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 Linkedln profiles". In: Proc. of Int. Conf. on Research Challenges in Information Science (2014), pp. 1—6. issn: 21511357. doi:
10.1109/RClS.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- O.
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

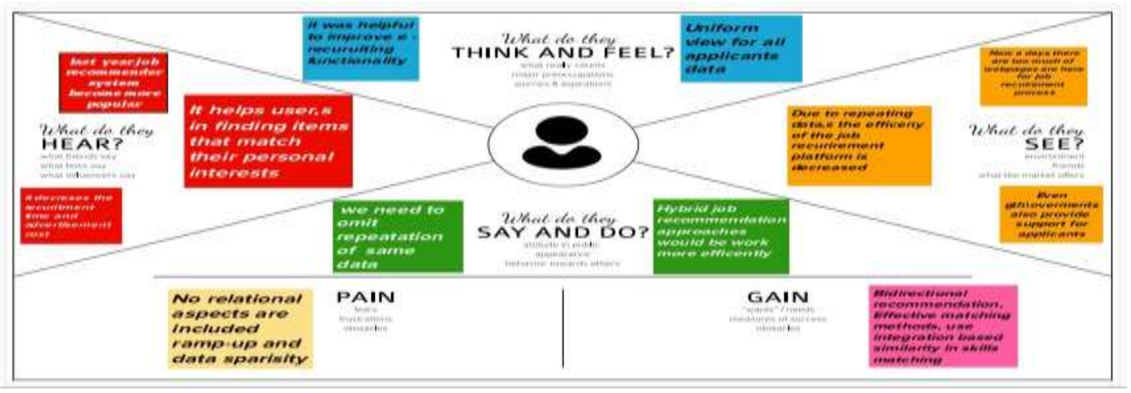
An empathy map is a collaborative visualization used to articulate what we know about a particular type of user. It externalizes knowledge about users in order to

1) Create a shared understanding of user needs, and

2) Aid Decision Making

IDEATION AND BRAINSTROMING

JOB/SKILL RECOMMENDER APPLICATION


It was helpful to improve e-recuruiting functionality
What do they THINK AND FEEL?
Uniform view for all applicants data
last year job recommender system become more popular
It helps user,s in finding items that match their personal interests
What do they HEAR?
it decreases the recruitment time and advertisement cost
Due to repeating data,s the efficeny of the job recuirement platform is decreased
What do they SEE?
Now a days there are too much of webpages are here for job recuirement process
Even governments also provide support for applicants
we need to omit repeatation of same data
What do they SAY AND DO?
Hybrid job recommendation approaches would be work more efficently
No relational aspects are included ramp-up and data sparisity
PAIN
GAIN
Bidirectional recommendation, Effective matching methods, use integration based similarity in skills matching

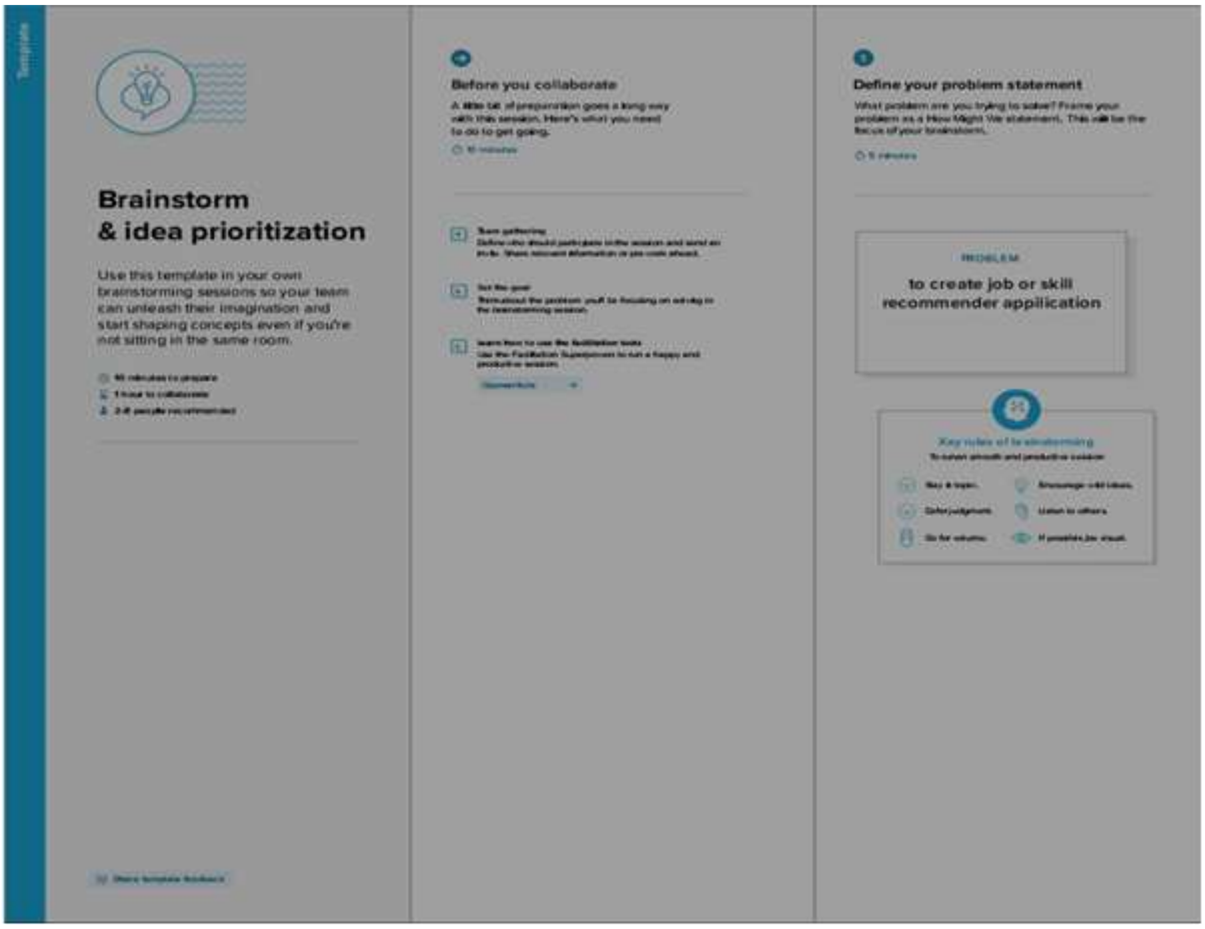
Pain
Hard to find their food interests
Uncomfortable paying cash
Looking for food that matches interest
Gain
Comfortable
Accuracy

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-thebox 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

Share template feedback

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

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

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

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

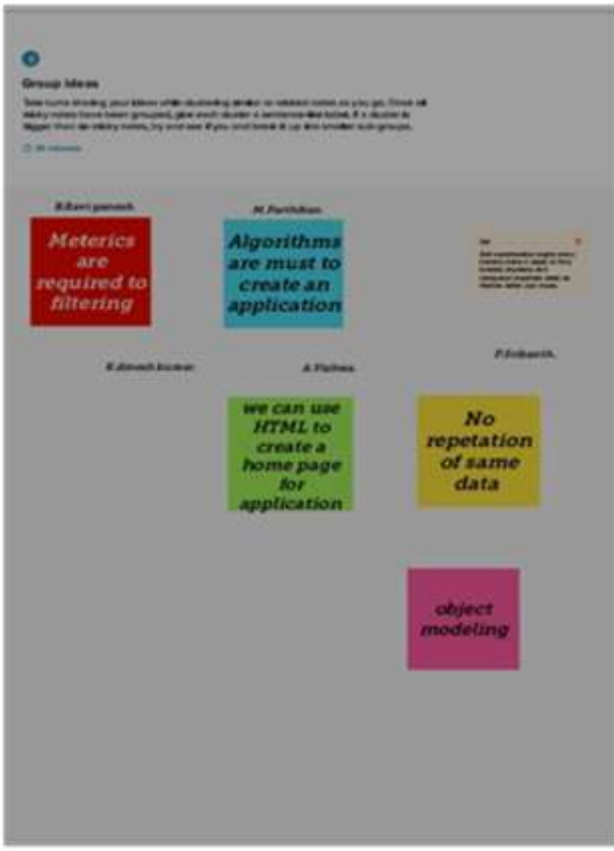
to create job or skill recommender appilication

Key rules of brainstorming
To run an smooth and productive session

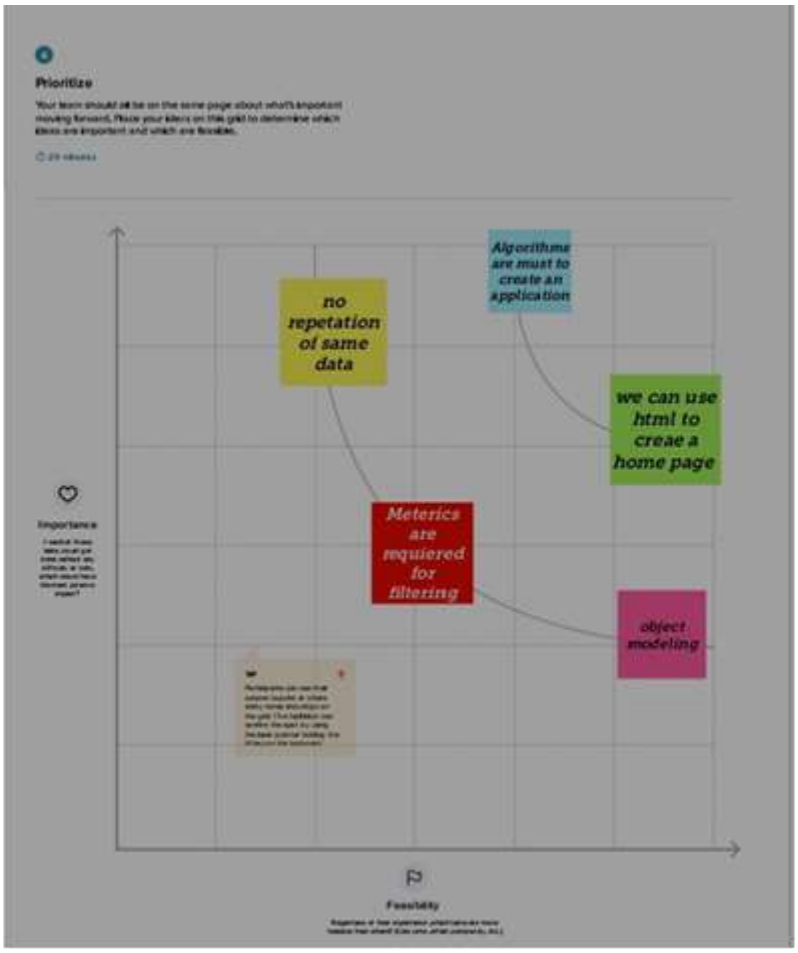
Stay in topic.	Encourage wild ideas.
Defer judgment.	Listen to others.
Go for volume.	If possible, be visual.

STEP2: 2:

Step-2: Brainstorm, Idea Listing and Grouping


Group Ideas
Meterics are required to filtering
Algorithms are must to create an application
we can use HTML to create a home page for application
No repetation of same data
object modeling

Step-3: Idea Prioritization


Prioritize
no repetation of same data
Algorithms are must to create an application
we can use html to creae a home page
Importance
Meterics are requiered for filtering
object modeling
Feasibility

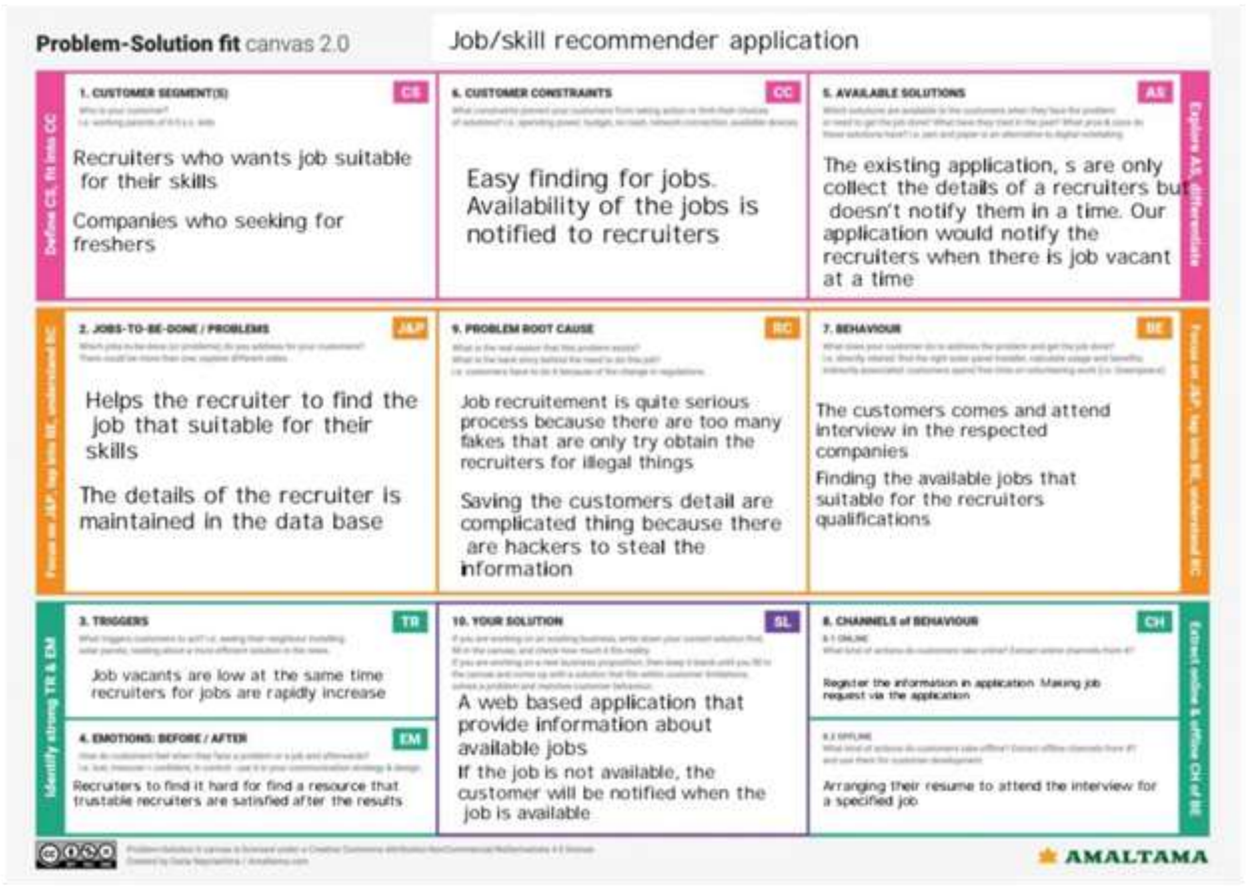
PROPOSED SOLUTION

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

4. REQUIREMENT ANALYSIS

Problem solution Fit:

Template:



Problem-Solution fit canvas 2.0

Job/skill recommender application

1. CUSTOMER SEGMENT(S) CS

Recruiters who wants job suitable for their skills

Companies who seeking for freshers

6. CUSTOMER CONSTRAINTS CC

Easy finding for jobs.
Availability of the jobs is notified to recruiters

5. AVAILABLE SOLUTIONS AS

The existing application, s are only collect the details of a recruiters but doesn't notify them in a time. Our application would notify the recruiters when there is job vacant at a time

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

Helps the recruiter to find the job that suitable for their skills

The details of the recruiter is maintained in the data base

9. PROBLEM ROOT CAUSE RC

Job recruitement is quite serious process because there are too many fakes that are only try obtain the recruiters for illegal things

Saving the customers detail are complicated thing because there are hackers to steal the information

7. BEHAVIOUR BE

The customers comes and attend interview in the respected companies

Finding the available jobs that suitable for the recruiters qualifications

3. TRIGGERS TR

Job vacants are low at the same time recruiters for jobs are rapidly increase

4. EMOTIONS: BEFORE / AFTER EM

Recruiters to find it hard for find a resource that trustable recruiters are satisfied after the results

10. YOUR SOLUTION SL

A web based application that provide information about available jobs

If the job is not available, the customer will be notified when the job is available

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

Register the information in application. Making job request via the application

8.2 OFFLINE

Arranging their resume to attend the interview for a specified job

AMALTAMA

FUNCTIONAL REQUIREMENT

	Functional Requirement (Epic)	Sub Requirement (Story I 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
	IJser 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.

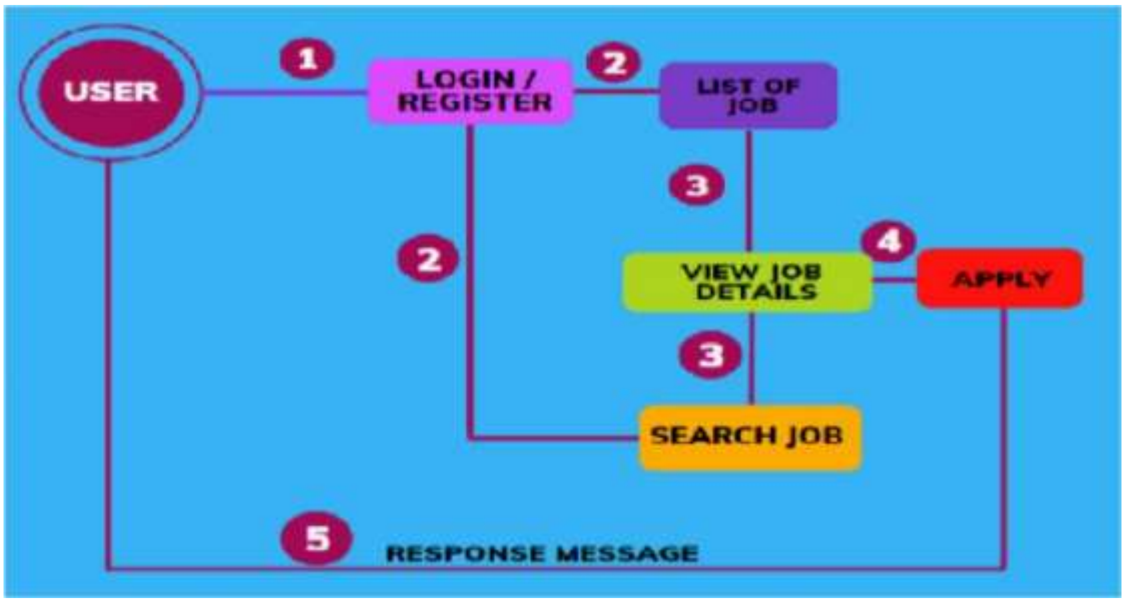
NON FUNCTIONAL REQUIREMENTS

Non functional Requirements are :

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

5 PROJECT DESIGN

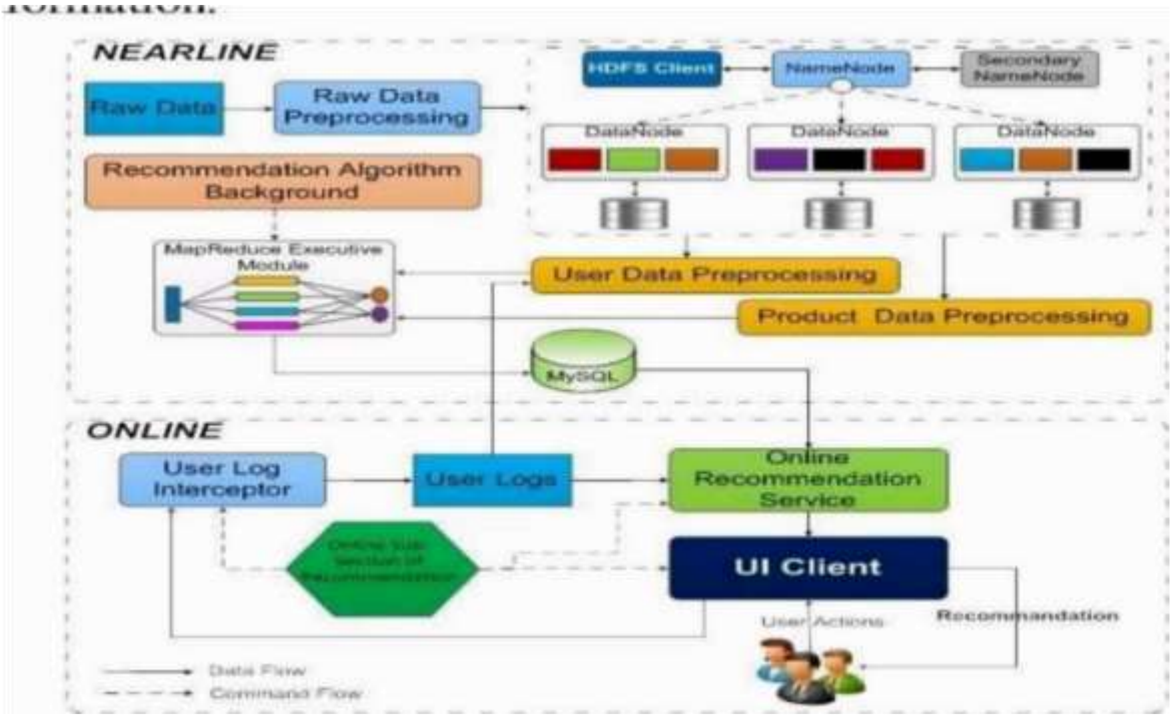
DATAFLOW DIAGRAM



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



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	Ravi ganesh.B, Parthiban.M,Dinesh Kumar.K, Vishwa.A,Srikanth.P
SPRINT 2	Connect application to ibm db2	Ravi ganesh.B, Parthiban.M,Dinesh Kumar.K, Vishwa.A,Srikanth.P
SPRINT 3	Integrate ibm Watson assisstant	Ravi ganesh.B, Parthiban.M,Dinesh Kumar.K, Vishwa.A,Srikanth.P
SPRINT 4	Containerize the app and Deploy the application in ibm cloud	Ravi ganesh.B, Parthiban.M,Dinesh Kumar.K, Vishwa.A,Srikanth.P

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[l])
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['SKlLLI ' ]
   ski112          = skillres['SKlLL2 1 ] ski113          =
   skillres['SKlLL3  1  ]  print(skillres)  jobids  =  []  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[O])
      jobnames.append(joblist[l])
      jobimages.append(joblist[2])
      jobdescription.append(joblist[3]) joblist
      = ibm_db.fetch_tuple(stmt)
      jobinformation = [l

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

    for i in range(size): col =
      []
print("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@",jobdescription[i])
                    skilll.lower()or jobdescription[i].lower()== ski112.lower() or
if jobdescription[i].lower() 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_templateCjobmarket.html', jobinformation = jobinformation)
```

Feature 2:

ChatBot (using IBM Watson)

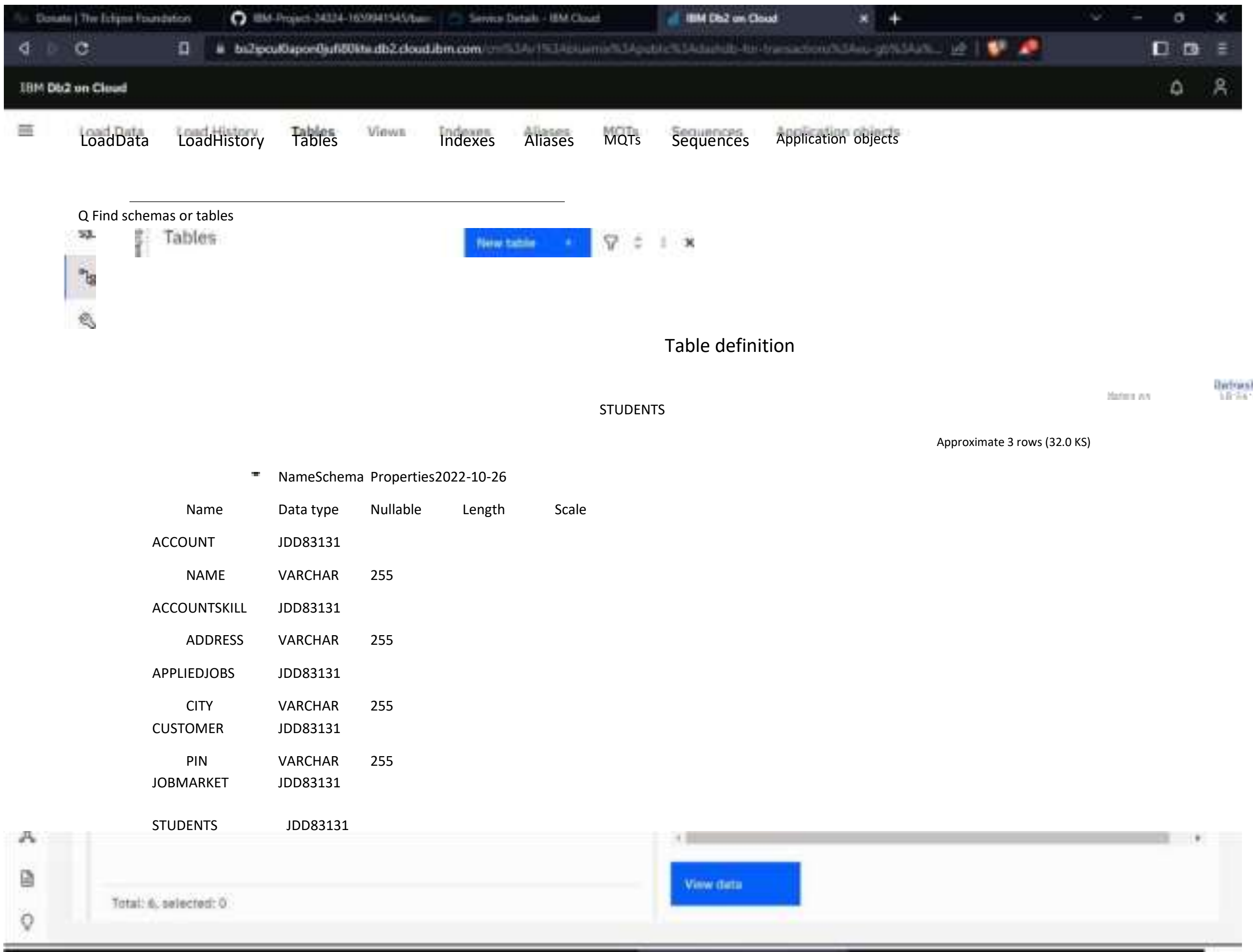
This chat bot feature provides help tooltip for end users if any help needed for users

```
<script> window.watsonAssistantChatOptions = { integrationlD: "9be41b76-06bO-426f8469-
      962f2963cdb6", // The ID of this integration. region: "au-syd", // The region your
      integration is hosted in.
        servicelnstancelD: "76838ca2-a227-4f56-b180-94f01901cdbf", // The ID of your service instance.
        onLoad: function(instance) { instance.render(); }

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

Database Schema:

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



LoadData LoadHistory Tables Indexes Aliases MQTs Sequences Application objects

Q Find schemas or tables

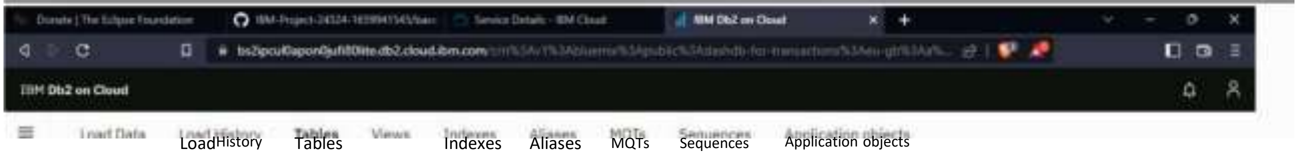
Table definition

STUDENTS

Approximate 3 rows (32.0 KS)

NameSchema Properties2022-10-26

Name	Data type	Nullable	Length	Scale
ACCOUNT	JDD83131			
NAME	VARCHAR	255		
ACCOUNTSKILL	JDD83131			
ADDRESS	VARCHAR	255		
APPLIEDJOBS	JDD83131			
CITY	VARCHAR	255		
CUSTOMER	JDD83131			
PIN	VARCHAR	255		
JOBMARKET	JDD83131			
STUDENTS	JDD83131			



LoadHistory Tables Indexes Aliases MQTs Sequences Application objects

JOBMARKET JDD83131

STUDENTS JDD83131

Total: 6, selected: o

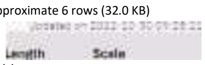
Donate | The Eclipse Foundation IBM-Project-24324-1659941545/base Service Details - IBM Cloud

IBM Db2 on X

IBM Db2 on Cloud

Q Find schemas or

JOBMARKET Approximate 6 rows (32.0 KB)



Properties

Data type Nullable

tables Refresh

JOBID INTEGER

SQLTable definition

ACCOUNTSKILL JDD83131

JOBCOMPANY VARCHAR 255

APPLIEDJOBS JDD83131

ACCOUNT JDD83131

JOBIMAGE VARCHAR 500

C] CUSTOMER JDD83131

JOBSKILL VARCHAR 255

COMPANY_EMAIL

VARCHAR 255

Q Find schemas or tables Refresh

SQL Tables

Table definition

CUSTOMER Approximate O rows (O RE) Name Schema Properties

Updated on 2022-10-29 04:42:22

Data type Nullable Length Scale

ACCOUNT JDD83131

CUSTOMERID INTEGER

ACCOUNTSKILL JDD83131

LASTNAME VARCHAR 255

APPLIEDJOBS JDD83131

FIRSTNAME VARCHAR 255

CUSTOMER JDD83131

ADDRESS VARCHAR 255

JOBMARKET JDD83131

CITY VARCHAR 255

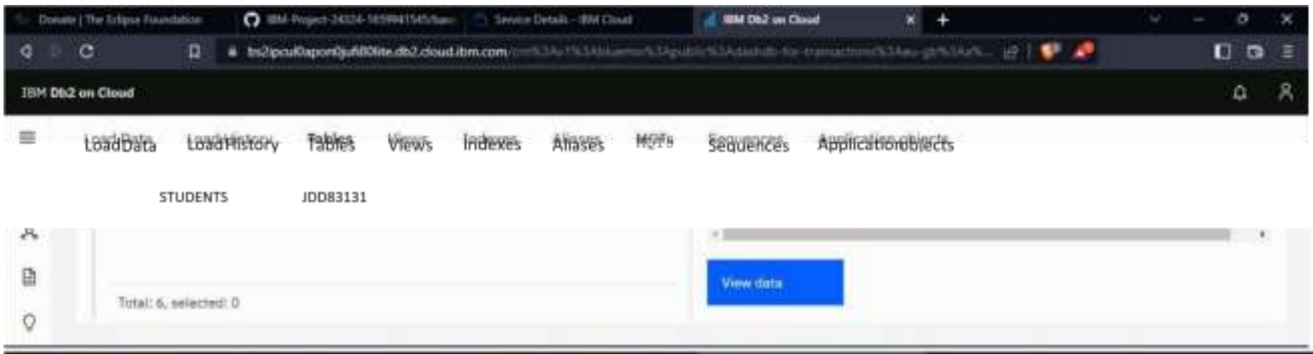
JOBMARKET JDD83131

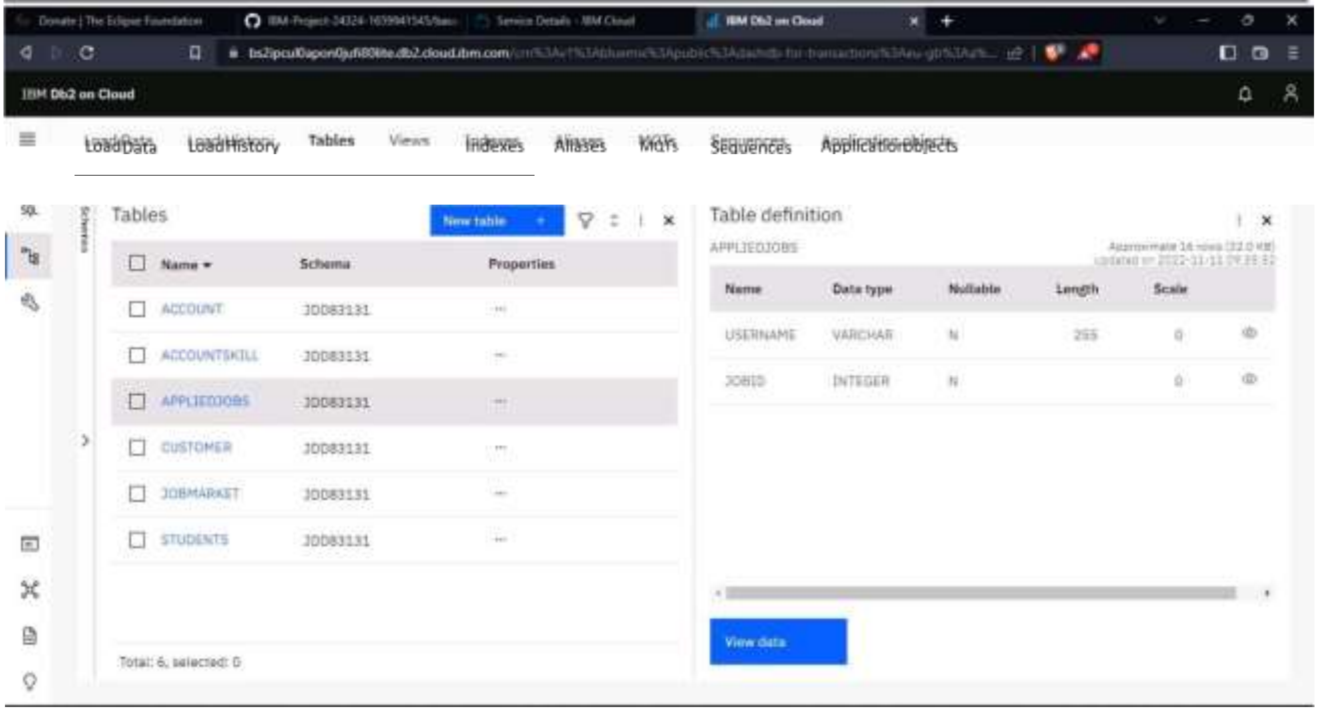
STUDENTS
JDD83131

Total: 6, selected:
O

Donate | The Eclipse Foundation IBM-Project-24324-1659941545/base Service Details - IBM Cloud IBM Db2 on X

IBM Db2 on Cloud





APPLIEDJOBS Approximate 16 rows (32.0 KB)

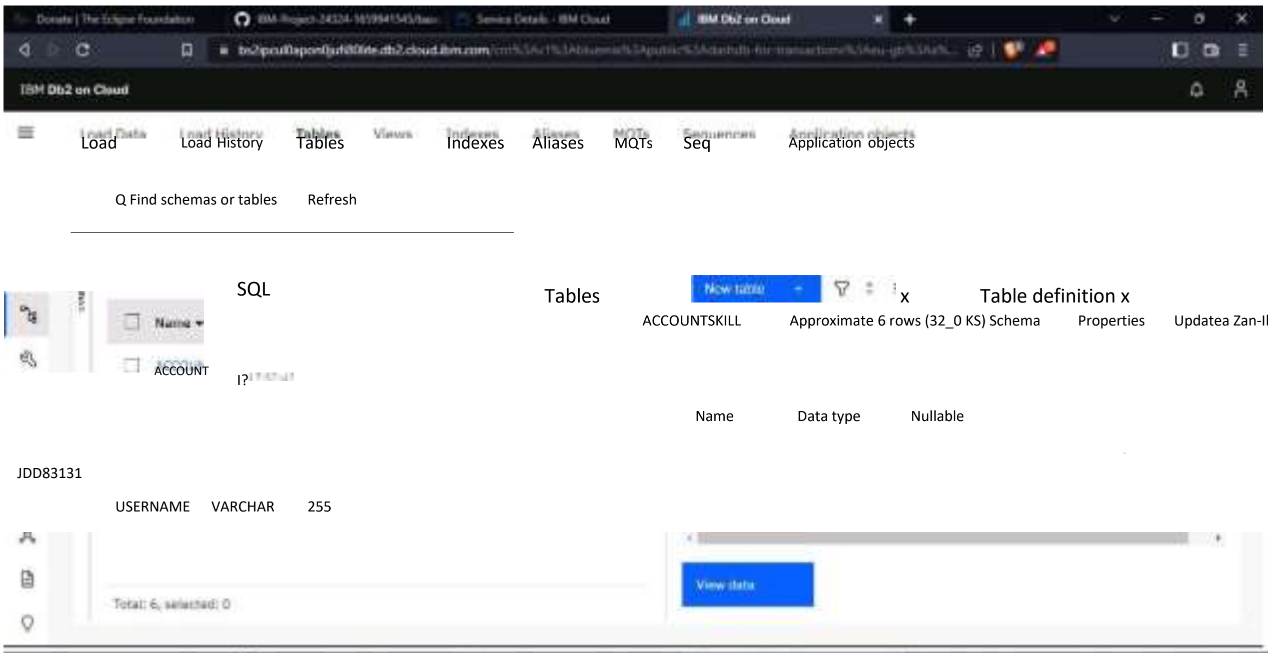
Q Find schemas or tables Refresh

Name	Schema	Properties	2022-11-21	
Name	Data type	Nullable	Length	Scale
ACCOUNT	JDD83131			
USERNAME	VARCHAR	255		
ACCOUNTSKILL		JDD83131		
JOBID	INTEGER			
APPLIEDJOBS	JDD83131			
[3 CUSTOMER	JDD83131			
JOBMARKET	JDD83131			
STUDENTS	JDD83131			

Total: 6, selected: 0

Donate | The Eclipse Foundation IBM-Project-24324-1659941545/base Service Details - IBM Cloud IBM Db2 on X

IBM Db2 on Cloud



Load Load History Tables Indexes Aliases MQTs Seq Application objects

Q Find schemas or tables Refresh

SQL

Tables

ACCOUNTSKILL

Approximate 6 rows (32_0 KS) Schema Properties Updatea Zan-II-

x Table definition x

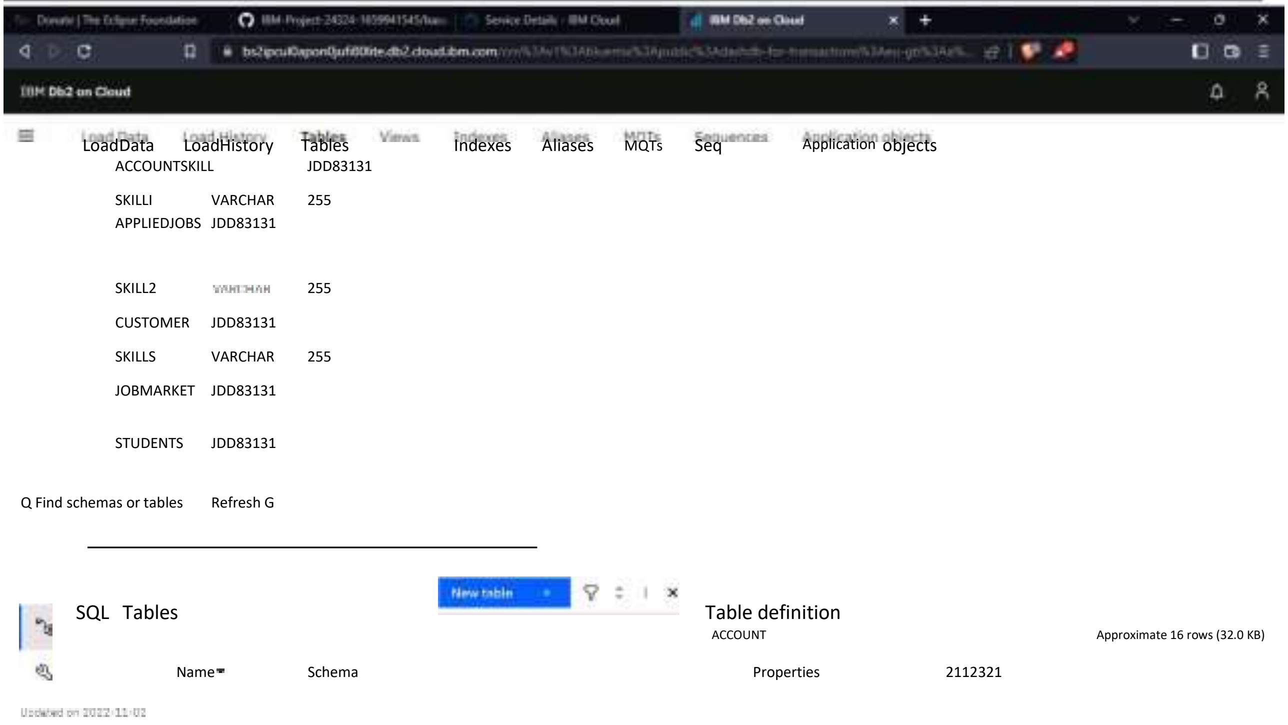
ACCOUNT

I?

Name	Data type	Nullable

JDD83131

USERNAME VARCHAR 255



LoadData LoadHistory Tables Indexes Aliases MQTs Seq Application objects

ACCOUNTSKILL JDD83131

SKILLI VARCHAR 255

APPLIEDJOBS JDD83131

SKILL2 VARCHAR 255

CUSTOMER JDD83131

SKILLS VARCHAR 255

JOBMARKET JDD83131

STUDENTS JDD83131

Q Find schemas or tables Refresh G

SQL Tables

Table definition

ACCOUNT

Approximate 16 rows (32.0 KB)

Name Schema

Properties 2112321

Updated on 2022-11-02

Name	Data type	Nullable	Length	Scale

JOBMARKET JDD83131

STUDENTS JDD83131

Total: 6, selected: 0

Donate | The Eclipse Foundation IBM-Project-24324-1659941545/base Service Details - IBM Cloud IBM Db2 on X

IBM Db2 on Cloud

ACCOUNT	JDD83131	
USERNAME	VARCHAR	255
C] ACCOUNTSKILL		JDD83131
UPASSWORD	VARCHAR	255
APPLIEDJOBS	JDD83131	
EMAILID	VARCHAR	255
CUSTOMER	JDD83131	
LASTNAME	VARCHAR	255
FIRSTNAME	VARCHAR	255

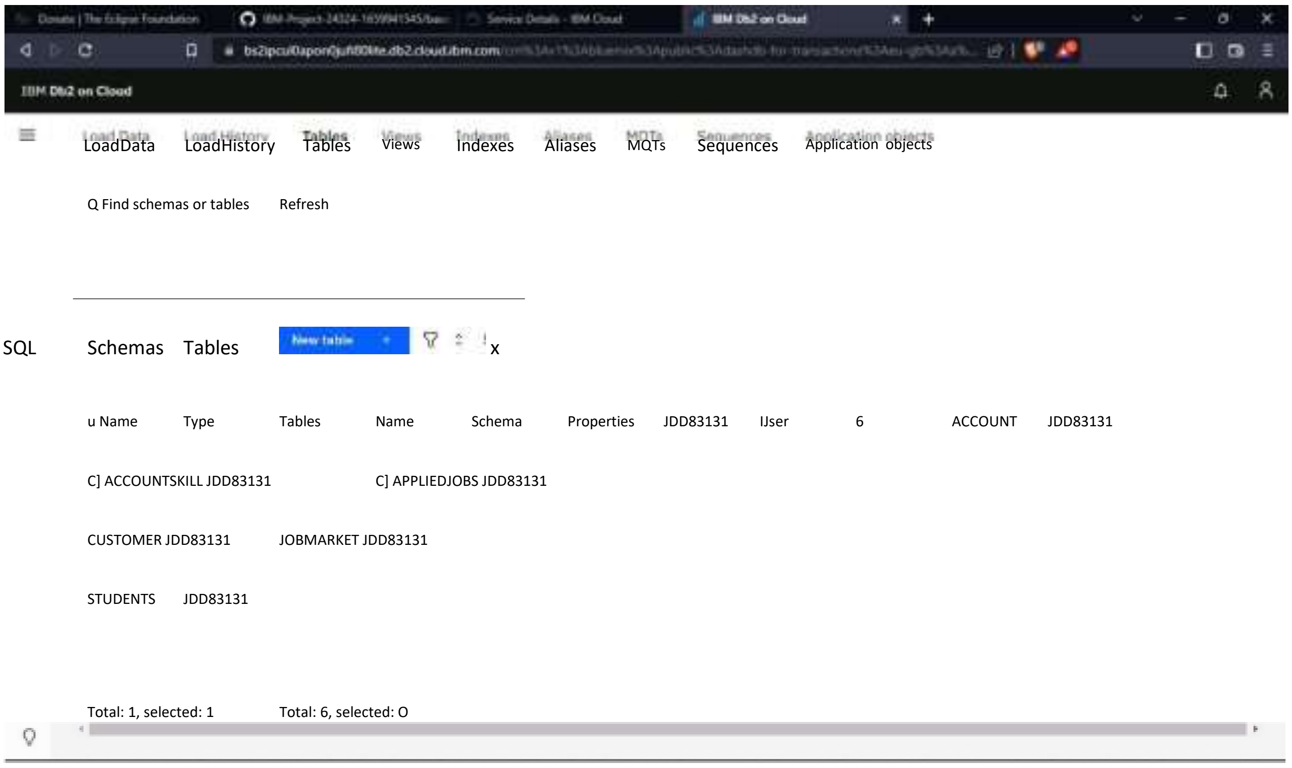
O

JOBMARKET	JDD83131
STUDENTS	JDD83131

Total: 6, selected: O

Donate | The Eclipse Foundation IBM-Project-24324-1659941545/base Service Details IBM Db2 on

IBM Db2 on Cloud



LoadData LoadHistory Tables Views Indexes Aliases MQTs Sequences Application objects

Q Find schemas or tables Refresh

SQL Schemas Tables New table x

u Name Type Tables Name Schema Properties JDD83131 IJser 6 ACCOUNT JDD83131

C] ACCOUNTSKILL JDD83131 C] APPLIEDJOBS JDD83131

CUSTOMER JDD83131 JOBMARKET JDD83131

STUDENTS JDD83131

Total: 1, selected: 1 Total: 6, selected: O

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.

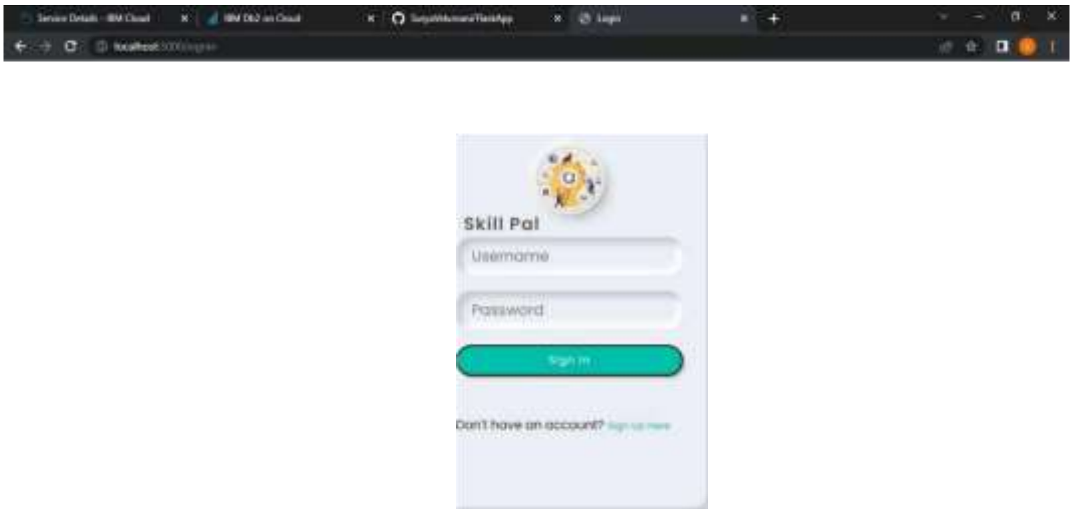
User Acceptance Testing:

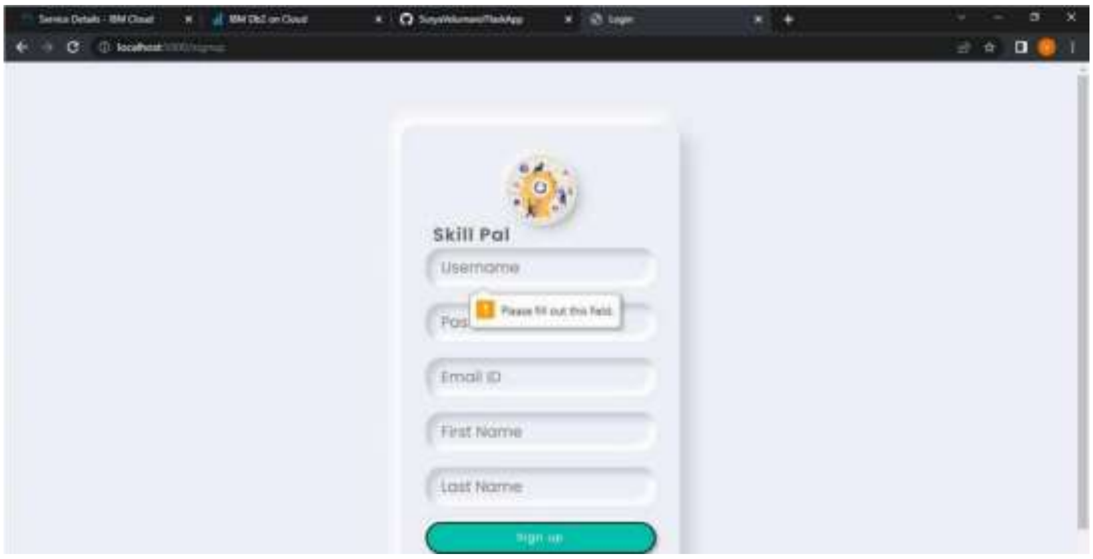
Service Details - IBM on Cloud x

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.

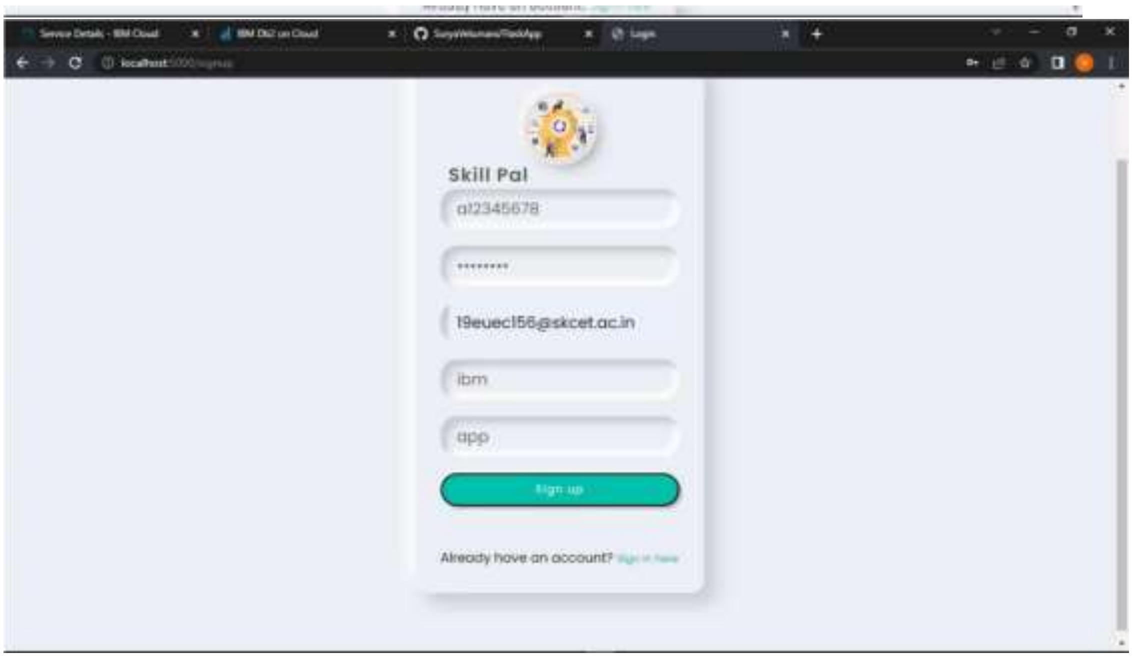
9.RESULTS

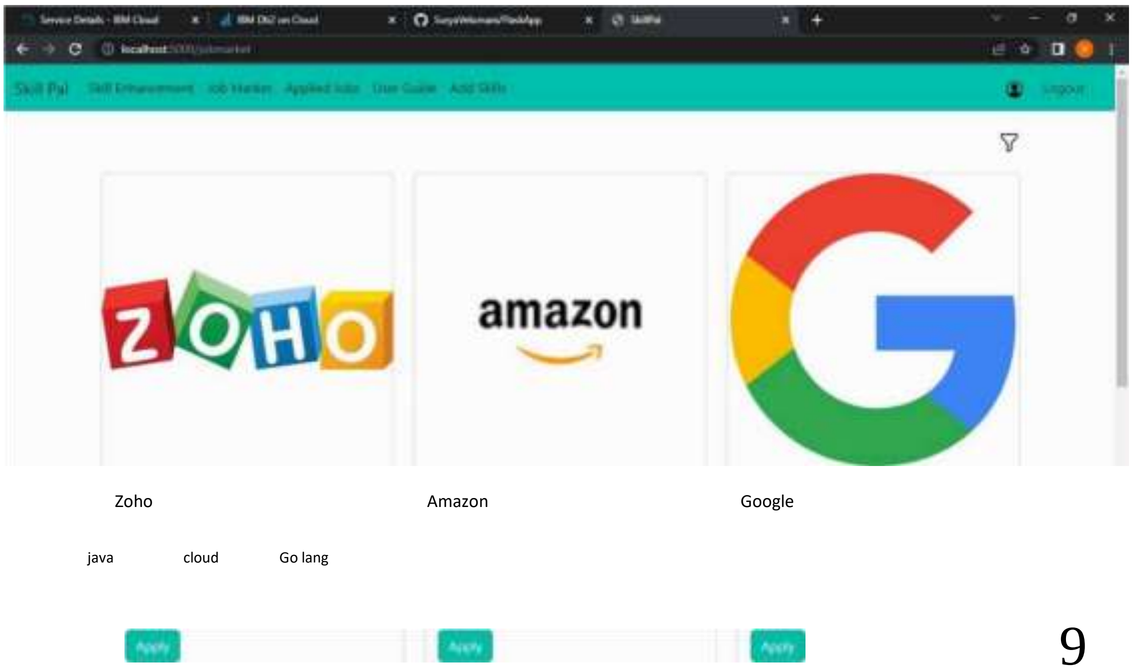
Perfomance Metrics:





Already have an account? Sign In here





Zoho Amazon Google

java cloud Go lang

9

Service Details - IBM IBM on Cloud X

Skill Pal Skill Jobs

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

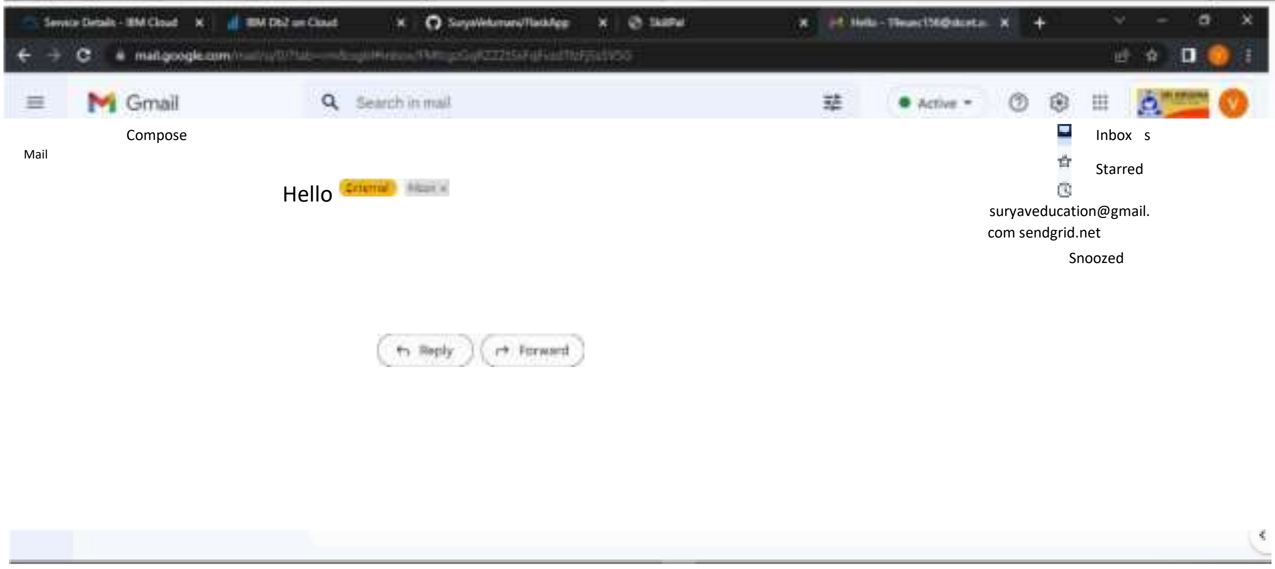
Portfolio Link :

www.demoPortfolio.com

Preferred Location :

CBE

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



Compose

Mail

Inbox s

Starred

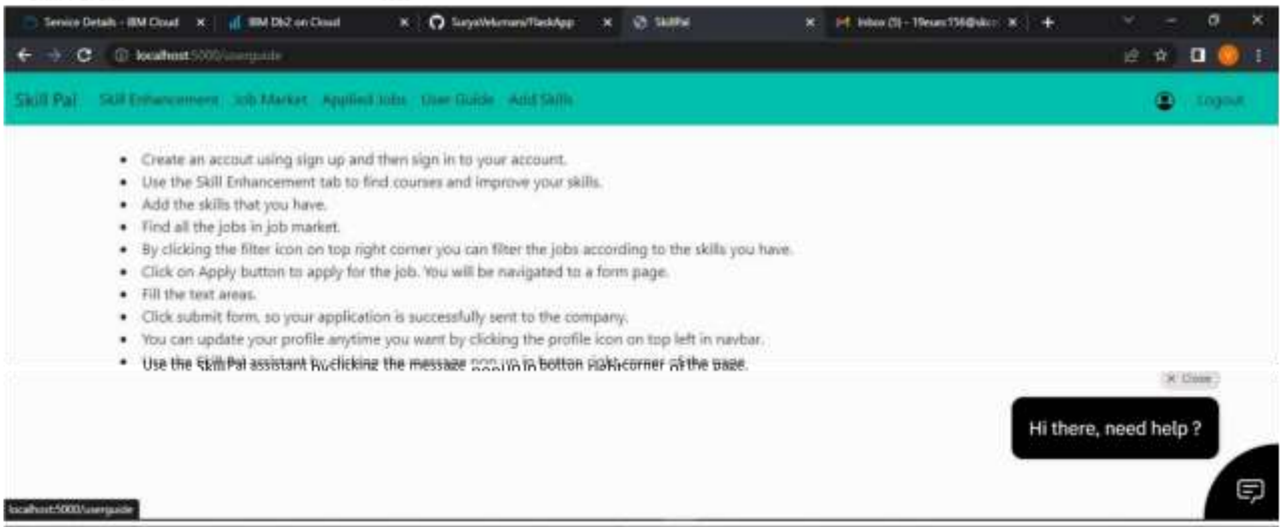
Hello

suryaveducation@gmail.com sendgrid.net

Snoozed

Service Details - IBM IBM

Skill Pal Skill Jobs


Service Details - IBM Cloud | IBM Db2 on Cloud | SuryaVelumani/FlaskApp | SkillPal | Inbox (5) - 19eucs156@skc

localhost:5000/userguide

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

- Create an accout 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 SkillPal assistant by clicking the message popup in bottom right corner of the page.

Service Details - IBM Cloud | IBM Db2 on Cloud | SuryaVelumani/FlaskApp | SkillPal | Inbox (5) - 19eucs156@skc

localhost:5000/jobsapplied

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

Applicant Email : suryaueducatign@gmail.com

Sent

Drafts

About Me
hi

More

amazo

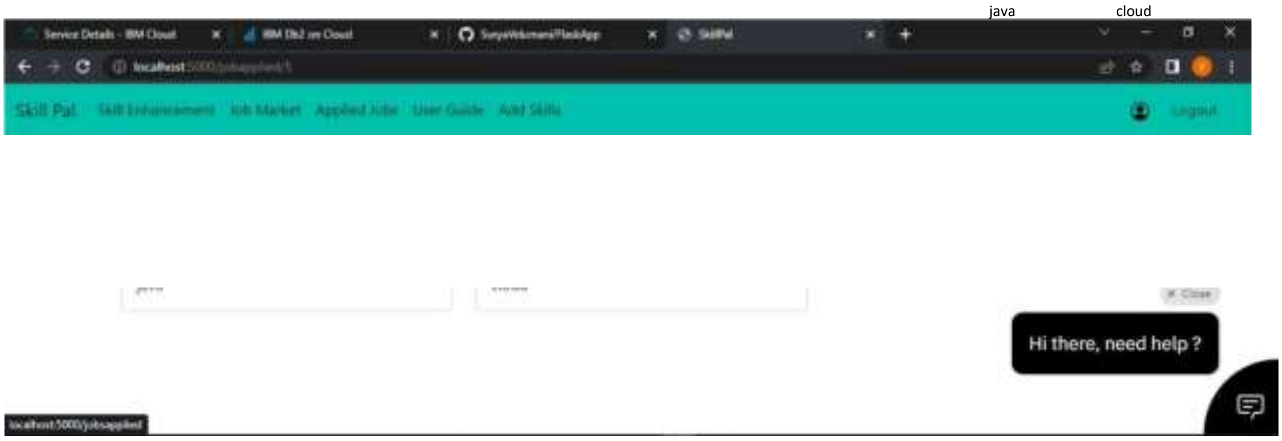
Portfolio Link

n

Labels Prefered City 1 of 7,902

Zoho Amazon
5 6
java cloud

3:00 PM (O minutes ago) ☆


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

localhost:5000/jobsapplied/5

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

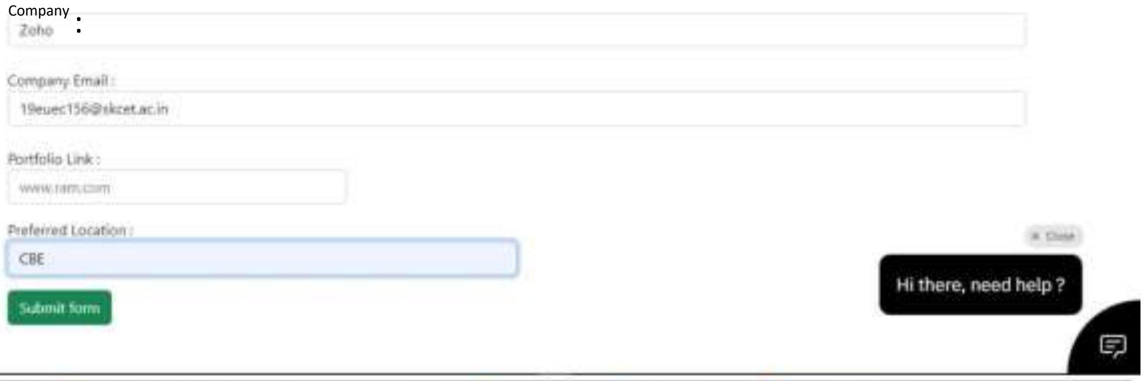
Service Details - IBM on Cloud X

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 :

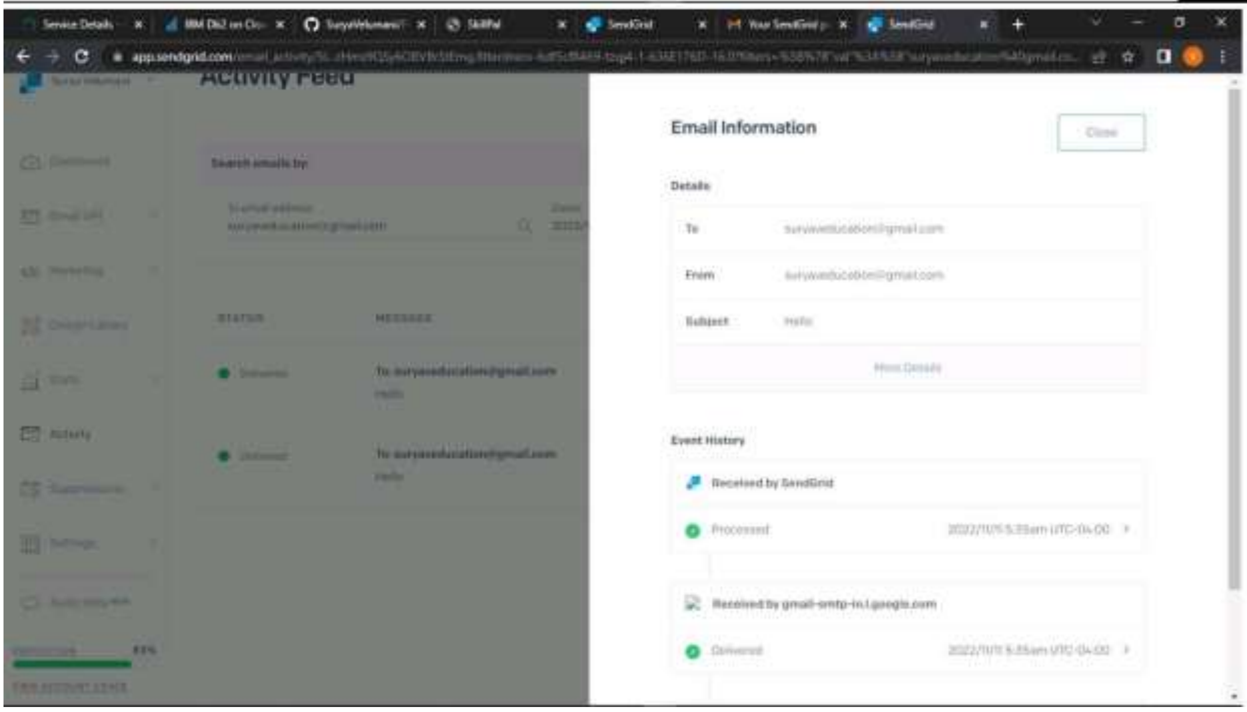
19euec156@skcet.ac.in

Portfolio Link :

www.ram.com

Preferred Location :

CBE



Activity Feed

Email Information

Details

To suryaveducation@gmail.com

From suryaveducation@gmail.com

Subject Hello

Event History

Received by SendGrid

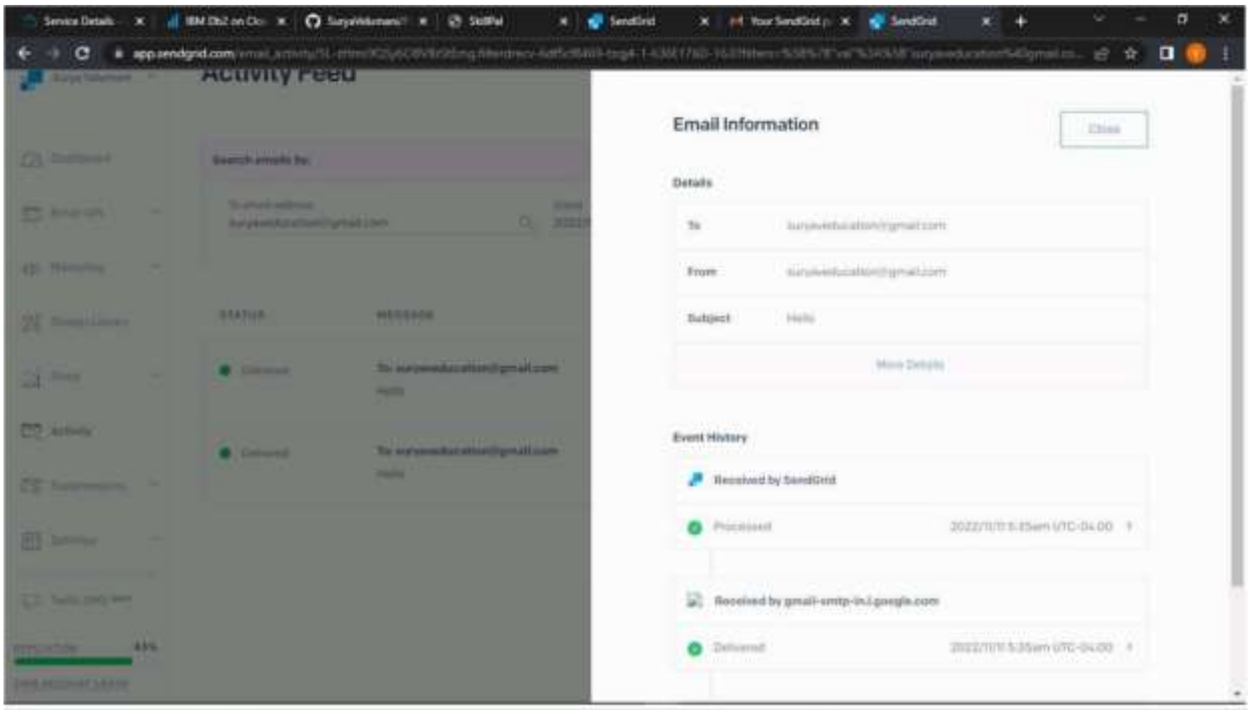
Processed 2022/11/11 5:33am UTC-04:00

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

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

Zoho

Service Details IBM



Activity Feed

Email Information

Details

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

More Details

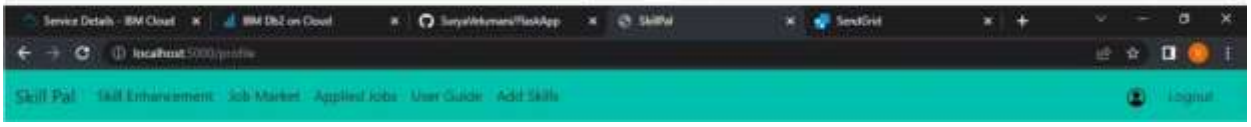
Event History

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



User Name :
c12345678

Password :

••••••••

Email Id :

suryaveducation@gmail.com

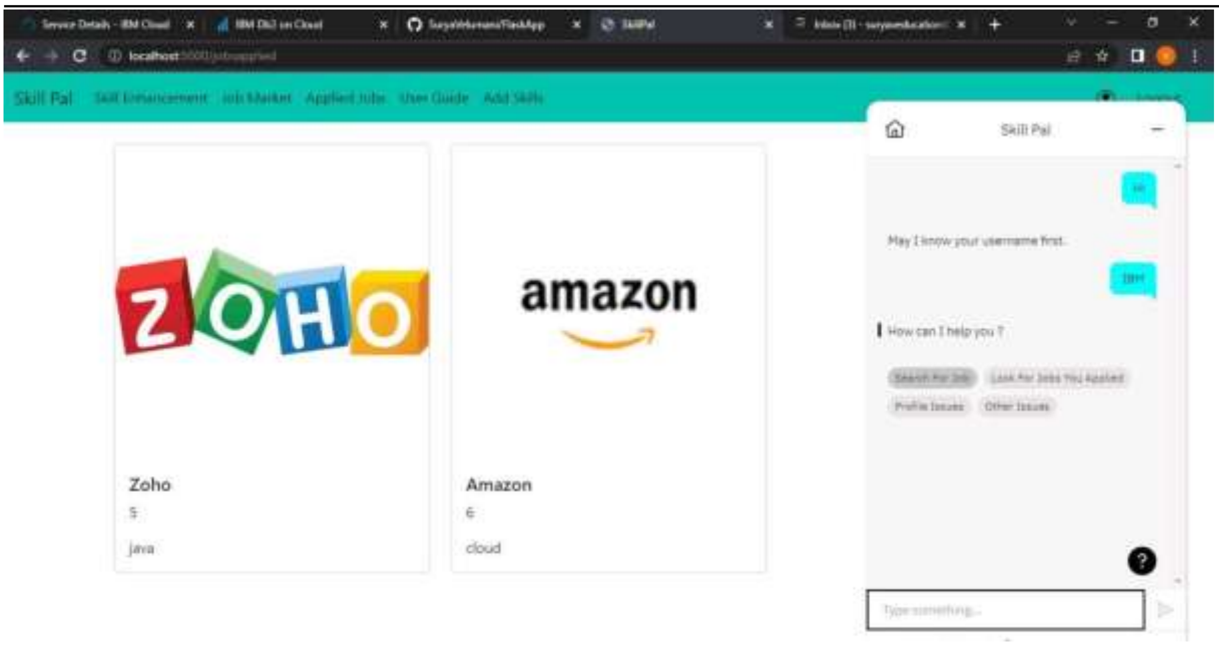
First Name :

Surya

Last Name :
V

Save

Service Details



Zoho
5
java

Amazon
6
cloud

Skill Pal

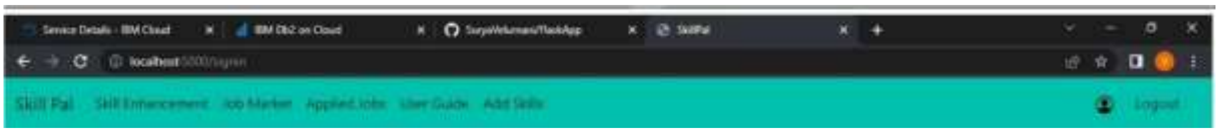
Hi

May I know your username first.

IBM

How can I help you ?

Builtwith IBMWatson* @



Please go through the courses below to enhance your skills

Java Courses

Click

Python Courses

Click

C++ Courses

Click

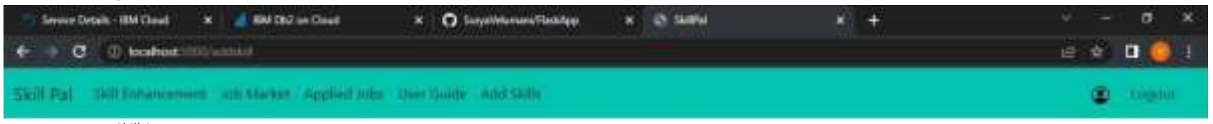
Javascript Courses

Click

Skill 1

Java

Skill 2



Skill 3

Save

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.APPENDlX

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_APl 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 threadhold 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 {O} Complete!\n".format(item_name))
  except ClientError as be: print("CLlENT
    ERROR:    .format(be))
    except Exception as e: print("Unable to complete multi-part
        upload: {O}".format(e))

@app.routeC/uploadResume', methods = ['GET', ' POST']) def
upload():

   if request.method - 'POST':
      bucket='svdemoibmll name_file = session[ '
      username']           name_file         +=
          '.png' filenameis = request.files['file ' ]
      filepath = request.form[ ' filepath ' ] f =
      filepath f = f+filenameis.filename print("-    -
      Il,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[IMAlL

_SERVER']='smtp.sendgrid.net'
app.config[IMAlL_       PORT']  =          465
app.config[IMAlL_ USERNAME'] = ' apikey'
app.config[IMAlL_       USE_TLS']          =          False
app.config[IMAlL_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['SKlLL1'] ski112 =
    skillres['SKlLL2 1 ] ski113 = skillres['SKlLL3 1
    ]    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,l,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 SKILLI - -?,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,l,usernameskill)
        ibm_db.bind_param(stmt,2,ski1112)
        ibm_db.bind_param(stmt,3,ski1122)
        ibm_db.bind_param(stmt,4,ski1132)
        print("::::::::::::::::::::::::::::::::::",sql)
        ibm_db.execute(stmt)     msg          =        "Saved  Successfully     !"        return
        emplate('addSkill.html',msg                                                   render_
@app.route('/jobmarket')                 = msg, ski111=ski1112,ski112=ski1122,ski113=ski1132)
def      jobmarket():

    jobids          =          l]
    jobnames        =          [l
    jobimages       =          []
    jobdescription =[]


                    =
                                 JOBMARKET"
         = username = session[ꞌusername']
    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[l])
    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    ꞌ    ,
    jobinformation = jobinformation)

@app.route('/filterjobs')
def filterjobs():
    skilll = ski112 = ski113 = user = session['username ꞌ] 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['SKlLL1 1
        ]    ski112   = skillres['SKlLL2']
        ski113        = skillres['SKlLL3
             1          ] print(skillres)
        jobids = []   jobnames          =
             [l jobimages       [l
        jobdescription =[]


    sql = "SELECT * FROM
        sql =          "SELECT           *           FROM
        JOBMARKET"          stmt      =
        ibm_db.prepare(conn,        sql) username =
        session[ ꞌusername ꞌ] print(username)
        #ibm_db.bind_param(stmt,l,username
        )    ibm_db.execute(stmt)   joblist   =
        ibm_db.fetch_tuple(stmt) print(joblist)
        while          joblist    !=        False:
```

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




      jobids.append(joblist[0])
      jobnames.append(joblist[l])
      jobimages.append(joblist[2])
      jobdescription.append(joblist[3]) joblist
      = ibm_db.fetch_tuple(stmt)
      jobinformation = [l

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

      for i in range(size): col =
         []
 print("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 @@@@@@@@@@@@@@",jobdescription[i])
                                   if jobdescription[i].lower() == skilll.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 z
         ?"        stmt     =        ibm_db.prepare(conn,   passCheck)
         ibm_db.bind_param(stmt,l,username) ibm_db.execute(stmt) result =
         ibm_db.fetch_assoc(stmt) passWordlnDb = result["UPASSWORD"] if
         passWordlnDb == 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) usernamelnUser = account[ '
    USERNAME'] userPassword = account[ ' UPASSWORD']
             userEmail = account[ ' EMAlLID ' ] firstName = account[ ' FlRSTNAME ' ] lastName = account[ '
LASTNAME ' ] print(account) return render_template('profile.html',



      sql "SELECT * FROM
      usernamelnUser=usernamelnUser,userPassword=userPassword,userEmail=userEmail,firstName
      =firs tNa me, lastName=lastName)

@app.route('/editProfile',methods =['GET','POST']) def
editProfile():  if
request.method
           'POST:
```

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



    msg = username = request.form['usernamelnUser'] password = request.form[ 'userPassword']
    email = request.form[ 'userEmail'] fname = request.form['firstName            '] lname =
    request.form['lastName '] sq - "UPDATE ACCOUNT SET UPASSWORD = EMAILID = FIRSTNAME =
    LASTNAME = ? WHERE
USERNAME = ?: stmt = ibm_db.prepare(conn, sql) ibm_db.bind_param(stmt,l,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 ,
usernamelnUser=username,userPassword=password,userEmail=email,firstName=fname,lastName
=lna me)

@app.route('/logout')
def logout():
   session.pop( 'loggedin ', None) session.pop(
   'username', None)
   returnredirect(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']
        =        ACCOUNT WHERE username =?"
          =
     ibm_db.bind_param(stmt,l,username )
     ibm_db.execute(stmt) account = ibm_db.fetch_assoc(stmt)

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




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


@app.route(l/jobapplied/<int:jobid>l) 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) jobemall print('l-— ,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 z ?" 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["EMAlLID"]

    sql "SELECT * FROM
  msgcontent = request.form[ ˡreasoncontent ˡ]
  emailJob = request.form[ ˡjobEmailForm ˡ]
  portfolioLink = request.form[ ˡportfolio'] city =
  request.form[ ˡcitypreffered'] appliedJobld =
  request.form[ ˡappliedJobld ˡ] print("-
  ------------------------",appliedJobld) insert_sql =
  "INSERT INTO APPLIEDJOBS VALUES (?,?)"prep_stmt
  = ibm_db.prepare(conn, insert_sql) ibm
  db.bind_param(prep_stmt, 1, username)
  ibm_db.bind_param(prep_stmt, 2,
  int(appliedJobld)) ibm_db.execute(prep_stmt)

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


@app.route('/jobsapplied')
def       jobsapplied():
   jobidsl          =          [l
   jobinformation =
  [l

  sql = "SELECT * FROM APPLIEDJOBS WHERE USERNAME = ?"
  stmt = ibm_db.prepare(conn, sql) username = session[ ˡ
  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:
     pri t("----------------------------------------" ,jobl i st)
          jobidsl.append(joblist[l])           joblist   =
    ibm_db.fetch_tuple(stmt)

  print(jobidsl) for x in range(len(jobidsl)):
  jobids
  =       l]         jobnames          =
        []jobimages       =          [l
  jobdescription =[]

    print("nnnnnnnnnnnnnnnnnnnnnnnnnn",len(jobidsl)) sql
    = "SELECT * FROM JOBMARKET WHERE JOBID = ?"
    stmt = ibm_db.prepare(conn, sql) ibm_db.bind_param(stmt,l,jobidsl[x])
    print("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:               ",jobidsl             [x])
    ibm_db.execute(stmt)          joblist          =          ibm_db.fetch_tuple(stmt)
    print(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>",jobl  ist)  while  joblist  !=
    False: jobids.append(joblist[0])
    jobnames.append(joblist[l]) jobimages.append(joblist[2])
    jobdescription.append(joblist[3]) joblist =
    ibm_db.fetch_tuple(stmt) 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])
    print("CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

  CCCCCCCCCCCCCCCcc",col) jobinformation.append(col) print(jobinformation)

  print("/////////////////////////////////////////////",jobinformation) return

  render_template('appliedjobs.html', jobinformation = jobinformation)
```

#OOCIAB

GitHub & Project Demo Link:

https://github.com/lBM-EPBL/lBM-Project-24324-1659941545