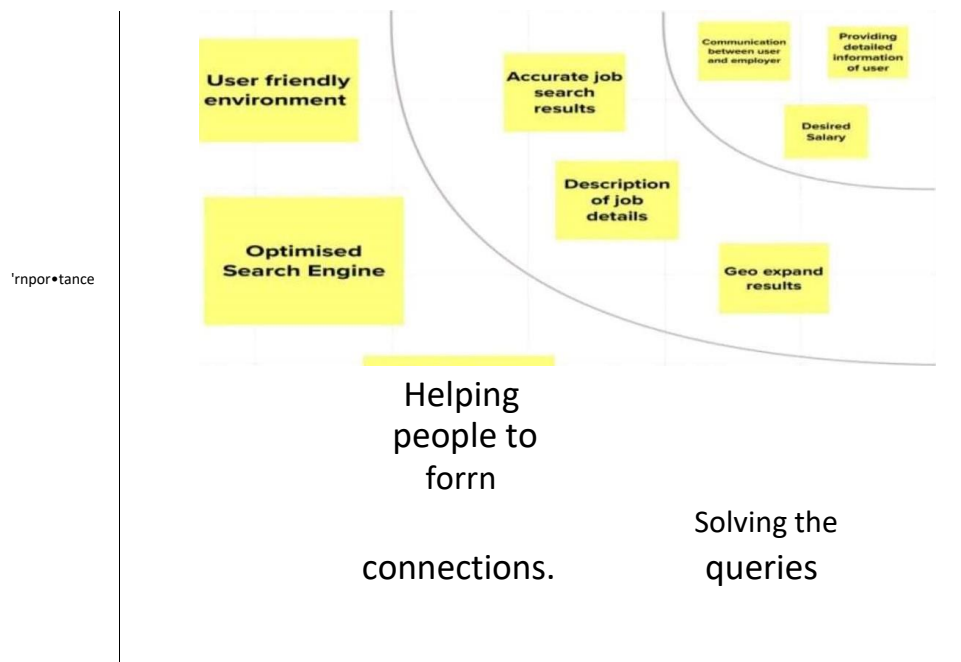
Group ideas

Take sticky      have been grouped, give      sentence-jibe      i' a Ouster      Sin      try ytd see i' you      up      swgrouOS.



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

	<p><b>CUSTOMER SEGMENT(S)</b> Who your customer?</p> <p>Custcyner who are rot able to solve ttwir own Problem aNi need for possible solution from their agentsht providers.</p>	<p><b>6. CUSTOMER CONSTRAIN%</b> What constriant prevents pur custM*t from taking action or</p> <p>imiting their choice of solution?</p> <p>The problem ot the agent and all the problems and procedure in it.</p>	<p><b>5. AVARABIE SOLUTION</b> Which solutions are available to the customer w*ten they face the poblan.</p> <ul style="list-style-type: none"> <li>They can check FAQs Session for fast</li> <li>It the problem is not Osted, they can post the problem section. •n new Which will further assisted by the agent team.</li> </ul>	Explore AS, Differentiate
Focus on job, Top	<p><b>2. JOBS.TOOE.DONE/ROUEMS</b> Which jobs-to-be-done (or plot*ems) do you for pur customers? There could be thn orr,• Explore different sides?</p> <p>this Application Alcyws Custom.ers to get recommended job according to thfr skillset</p> <p>They will be post their resume and wait for the solution,</p> <p>They will also get solutions to their queries</p> <p>They can also out FAQ's Section on our</p>	<p><b>9. PROBLEM ROOT CAUSE.</b> What is the real reason that the problem exists?</p> <p>ttw only real reason that tha problem exists is the lack of awaret*ss and ratio of pmen resets which create trust issues their agent</p>	<p><b>7. BEHAVIOR</b> What does your ctBtomet do to address the problem and tet the job</p> <ul style="list-style-type: none"> <li>They must first Post their reswne and then wait for 2 hours,</li> <li>They can use chatbot to easily contKt wr Team</li> <li>They can •iso refer the FAQs session</li> </ul>	BE
Focus on job, Top into BE, Understand RC	<p><b>3. TRIGGERS</b> What triggers customers to act.</p> <ul style="list-style-type: none"> <li>Customers get to know the absolute recommendation to their need.</li> <li>Fast Response.</li> </ul>	<p><b>10. YOUR SOLUTION</b> Our solution involves autonomous system which does the following:</p> <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>	<p><b>8. CHANNELS of BEHAVIOR</b></p> <p><b>ONLINE</b></p> <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> <p><b>OFFLINE</b></p> <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>	CH
Identify string TR & ME	<p><b>4. EMOTIONS: BEFORE/AFTER</b> How do customers feel when they face a problem or a job and afterwards.</p> <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>			Identify string TR & ME

## 4. REQUIREMENT ANALYSIS

### 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 fitters and skill recommendations.
	User Profile	Updation of the user profile through the login credentials
	User Acceptance	Confirmation of the Job.

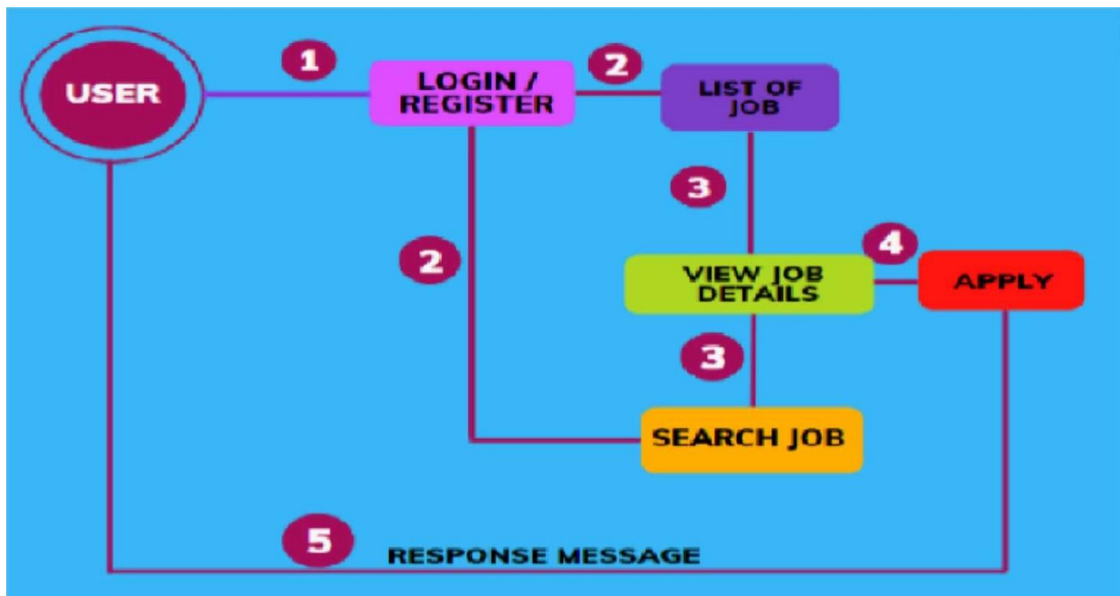
### NON FUNCTIONAL REQUIREMENTS

Non functional Requirements are :

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

## 5 PROJECT DESIGN

### DATAFLOW DIAGRAM



## TECHNICAL ARCHITECTURE

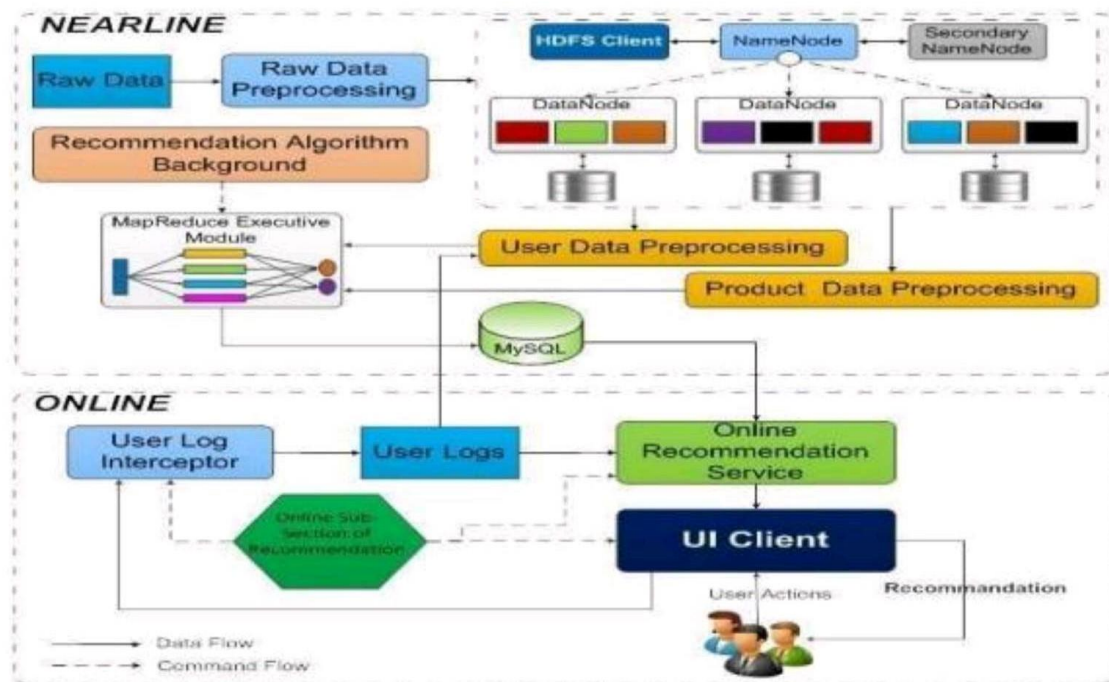
Solution architecture is a complex process with many sub-processes — that bridges the gap between business problems and technology solutions.

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

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	Develop and submit the developed code by testing it.

## SPRINT DELIVERY SCHEDULE

SPRINT	TASK	MEMBERS
SPRINT 1	Create Registration page , login page , Job search portal , job apply portal in flask	ANUSIYA M HEMALATHA S JEEVITHA R RUPAVATHI L
SPRINT 2	Connect application to ibm db2	ANUSIYA M HEMALATHA S JEEVITHA R RUPAVATHI L

SPRINT 3	Integrate ibm Watson assistant	ANUSIYA M HEMALATHA S JEEVITHA R RUPAVATHI L
SPRINT 4	Containerize the app and Deploy the application in ibm cloud	ANUSIYA M HEMALATHA S JEEVITHA R RUPAVATHI L

#### 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 = 1]
```

```
    jobnames = [l
```

```
    jobimages = [l
```

```
    jobdescription =[]
```

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

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

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

```
    print(username)
```

```
    #ibm_db.bind_param(stmt,l,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(): skilll = "" ski112 = "" ski113 = "" 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:
    skilll = skillres['SKILL1']
    ski112 =
    skillres['SKILL2']
    ski113 =
    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])

```

```
<script> window.watsonAssistantChatOptions = { integrationID: "9be41b76-06b0-426f-8469-962f2963cdb6", // The ID of this integration. region: "au-syd", // The region your integration is hosted in.
  serviceInstanceID: "76838ca2-a227-4f56-b180-94f01901cdbf", // The ID of your service instance.
  onLoad: function(instance) { instance.render(); }

  setTimeout(function(){
    const
    t=document.createElement( 'script ');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
```



</script>

Database Schema:

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

IBM Db2 on Cloud

Load Data

Load History

Tables

Views

Indexes

Aliases

MQTs

Sequences

Application objects

Q Find schemas or tables

SQL

Schemas

Tables

New table

Table definition

STUDENTS

Approximate 3 rows (32.0 KS)

dated on

Refresh

Name	Schema	Properties	2022-10-26
ACCOUNT	JDD83131	Name	Data type
ACCOUNTSKILL	JDD83131	Nullable	Length
APPLIEDJOBS	JDD83131	Scale	
CUSTOMER	JDD83131		
JOBMARKET	JDD83131		
STUDENTS	JDD83131		

Total: 6, selected: 0

View data

IBM Db2 on Cloud

Load Data

Load History

Tables

Views

Indexes

Aliases

MQTs

Sequences

Application objects

JOBMARKET	JDD83131
STUDENTS	JDD83131

Total: 6, selected: 0

## IBM Db2 on Cloud

Q Find schemas or

### SQLTable definition

			JOBMARKET		Approximate 6 rows (32.0 KB)	
				Data type	Nullable	Length
C] CUSTOMER	ACCOUNT	JDD83131				
			JOBID	INTEGER		
	ACCOUNTSKILL	JDD83131				
			JOBCOMPANY	VARCHAR		255
	APPLIEDJOBS	JDD83131				
			JOBIMAGE	VARCHAR		500
			JOBSKILL	VARCHAR		255
			COMPANY_EMA	VARCHAR		255
			IL			

### Q Find schemas or tables

Refresh

Total: 6, selected: 0 [View data](#)

## SQL Tables

SQL Tables		Table definition	
CUSTOMER	Approximate O rows (O RE) Name	Schema	Properties
			Updated on 2022-10-19 04:42:20
ACCOUNT	JDD83131		
ACCOUNTSKILL	JDD83131	CUSTOMERID	INTEGER
APPLIEDJOBS	JDD83131	LASTNAME	VARCHAR
		FIRSTNAME	VARCHAR
CUSTOMER	JDD83131	ADDRESS	VARCHAR
JOBMARKET	JDD83131	CITY	VARCHAR
JOBMARKET	JDD83131		
STUDENTS	JDD83131		

Total: 6, selected:  
0

IBM Db2 on Cloud

IBM Db2 on Cloud

Load Data

Load History

Tables

Views

Indexes

Aliases

Materialized Views

Sequences

Application objects

STUDENTS

JDD83131

Total: 6, selected: 0

View data

IBM Db2 on Cloud

Load Data

Load History

Tables

Views

Indexes

Aliases

Materialized Views

Sequences

Application objects

Q.Find schemas or tables

Refresh

SQL

Schemas

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

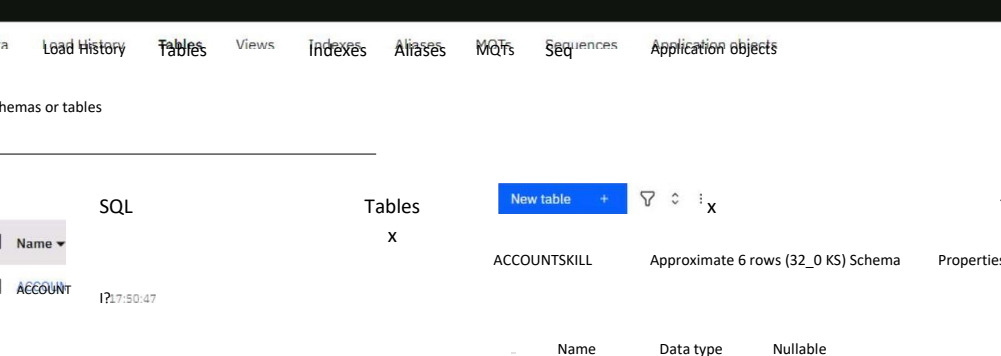
View data

Name	Schema	Properties	APPLIEDJOBS	Approximate 16 rows (32.0 KB)
ACCOUNT	JDD83131		Name	2022-11-21
ACCOUNTSKILL	JDD83131		Data type	Scale
APPLIEDJOBS	JDD83131		Nullable	
[3 CUSTOMER	JDD83131		Length	

ACCOUNT	JDD83131		USERNAME	VARCHAR	255
ACCOUNTSKILL	JDD83131		JOBID	INTEGER	
APPLIEDJOBS	JDD83131				
[3 CUSTOMER	JDD83131				
JOBMARKET	JDD83131				
STUDENTS	JDD83131				

Total: 6, selected: 0

## IBM Db2 on Cloud



The screenshot displays the IBM Db2 on Cloud console interface. The 'Tables' tab is selected, showing a list of tables. The 'ACCOUNTSKILL' table is highlighted, and its definition is shown in the 'Table definition' tab. The table definition includes columns: USERNAME (VARCHAR, 255), SKILL1 (VARCHAR, 255), SKILL2 (VARCHAR, 255), and SKILLS (VARCHAR, 255). The table is owned by JDD83131. The console also shows a 'SQL' tab with a query editor and a 'Table definition' tab with a 'View data' button.

Name	Data type	Nullable
USERNAME	VARCHAR	255
SKILL1	VARCHAR	255
SKILL2	VARCHAR	255
SKILLS	VARCHAR	255

IBM Db2 on Cloud

APPLIEDJOBS	JDD83131	UPASSWORD	VARCHAR	255
CUSTOMER	JDD83131	EMAILID	VARCHAR	255
		LASTNAME	VARCHAR	255
		FIRSTNAME	VARCHAR	255

View data

O

JOBMARKET	JDD83131
STUDENTS	JDD83131

The screenshot shows the IBM Db2 on Cloud web interface. The browser address bar displays the URL: `bs2ipcu0apon0jufi80lite.db2.cloud.ibm.com/cn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%...`. The interface has a top navigation bar with tabs: Load Data, Load History, Tables (selected), Views, Indexes, Aliases, MQTs, Sequences, and Application objects. Below the navigation bar is a search bar with the placeholder text "Find schemas or tables" and a "Refresh" button. The main content area is divided into two sections: "SQL" and "Tables". The "Tables" section displays a table with the following columns: Name, Schema, and Properties. The table lists six tables in the JDD83131 schema: ACCOUNT, C] ACCOUNTSKILL, C] APPLIEDJOBS, CUSTOMER, JOBMARKET, and STUDENTS. At the bottom of the interface, there is a status bar showing "Total: 1, selected: 1" and "Total: 6, selected: 0".

Name	Schema	Properties
ACCOUNT	JDD83131	
C] ACCOUNTSKILL	JDD83131	
C] APPLIEDJOBS	JDD83131	
CUSTOMER	JDD83131	
JOBMARKET	JDD83131	
STUDENTS	JDD83131	

## 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.

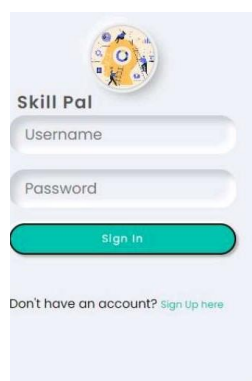
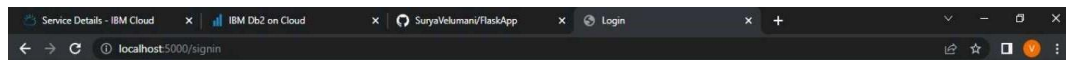
Testing was done in phase 1 and phase 2, where issues found in phasel were fixed and then tested again in phase2.

### User Acceptance Testing:

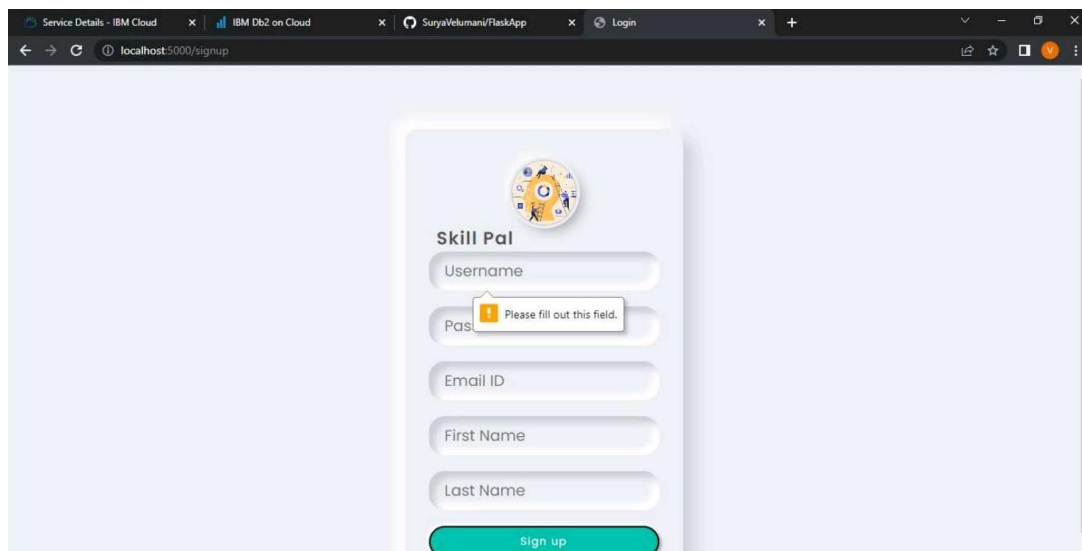
Real world testing was also done, by giving to remote users and asking them to use the application. Their difficulties were fixed and tested again until all the issues were fixed.

## 9.RESULTS

### Performance Metrics:



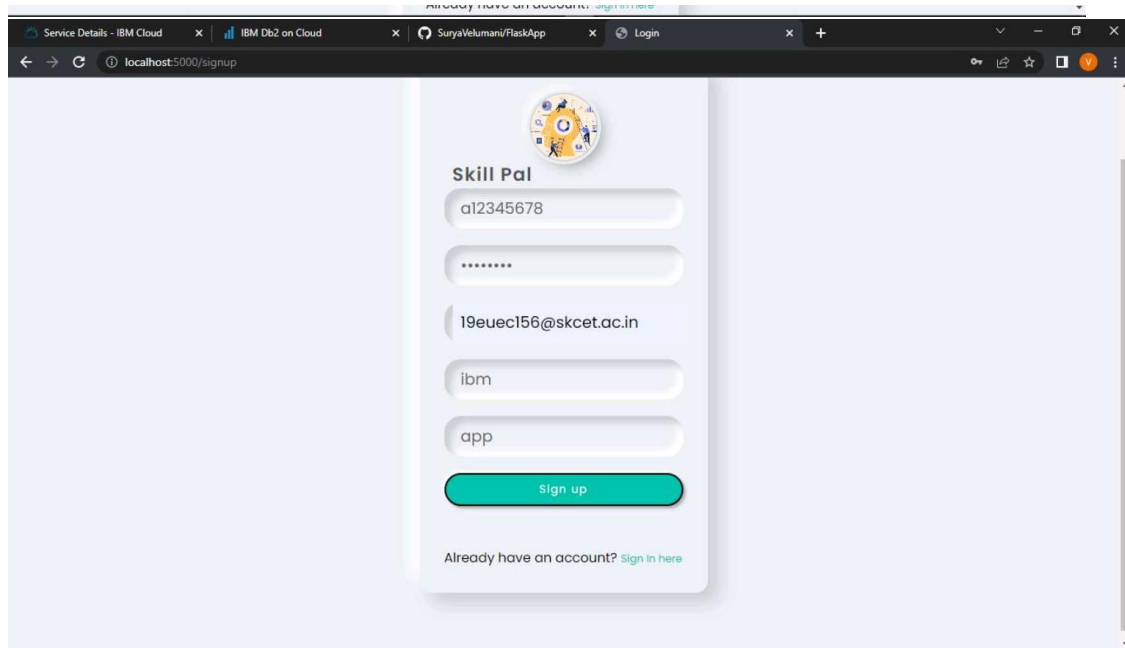
The login form for Skill Pal features a circular logo at the top. Below it, the title 'Skill Pal' is displayed. The form includes two input fields: 'Username' and 'Password'. A green 'Sign in' button is positioned below the password field. At the bottom, a link reads 'Don't have an account? [Sign Up here](#)'.



The sign-up form for Skill Pal is displayed on a light blue background. It features a circular logo at the top. Below it, the title 'Skill Pal' is displayed. The form includes 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 red error message 'Please fill out this field.' is visible next to the Password field.




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



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

← → ↻ localhost:5000/signup



**Skill Pal**

Username

a12345678

Password

\*\*\*\*\*

Email

19euecl56@skcet.ac.in

First Name

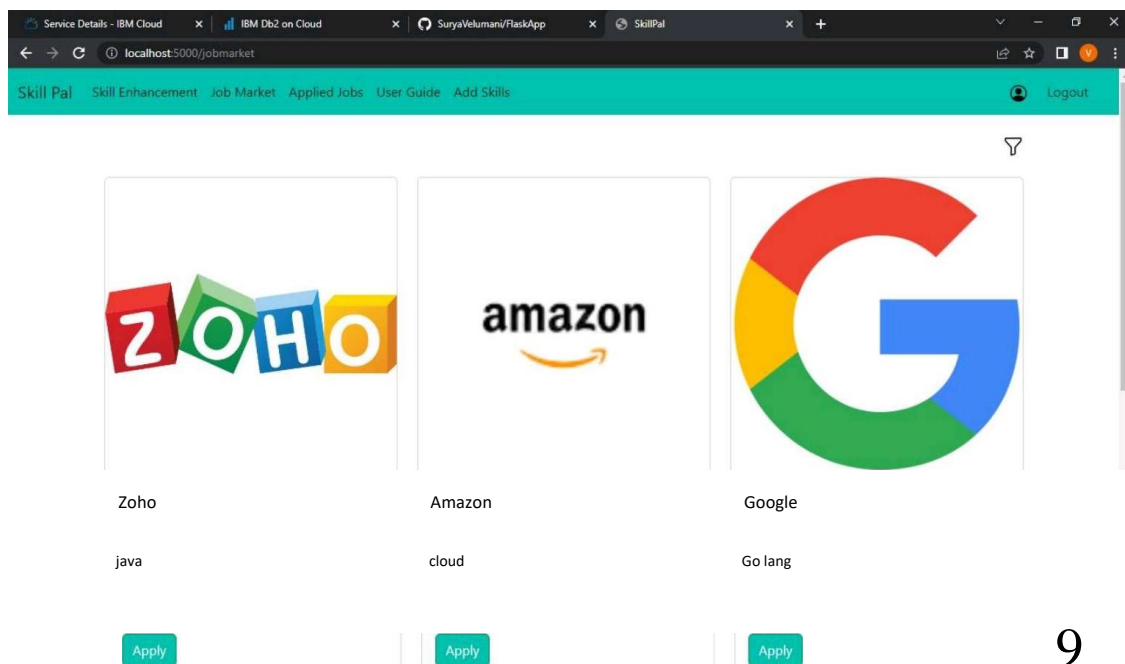
ibm

Last Name

app

[Sign up](#)

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




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

← → ↻ localhost:5000/jobmarket

Skill Pal Skill Enhancement Job Market Applied Jobs User Guide Add Skills [Logout](#)


🔍



Zoho

java


[Apply](#)



Amazon

cloud

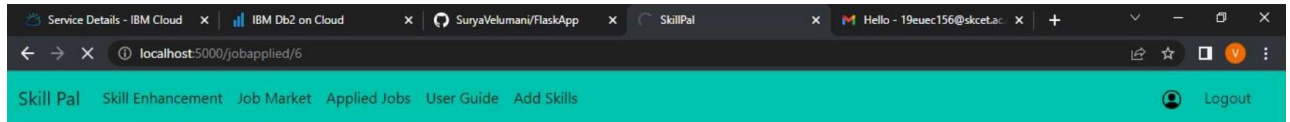
[Apply](#)



Google

Go lang

[Apply](#)



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

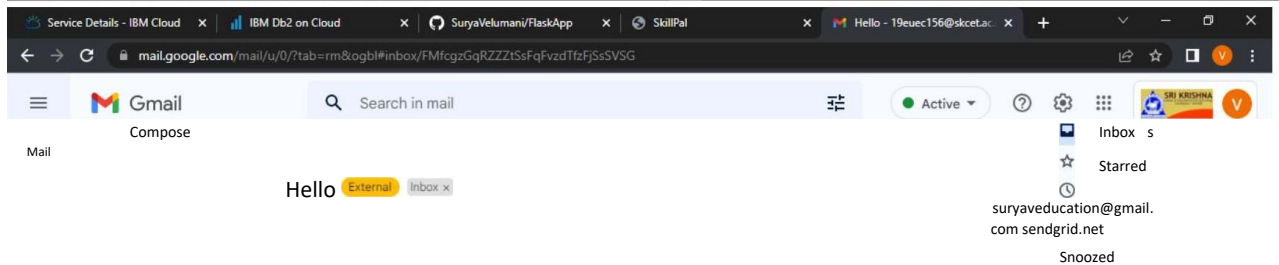
Preferred Location :

CBE

Submit form

Hi there, need help ?

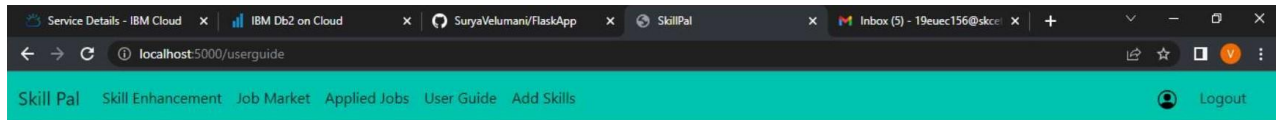
Waiting for integrations.au-syd.assistant.watson.appdomain.cloud...



Reply Forward

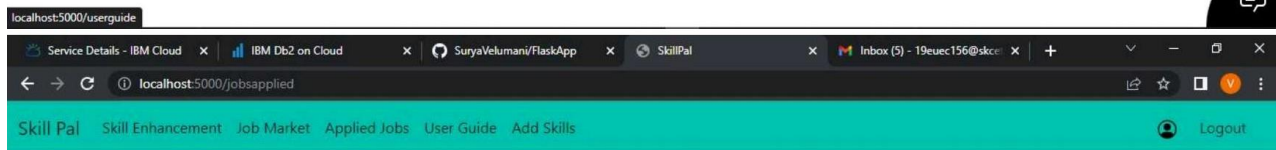
## Skill Pal Skill

## Jobs



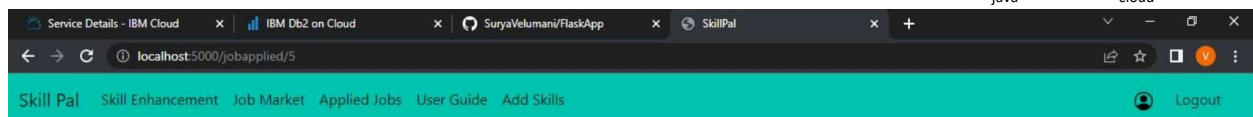
- Create an account using sign up and then sign in to your account.
- Use the Skill Enhancement tab to find courses and improve your skills.
- Add the skills that you have.
- Find all the jobs in job market.
- By clicking the filter icon on top right corner you can filter the jobs according to the skills 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 icon in bottom right corner of the page.

Hi there, need help ?



3:00 PM (0 minutes ago) ☆

Zoho	Amazon
5	6
java	cloud



Hi there, need help ?

Skill Pal Skill

Jobs

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

Hi, I am highly skilled in java, so I am interested in applying for this job.

5

Company .  
Zoho

Company Email :  
19euec156@skcet.ac.in

Portfolio Link :  
www.ram.com

Preferred Location :  
CBE

Submit form

Close

Hi there, need help ?



Service Details - x

IBM Db2 on Clo... x

SuryaVelumani/F... x

SkillPal x

SendGrid x

Your SendGrid p... x

SendGrid x

+ -

app.sendgrid.com/email\_activity/5L-zHms9QSy6OBVBrStEmg.filterdrecv-6df5cf8469-tzqj4-1-636E176D-16.0?filters=%5B%7B%22val%3A%5B%22suryaveducation%40gmail.co...

Surya Velumani

Activity Feed

Dashboard

Email API

Marketing

Design Library

Stats

Activity

Suppressions

Settings

Twilio SMS NEW

REPUTATION 83%

VIEW ACCOUNT USAGE

Search emails by:

To email address suryaveducation@gmail.com

Dates: 2022/11/11

STATUS MESSAGE

Delivered To: suryaveducation@gmail.com Hello

Delivered To: suryaveducation@gmail.com Hello

Email Information

Close

Details

To suryaveducation@gmail.com

From suryaveducation@gmail.com

Subject Hello

More Details

Event History

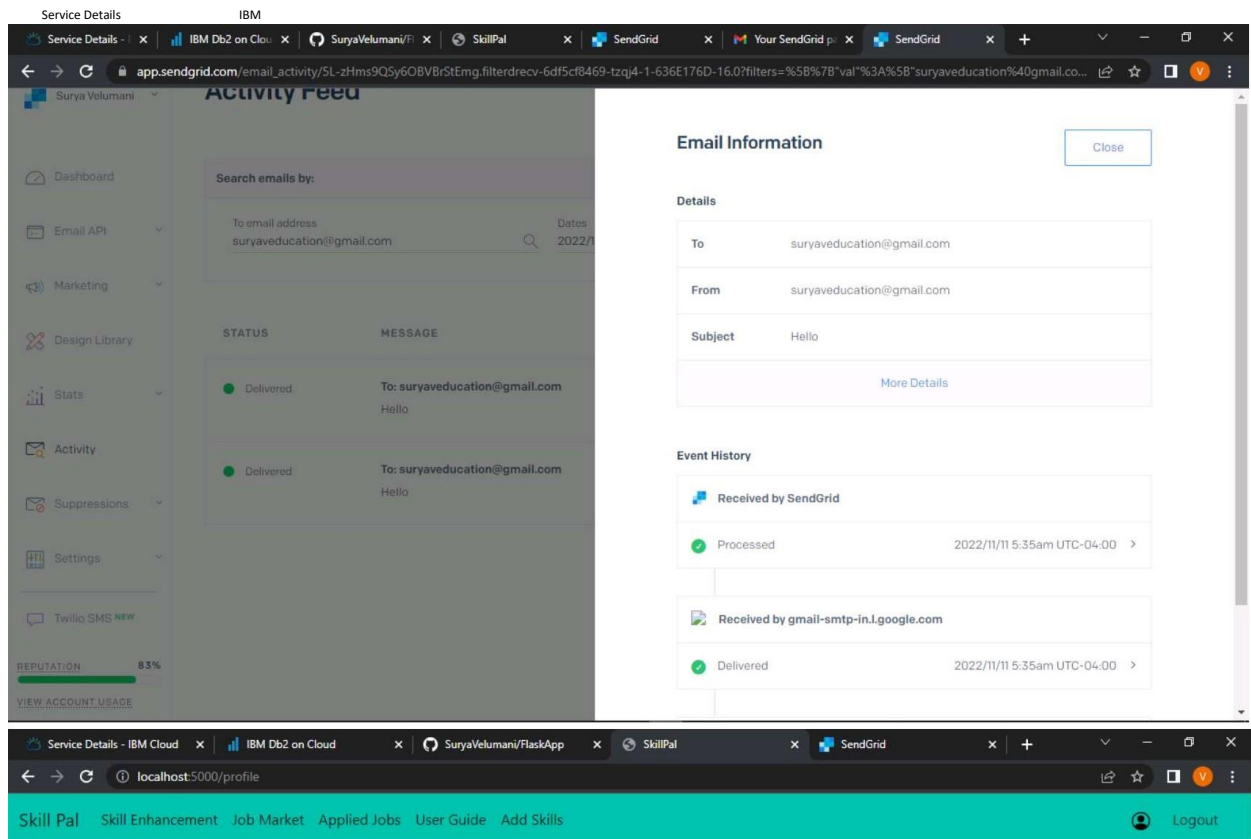
Received by SendGrid

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

Zoho



User Name :  
c12345678

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

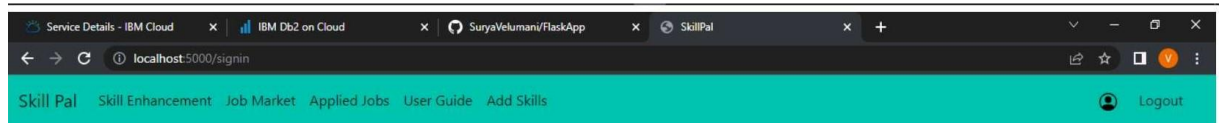
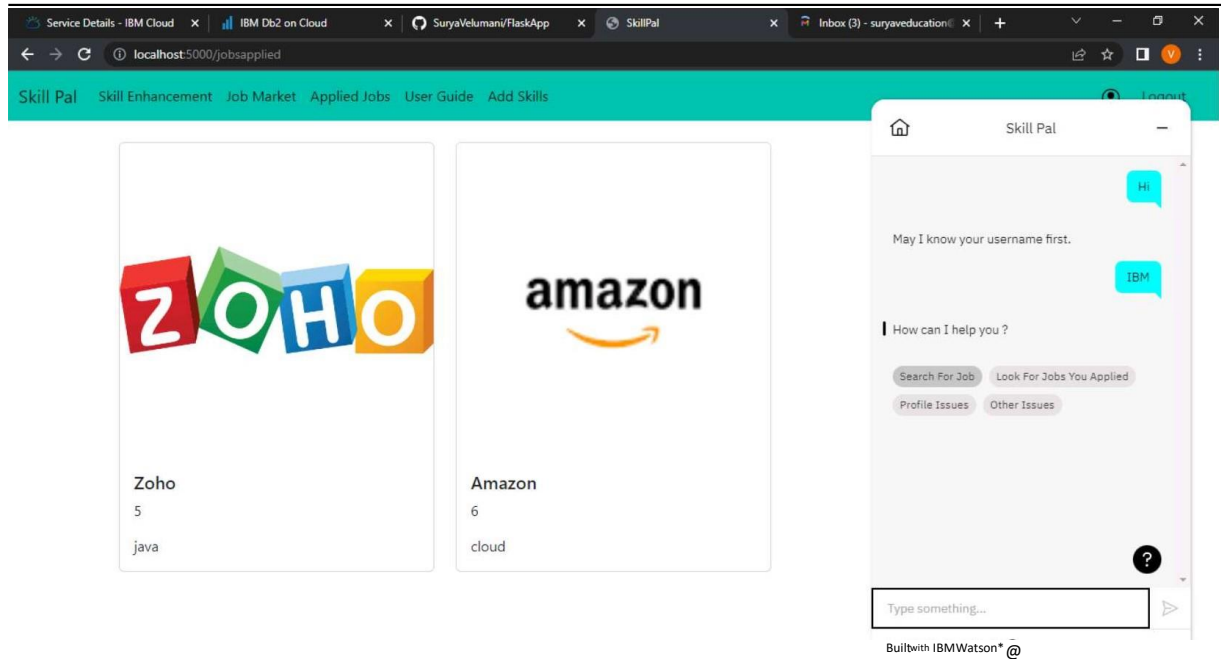
Email Id :  
suryaveducation@gmail.com

First Name :  
Surya

Last Name :  
V

Save





Please go through the courses below to enhance your skills

#### Java Courses

 Click

#### Python Courses

 Click

#### C++ Courses

 Click

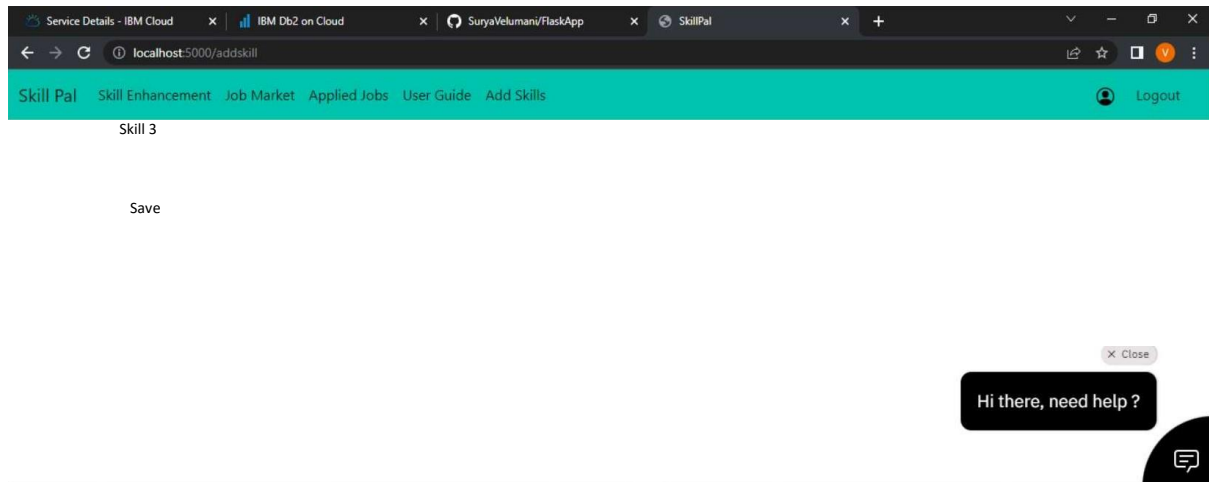
#### Javascript Courses

 Click

Skill 1

Java

Skill 2



## 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 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_bot03 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_bot03.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:      .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_bot03.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:      .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-
```

```
demoibmll name_file = session[ ']
```

```
username'] name_file += '.png'
```

```
filenameis = request.files['file ' ]
```

```
filepath = request.form[ 'filepath ' ] f =
```

```

        filepath f = f+filenameis.filename
        print("-      -----      -ll,f)
        multi_part_upload(bucket,name_file,
        f) returnredirect(url_for('dashboard'))
    if request.method == 'GET':
        return render_template('upload.html ')

```

```
mail = Mail(app) # instantiate the mail class
```

```

app.config[IMAIL
_SERVER']='smtp.sendgrid.net'
app.config[IMAIL_    PORT']    =    465
app.config[IMAIL_    USERNAME'] = 'apikey'
app.config[IMAIL_    USE_TLS']   =    False
app.config[IMAIL_USE_SSL'] = True mail =
Mail(app)
@app.route('/')
def    home():    return
    redirect(url_for('signin'))

@app.route(l/dashboardl)
def dashboard():
    return render_template('dashboard.html')
@app.route('/userguide')
def userguide():
    return render_template('userguide.html ')
@app.route('/addskill')
def addskill():
    skilll = ""ski112 = ""ski113 = ""user = session['username '] sql
    = "SELECT * FROM ACCOUNTSKILL WHERE USERNAME = ?"
    stmt    =    ibm_db.prepare(conn,    sql)
    ibm_db.bind_param(stmt,l,user)    ibm_db.execute(stmt)
    skillres = ibm_db.fetch_assoc(stmt) if skillres:
        skilll = skillres['SKILL1'] ski112 = skillres['SKILL2 ' ] ski113 = skillres['SKILL3 '
        ] print(skillres)    return    render_template(    '    addSkill.html    '    ,
        ski111=ski111,ski112=ski112,ski113=ski113) else :
        return render_templateCaddSkill.html ', ski111=ski111,ski112=ski112,ski113=ski113)

@app.route('/editskill',methods    'POST'])

```

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

```
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']
    ski1121 = request.form['ski112 1']
    ski1131 = request.form['ski113 1']
    print(skill11,"---",skill21,"--",skill31)
    sq - "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,ski1121)
    ibm_db.bind_param(stmt,3,ski1131)
    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']
    ski1122 = request.form['ski112 1'] ski1132 =
    request.form['ski113 1'] print("-----
",usernameskill ) sq = "INSERT INTO ACCOUNTSKILL
VALUES (?, ?, ?, ?)" stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,usernameskill)
    ibm_db.bind_param(stmt,2,ski1122)
    ibm_db.bind_param(stmt,3,ski1122)
    ibm_db.bind_param(stmt,4,ski1132)
    print("::::::::::::::::::::::::::",sql)
    ibm_db.execute(stmt) msg = "Saved Successfully !" return render_
emplate('addSkill.html',msg= msg, ski111=ski1112,ski112=ski1122,ski113=ski1132)
```

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

```
def jobmarket():
```

```
    jobids = []
    jobnames = []
    jobimages = []
    jobdescription = []
```

```
=
```

```

        JOBMARKET"
        = username = session['username'] print(username)
        #ibm_db.bind_param(stmt,l,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():
    skilll = ""ski112 = ""ski113 = ""user = session['username'] sql
    = "SELECT * FROM ACCOUNTSKILL WHERE USERNAME = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,l,user) ibm_db.execute(stmt)
    skillres = ibm_db.fetch_assoc(stmt) if skillres:
        skilll = skillres['SKILL1']
        ] ski112 =
        skillres['SKILL2']
        ski113 =
        skillres['SKILL3']
        print(skillres) jobids =
        [] jobnames = []
        jobimages = []
        jobdescription =[]

    sql = "SELECT * FROM

```

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

```
sql = "SELECT * FROM
JOBMARKET" stmt =
ibm_db.prepare(conn, sql)
username = session[ 'username ' ]
print(username)
#ibm_db.bind_param(stmt,l,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() ==
ski112.lower() or jobdescription[i].lower() == ski113.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.routeC/signin', methods =['GET','POST'])
def signin(): msg = ' ' if request.method
'POST':
username = request.form['username ' ]
password = request.form[ 'password ' ]
ACCOUNT WHERE username=?"
```

=

```

        =
        ibm_db.bind_param(stmt,l,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,l,username) ibm_db.execute(stmt) result =
        ibm_db.fetch_assoc(stmt) passWordInDb = result["UPASSWORD"] if
        passWordInDb == password: session['loggedin'] = True
            on['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,l,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',

```

```
sql "SELECT * FROM
```

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

```
usernameInUser=usernameInUser,userPassword=userPassword,userEmail=userEmail,firstName=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,lastName=lname)
```

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

```
def logout():
```

```
    session.pop('loggedin', None)
```

```
    session.pop('username', None)
```

```
    return redirect(url_for('signin'))
```

```
@app.route('/signup',methods=['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 = "INSERT INTO ACCOUNT WHERE username =?"
```

```
        =
```

```
        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,l,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,l,jobid) ibm_db.execute(stmt) result =
ibm_db.fetch_assoc(stmt) jobemail print('I--- ,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 =
        il = result['COMPANY_EMAIL']
        -----JOB APPLIED-----
    ibm_db.prepare(conn, passCheck)
    ibm_db.bind_param(stmt,l,username) ibm_db.execute(stmt) result
    = ibm_db.fetch_assoc(stmt) fromEmail = result["EMAILID"]

```

```

sql "SELECT * FROM

```





<https://github.com/IBM-EPBL/IBM-Project-24324-1659941545>