SKILL / JOB RECOMMENDER APPLICATION

PNT2022TMID18818

SUBMITTED BY

TEAM MEMBERS:

LEADER : SAMUKTHA M

MEMBERS: MANJU V,
SOUNDHARIYA S,
SRINIVASULU S

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.

1.1 PROJECT OVERVIEW

There has been a sudden boom in the technical industry and an increase in the number of good startups. Keeping track of various appropriate job openings in top industry names has become increasingly troublesome. This leads to deadlines and hence important opportunities being missed. Through this research paper, the aim is to automate this process to eliminate this problem. To achieve this, IBM cloud services like db2, Watson assistant , cluster, kubernetes have been used. A hybrid system of Content-Based Filtering and Collaborative Filtering is implemented to recommend these jobs. The intention is to aggregate and recommend appropriate jobs to job seekers, especially in the engineering domain. The entire process of accessing numerous company websites hoping to find a relevant job opening listed on their career portals is simplified. The proposed recommendation system is tested on an array of test cases with a fully functioning user interface in the form of a web application. It has shown satisfactory results, outperforming the existing systems. It thus testifies to the agenda of quality over quantity

1.2 PURPOSE

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

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

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

2.2 REFERENCES

Singh A, Catherine R, Visweswariah K (2010). PROSPECT: A System for Screening Candidates for Recruitment. In Proceedings of 19th ACM International Conference on Information and Knowledge Management (CIKM'10), Toronto, Ontario, Canada, ACM pp. 659- 668

Rajaraman A, WalmartLabs, Ullman JD (2011). Ch9: Recommendation Systems. In Mining of Massive Datasets California, USA, Stanford University pp. 287-321

Schafer J, Konstan J, Riedl J (1999). Recommender Systems in ECommerce. In Proceedings of the First ACM Conference on Electronic Commerce (EC 1999), Denver, Colorado, USA, ACM pp. 158-166.

Pazzani MJ, Billsus D (2007). Content-Based Recommendation Systems. The Adaptive Web: Methods and Strategies of Web Personalization 4321:325-341.

Yi X, Allan J, Croft W B (2007). Matching Resumes and Jobs Based on Relevance Models. In Proceedings of SIGIR, New York, NY, USA, ACM pp. 809-810.

Zhao Y, Feng X, Li J, Liu B (2011). Shared collaborative filtering. In Proceedings of the fifth ACM conference on Recommender systems (RecSys '11), New York, NY, USA, ACM pp. 29-36.

2.3 PROBLEM STATEMENT DEFINITION

Many new graduates are confused about what roles they fit in, skill and job recommender system helps to find specific jobs from a set of job postings to users based on their skill set. It is very easy and simple to find current job openings based on the eligibility criteria. The users and their information is stored in the database . An alert is sent when there is an opening based on the user skill set. By using cloud we can easily manipulate and retrieve the data. It is one of the easiest way for the recruiters to find the employee and their skills.

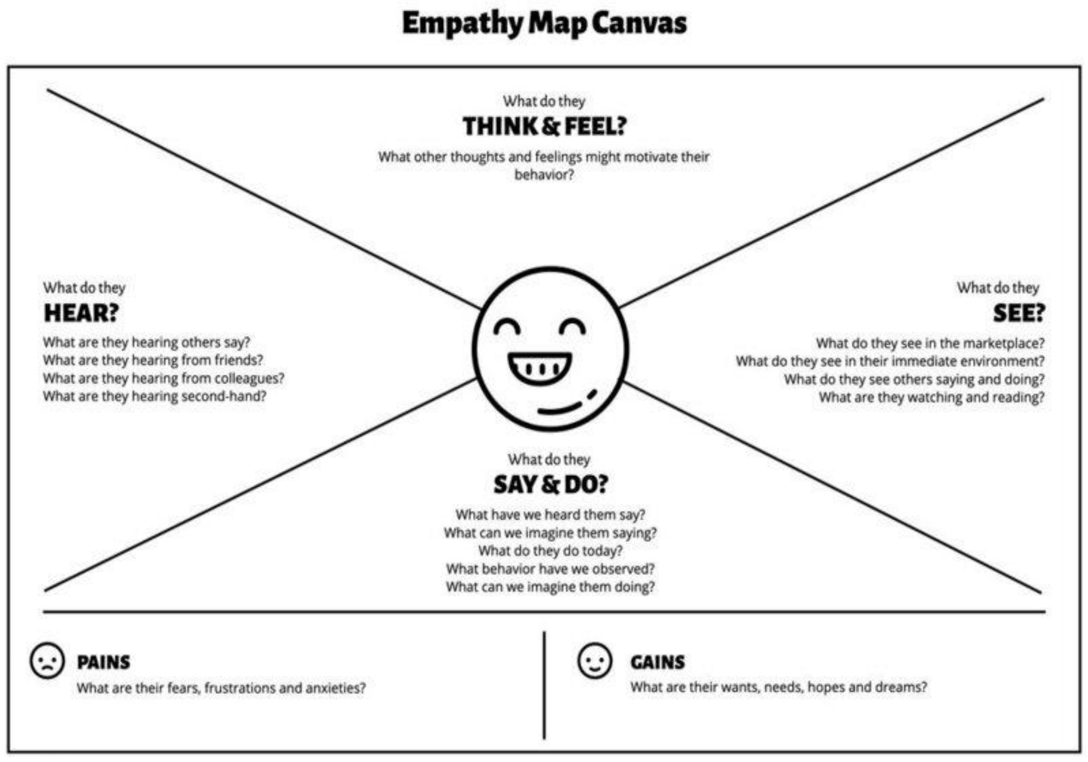
3. IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

An empathy map canvas serves as a foundation for outstanding user experiences, which focus on providing the experience customers want rather than forcing design teams to rely on guesswork.

To help understand the various reasons why a user might interact with the product so they can design a user-friendly experience.

Agile teams in a variety of departments use empathy map canvases to better understand how to meet their customers’ needs.

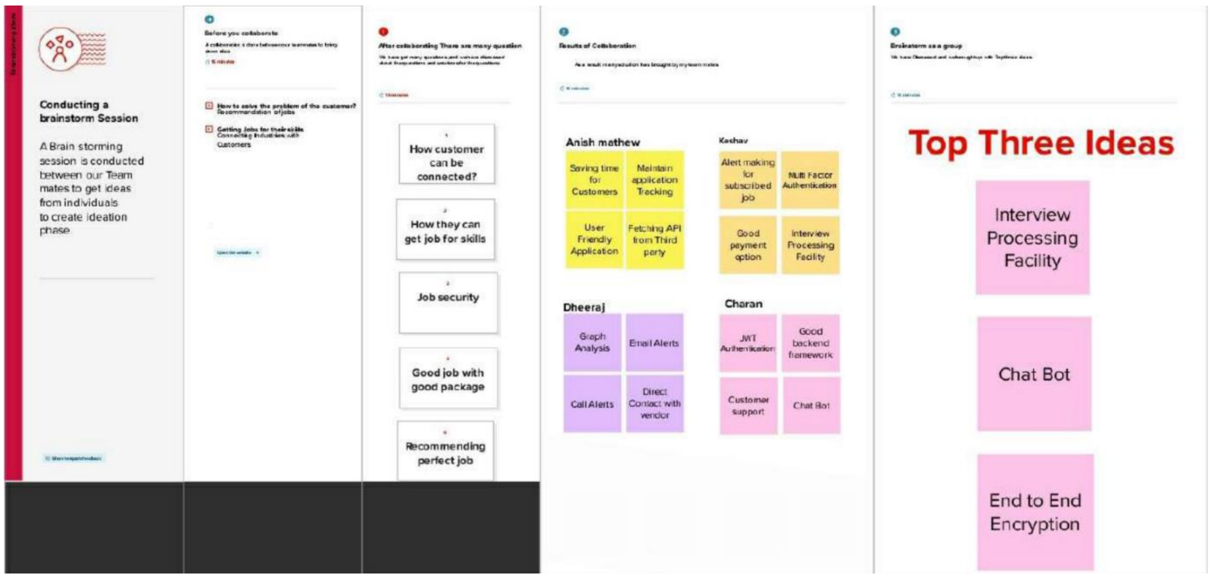

Empathy Map Canvas
What do they
THINK & FEEL?
What other thoughts and feelings might motivate their behavior?
What do they
HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
What do they
SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
What do they
SAY & DO?
What have we heard them say?
What can we imagine them saying?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
PAINS
What are their fears, frustrations and anxieties?
GAINS
What are their wants, needs, hopes and dreams?

3.2 IDEATION AND BRAINSTROMING

Ideation refers to the whole creative process of coming up with and communicating new ideas. It can take many different forms, from coming up with a totally new idea to combining multiple existing ideas to create a new process or organizational system. Ideation is similar to a practice known as brainstorming.

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.


Conducting a brainstorm Session
A Brain storming session is conducted between our Team mates to get ideas from individuals to create ideation phase
How to solve the problem of the customer? Recommendation of jobs
Getting Jobs for their skills
Connecting Industries with Customers
How customer can be connected?
How they can get job for skills
Job security
Good job with good package
Recommending perfect job
Anish mathew
Saving time for Customers
Maintain application Tracking
User Friendly Application
Fetching API from Third party
Keshav
Alert making for subscribed job
Multi Factor Authentication
Good payment option
Interview Processing Facility
Dheeraj
Graph Analysis
Email Alerts
Call Alerts
Direct Contact with vendor
Charan
JWT Authentication
Good backend framework
Customer support
Chat Bot
Top Three Ideas
Interview Processing Facility
Chat Bot
End to End Encryption

3.3 PROPOSED SOLUTION

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

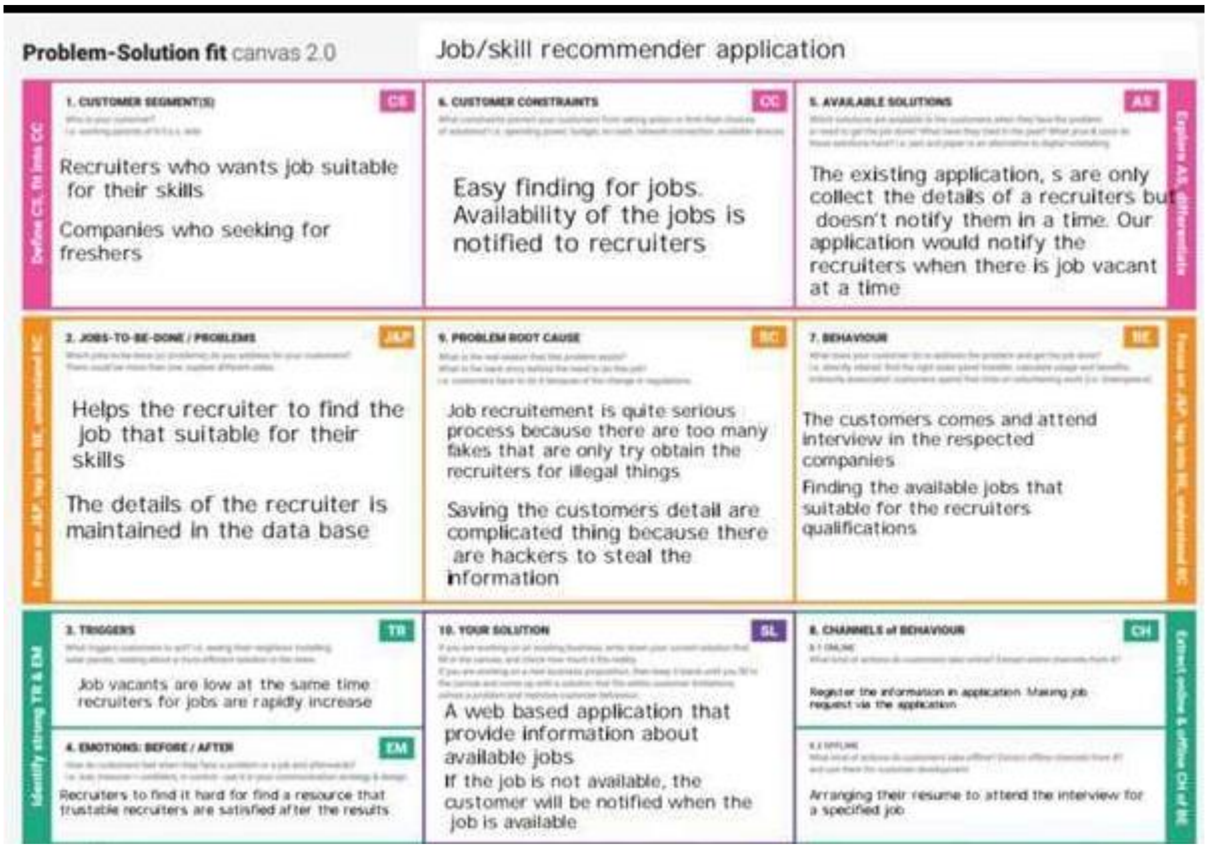
Convert unstructured data to structured data for processing

Providing weightage to designated elements (skills, experience, SME

Model based recommendation

3.4 PROBLEM SOLUTION FIT

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem.



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

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

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

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.

4.2 NON FUNCTIONAL REQUIREMENTS

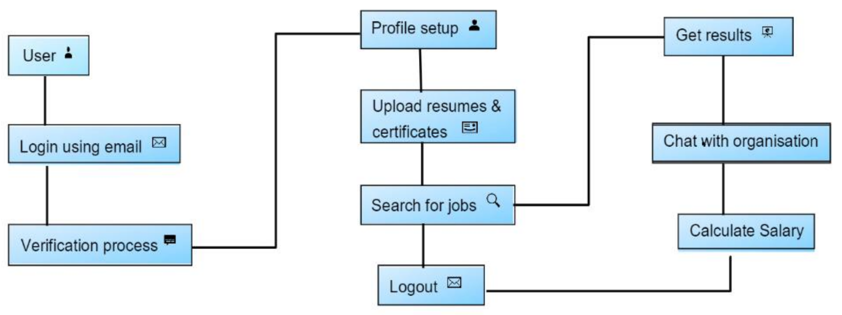
Non functional Requirements are :

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

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.
Each process should have at least one input and an output. Each data store should have at least one data flow in and one data flow out.



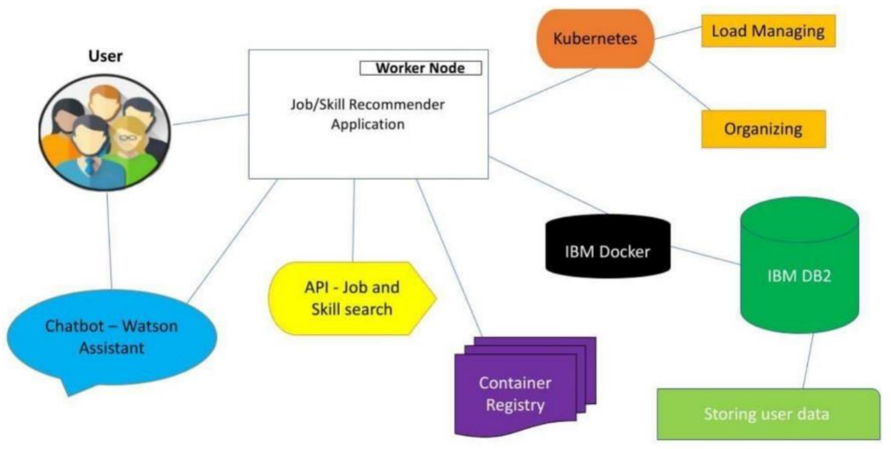
5.2 SOLUTION AND TECHNICAL ARCHITECTURE

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

5.3 USER STORIES

User can easily find the current job openings and they can apply for it. The Skill and Job Recommender System helps the user to find the job openings according to their relevant skills. So, this also increases the opportunity for the user to find their applicable jobs.

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Acceptance criteria	Team Members
Sprint-1	UI / UX Design	USN-1	As a user, I need to interact with websites. So, this will be the interface between me and website.	High	I can interact with website	Samuktha M Manju V Soundhariya S Srinivasulu Yadav
Sprint-1	Registration	USN-2	As a user, I can register for the application by entering my email, password, and confirming my password.	High	I can register with email	Samuktha M Manju V Soundhariya S Srinivasulu Yadav
Sprint-1		USN-3	As a user, I will receive confirmation email once I have registered for the application	High	I can receive confirmation mail	Samuktha M Manju V Soundhariya S Srinivasulu Yadav

Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	Low	I can login to the application by entering login credentials	Samuktha M Manju V Soundhariya S Srinivasulu Yadav

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

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.

6.2 SPRINT DELIVERY SCHEDULE

SPRINT	TASK	MEMBERS
SPRINT 1	Create Registration page login page , Job search portal , job apply portal in flask	Samuktha M Manju V Soundhariya S Srinivasulu S
SPRINT 2	Connect application to IBM db2	Samuktha M Manju V Soundhariya S Srinivasulu S
SPRINT 3	Integrate IBM Watson assistant	Samuktha M Manju V Soundhariya S Srinivasulu S
SPRINT 4	Containerize the app and Deploy the application in IBM cloud	Samuktha M Manju V Soundhariya S Srinivasulu S

6.3 REPORTS FROM JIRA

Resolution Time Report.
Single Level Group By Report.
Time Since Issues Report.
Time Tracking Report.
Average Age Report.
Created vs Resolved Issues Report.
Pie Chart Report.
Recently Created Issues Report.

7. CODING AND SOLUTIONING

7.1 FEATURE-1

```
# using SendGrid's Python Library
 # https://github.com/sendgrid/sendgrid-python
Import os
From sendgrid import SendGridAPIClient
From sendgrid.helpers.mail import Mail
# from_address we pass to our Mail object, edit with your name
FROM_EMAIL = ‘Your_Name@SendGridTest.com’
 Def SendEmail(to_email):
“”” Send an email to the provided email addresses
:param to_email = email to be sent to
:returns API response code
:raises Exception e: raises an exception “””
Message = Mail(
From_email=FROM_EMAIL,
To_emails=to_email,
 Subject=’A Test from SendGrid!’,
 Html_content=’Hello there from SendGrid your URL is: ‘ + ‘ ‘right here!’)
Try:
 Sg = SendGridAPIClient(os.environ.get(‘SENDGRID_API_KEY’))
Response = sg.send(message)
Code, body, headers = response.status_code, response.body, response.headers
Print(f’Response Code: {code} “)
Print(f’Response Body: {body} “)
Print(f’Response Headers: {headers} “)
Print(“Message Sent!”)
 Except Exception as e:
Print(“Error: {0}”.format€)
 Return str(response.status_code)
If __name__ == “__main__”:
 SendEmail(to_email=input(“Email address to send to? “))
```

7.2 FEATURE-2

```
<script> window.watsonAssistantChatOptions = { integrationlD: "9be41b76-
06bO-426f-8469-962f2963cdb6",
// 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( l script l );
t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion I I 'latest) +
"/WatsonAssistantChatEntry.js"; document.head.appendChild(t); </script>
```

App.py

```
import ibm_db
from flask import Flask, flash, redirect, render_template, request, url_for
from flask_mail import
Mail, Message from sendgrid import SendGridAPIClient
 from sendgrid.helpers.mail
import Mail from sendmailer import *
 import pandas as pd
 app = Flask(__name__)
# app.secret_key="
# configure the mail settings
 SENDGRID_API_KEY =
"SG.bnGBaY6cSGeU106QGq_H5Q.YhqfT29UYDRV9yWp3Rfn73LQykmE4
55 Zckt_qyJSR2U"
 app.config['SECRET_KEY'] = 'top-secret!'
 app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
 app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
 # app.config['MAIL_PASSWORD'] =
os.environ.get('SENDGRID_API_KEY')
 # app.config['MAIL_DEFAULT_SENDER'] =
 os.environ.get('vasanthias52@gmail.com') mail = Mail(app) conn =
ibm_db.connect("DATABASE=bludb;
HOSTNAME=b0aebb68-94fa46ec-
a1fc1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;
PORT=31249;
SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=pfd0
3782;
PWD=dnhtwrcfkZlhkhAO",",") # type: ignore print(conn)
 print("connection successful...")

return render_template('sendgrid.html')
@app.route('/loginpage', methods=['POST','GET'])
def loginpage(): if request.method == 'POST':
email = request.form['email'] password =
request.form['password'] if not email or not
password:
return render_template('login.html',error='Please fill all fields')
```

```
query = "SELECT * FROM USERS WHERE EMAIL=? AND
PASSWORD=?"
stmt = ibm_db.prepare(conn, query) # type:ignore
ibm_db.bind_param(stmt,1,email) # type:ignore
ibm_db.bind_param(stmt,2,password) # type:ignore
ibm_db.execute(stmt) # type:ignore isUser =
ibm_db.fetch_assoc(stmt) # type:ignore
print(isUser,password)
if not isUser:
return render_template('login.html',error='Invalid Credentials')
return redirect(url_for('home'))
return render_template('login.html',name='Home')
@app.route('/signup') def registration():
return render_template('signup.html')
@app.route('/signup', methods=['POST','GET']) def
signup():
if request.method == 'POST':
name = request.form['name']
email = request.form['email']
phone = request.form['phone']
password = request.form['password']
sql ="INSERT INTO USERS VALUES (?,?,?,?)"
stmt = ibm_db.prepare(conn,sql) # type: ignore
ibm_db.bind_param(stmt, 1, name) # type: ignore
ibm_db.bind_param(stmt, 2, email) # type: ignore
ibm_db.bind_param(stmt, 3, phone) # type: ignore
ibm_db.bind_param(stmt, 4, password) # type: ignore
ibm_db.execute(stmt) # type: ignore
sendemail(email,'') return redirect(url_for('home'))
return render_template('signup.html')
@app.route('/') @app.route('/login')
def login(): return
render_template('login.html')
@app.route('/stats') def stats():
return render_template('stats.html')
@app.route('/contacts') def
requester():
return render_template('contacts.html')
@app.route('/tech', methods=['POST']) def
by_tech():
jobs = pd.read_csv('jobs.csv')
input = request.get_json()
tech_name = input['techName']
```

```
page = int(input['page']) if input['page'] else 0
sorting = input['sorting'] if input['sorting'] else 0
filtered_jobs = jobs.loc[jobs['Tech Stack'].str.contains(tech_name, na=False)]
if sorting:
```

```
filtered_jobs = filtered_jobs.sort_values(by=[sorting])
return filtered_jobs[page*10:page*10+10].drop(['Unnamed: 0'],
axis=1).to_json(orient='records')
@app.route('/location', methods=['POST']) def
by_location(): jobs = pd.read_csv('jobs.csv')
input = request.get_json() location =
input['location'] page = int(input['page']) if
input['page'] else 0 sorting = input['sorting'] if
input['sorting'] else 0
filtered_jobs = jobs.loc[jobs['Location'] == location]
if sorting:
filtered_jobs = filtered_jobs.sort_values(by=[sorting])
return filtered_jobs[page*10:page*10+10].drop(['Unnamed: 0'],
axis=1).to_json(orient='records')
@app.route('/forgot') def
reques():
return render_template('forgotten-password.html')
@app.route('/forgot',methods=['POST','GET'])
def forgot(): if request.method == 'POST':
email = request.form['email']
query = "SELECT * FROM USERS WHERE EMAIL=?"
stmt = ibm_db.prepare(conn, query) # type:ignore
ibm_db.bind_param(stmt,1,email) ibm_db.execute(stmt)
# type:ignore
isUser = ibm_db.fetch_assoc(stmt) # type:ignore
# print(isUser,password)
print(isUser)
print(stmt)
sendemail(email,'We have recieved your email! from your email address to
reset the password, we will send you a link to reset your password') return
render_template('login.html')
return render_template('forgotten-password.html')
```

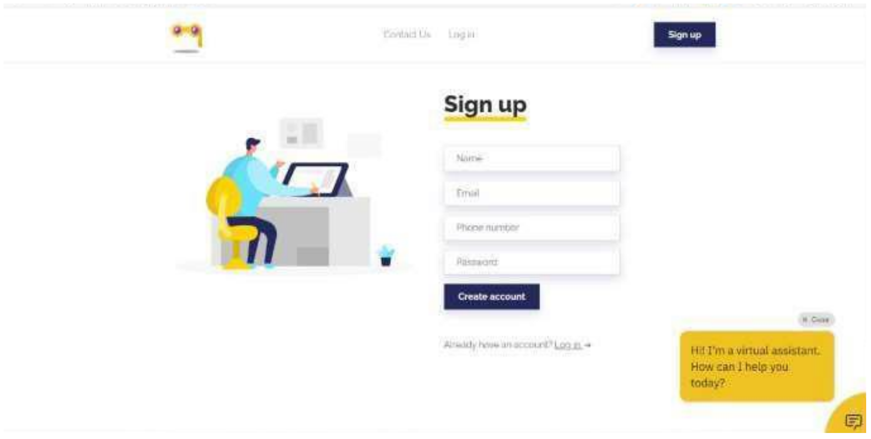
```
@app.route('/features') def
features():
return render_template('features.html')
@app.route('/home') def home():
return render_template('index.html')
```

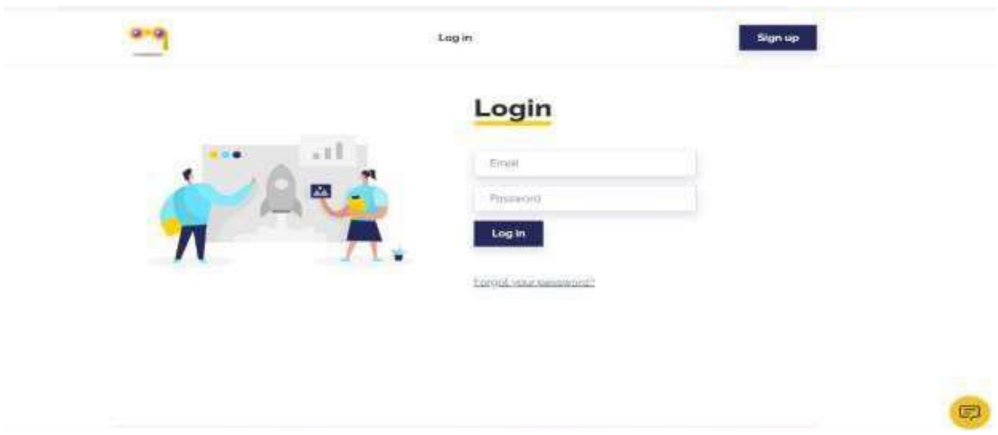
```
# @app.route('/') #
def home():
# return render_template('index.html')
@app.route('/contacts' ,methods=['POST'])
def contacts(): email =
request.form['email']
sendemail(email,'We have recieved your email!')
return render_template('contacts.html')
@app.route('/logout') def
logout():
session.pop('email', None) # type:ignore
return redirect(url_for('login'))
if __name__=='__main__':
app.run(debug=True)
```

OUTPUT:

```
connection successful...
* Serving Flask app 'app' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production
deployment.
Use a production WSGI server instead.
45
* Debug mode: on
WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
```

OUTPUT IMAGES:


Contact Us
Log in
Sign up
Sign up
Name
Email
Phone number
Password
Create account
Already have an account? Log in →
× Close
Hi! I'm a virtual assistant. How can I help you today?


Log in

Sign up

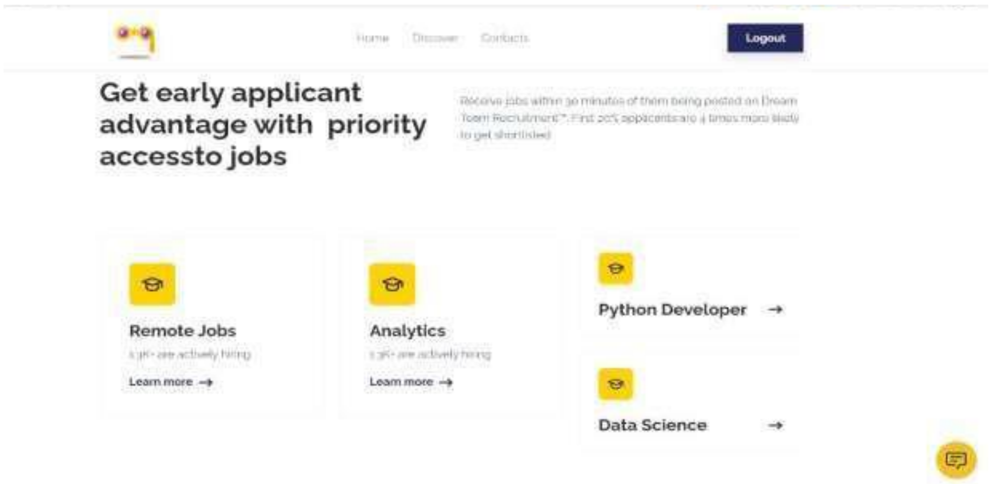
Login

Email

Password

Log in

Forgot your password?


Home Discover Contacts

Logout

Get early applicant advantage with priority accessto jobs

Receive jobs within 30 minutes of them being posted on Dream Team Recruitment™. First 20% applicants are 4 times more likely to get shortlisted

Remote Jobs

1.3K+ are actively hiring

Learn more →

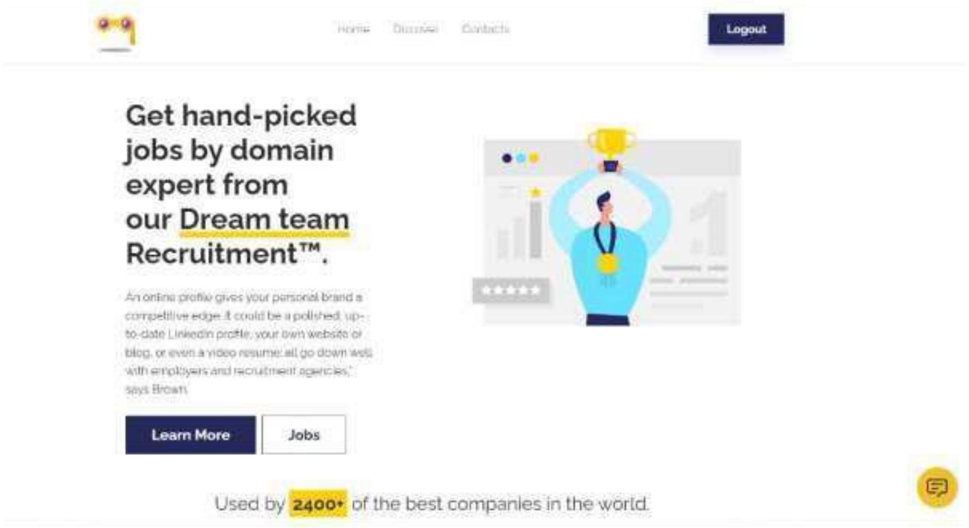
Analytics

1.3K+ are actively hiring

Learn more →

Python Developer →

Data Science →


Home Discover Contacts

Logout

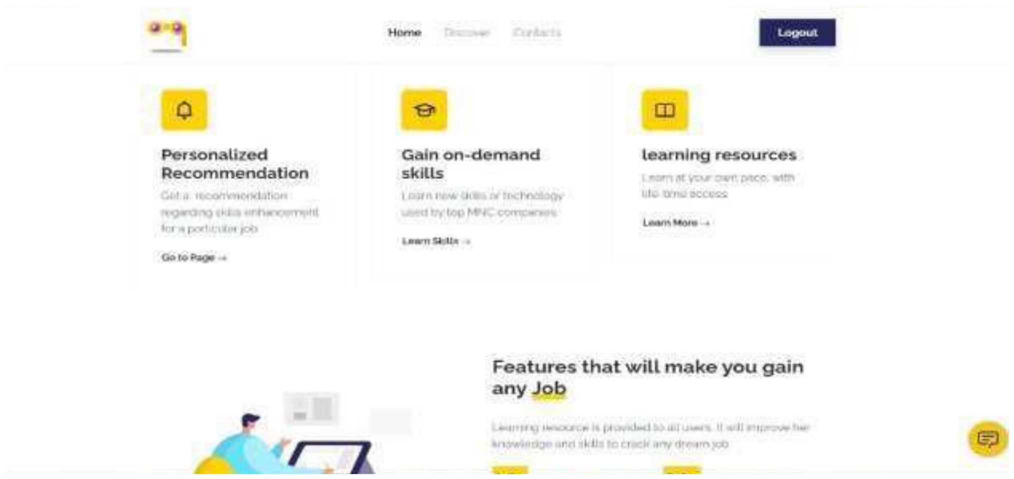
Get hand-picked jobs by domain expert from our Dream team Recruitment™.

An online profile gives your personal brand a competitive edge. It could be a polished, up-to-date LinkedIn profile, your own website or blog, or even a video resume: all go down well with employers and recruitment agencies," says Brown.

Learn More

Jobs

Used by 2400+ of the best companies in the world.



Personalized Recommendation

Get a recommendation regarding skills enhancement for a particular job

Gain on-demand skills

Learn new skills or technology used by top MNC companies

learning resources

Learn at your own pace, with life time access

Features that will make you gain any Job

Learning resource is provided to all users. It will improve her knowledge and skills to crack any dream job

8. TESTING

Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance.

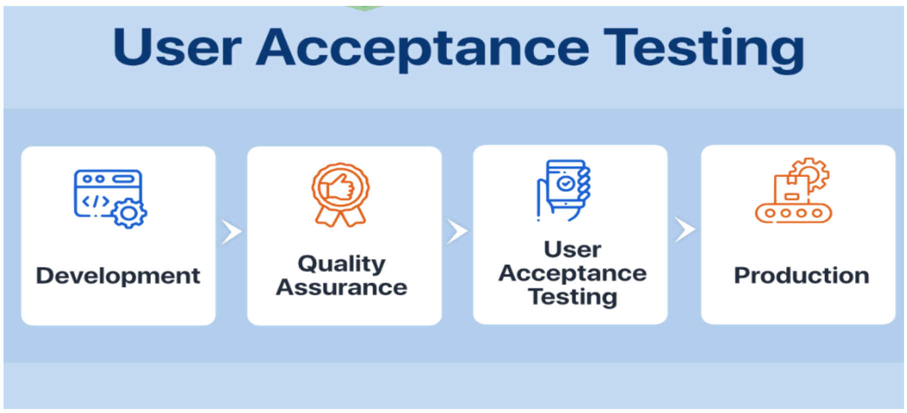
8.1 TEST CASES

Testcase1: Does the flask application is perfectly created and in works in very good condition?
Testcase2: Does the Send-Grid integration is working correctly?
Testcase3: Does the db2 is perfectly connected to the application?
Testcase4: Can the chat-bot which is created using Watson assistant is recommending correctly the job for the end users?
Testcase 5: Whether the application is working correctly without any interruptions?

8.2 USER ACCEPTANCE TESTING

User Acceptance Testing (UAT) is a process to check whether the system accepts a user's requirements. It's performed at a time when actual users use the system. This testing comes after - **Unit Test--->Integration Test, --->System Test, --->Acceptance Test** in the process. Its testing process related to another analogy, such as manufacturing pens. While production of a ballpoint pen, the cap, the body, the tail, the clip, the ink cartridge, with the help of things mentioned above, a full ballpoint pen manufactured after that single pen was produced with a combination of every single item.

Each component specified above was tested to ensure that each component will make the pen in a working condition. When a complete pen is integrated, System Test is performed. Once it is over, perform the Acceptance test to confirm that each ball pen is in working condition and ready for customers.



9. RESULTS

9.1 PERFORMANCE METRICS

a) Implementation of web application:
To create the web application to interact with the users. The users here is commonly job seeker and job provider. Login, Signup, Job searching have separate pages where we can access into different work functions.
b) SendGrid Integration:
The flask application that we created is to get integrated with sendgrid which provide the e-mail interface for communication purpose.
c) Developing chatbot:
To develop a chat-bot so that, that can be very interactive to the users who are using the application and to recommend the jobs based on the job seekers interests.
d) Deployment of Application:
Finally the developed application is to deployed in the cloud.
1 .Accuracy
The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions.
To implement an accuracy metric, we can compare ground truth and predicted value in a loop.

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

It helps the candidate to search for the job according to their skills and it creates the awareness to the candidate for all 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.

DISADVANTAGES:

It is costly.

Uninterrupted internet connection is required for smooth functioning of application.

11. CONCLUSION

The Skill and Job Recommender System helps to search and find the jobs according to the candidate skill sets.
We have used IBM cloud services like db2, cloud registry , kubernetes , Watson assistant to create this application , which will be very useful for the candidates who are searching for jobs.

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:

```
import { useToast } from "@chakra-ui/react";
import React, { useContext, useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import { AppContext } from "../context/AppContext";
import { loginUser } from "../proxies/backend_api";
import { emailRegex } from "../utils/helper";

const Login = () => {
  const toast = useToast();
  const { setUser } = useContext(AppContext);

  const navigate = useNavigate();

  const [inputs, setInputs] = useState({
    email: "",
    password: "",
  });

  const [error, setErrors] = useState({
    email: "",
    password: "",
  });

  const handleChange = ({ target: { name, value } }) => {
    setErrors((prev) => {
      return { ...prev, [name]: "" };
    });
    setInputs((prev) => ({ ...prev, [name]: value }));
  };

  const checkInputErrors = () => {
    let status = true;
    if (inputs.email.trim() === "" || !emailRegex.test(inputs.email.trim())) {
      setErrors((prev) => {
        return { ...prev, email: "Enter a valid email" };
      });
      status = false;
    }

    if (inputs.password.trim() === "") {
      setErrors((prev) => {
        return { ...prev, password: "Enter a valid password" };
      });
      status = false;
    }

    if (inputs.password.trim().length < 6) {
      setErrors((prev) => {
        return { ...prev, password: "Minimum 6 characters" };
      });
      status = false;
    }
    return status;
  };

  const handleLogin = async () => {
    if (checkInputErrors()) {
      const data = await loginUser(inputs);
      if (data.error) {
        toast({
          title: data.error,
          status: "error",
          duration: 3000,
          isClosable: true,
          variant: "left-accent",
          position: "top",
        });
        return;
      }
      setUser(data);
      toast({
        title: `Welcome back ${data.name}`,
        status: "success",
        duration: 3000,
        isClosable: true,
        variant: "left-accent",
        position: "top",
      });
      localStorage.setItem("user", JSON.stringify(data));
      navigate("/dashboard");
    }
  };

  return (
    <>
      <div>
        <button className="bg-base-300 rounded-box flex flex-row justify-
evenly items-center gap-10 px-10 py-5 w-fit mx-auto">
          <span>Sign in with Github</span>
          <img src={`github-dark.png`} alt="github" width="14%" />
        </button>
        <div className="divider max-w-xs">or</div>
        <form
          onSubmit={(e) => e.preventDefault()}
          className="card bg-base-300 rounded-box flex flex-col justify-center
items-center gap-5 px-10 py-5 w-fit mx-auto"
        >
          <div>
            <input
              value={inputs.email}
              type="text"
              name="email"
              placeholder="email"
              className="input input-bordered input-primary w-full"
              onChange={handleChange}
            />
            {error.email !== "" && (
              <p className="text-sm text-red-500 mt-1 font-medium">
                {error.email}
              </p>
            )}
          </div>
          <div>
            <input
              value={inputs.password}
              type="password"
              name="password"
              placeholder="password"
              className="input input-bordered input-primary w-full"
              onChange={handleChange}
            />
            {error.password !== "" && (
              <p className="text-sm text-red-500 mt-1 font-medium">
                {error.password}
              </p>
            )}
          </div>
          <div className="text-center">
            <button
              type="submit"
              onClick={handleLogin}
              className="btn btn-sm btn-primary mb-4"
            >
              Login
            </button>
          </div>
        </form>
      </div>
    </>
  );
};
export default Login;
```

```
import React, { useContext, useEffect, useState } from "react";
import { useNavigate } from "react-router-dom";
import { AppContext } from "../context/AppContext";
import { registerUser } from "../proxies/backend_api";
import { emailRegex } from "../utils/helper";

const SignUp = () => {
  const { setUser } = useContext(AppContext);

  const navigate = useNavigate();

  const [inputs, setInputs] = useState({
    name: "",
    email: "",
    phone_number: "",
    password: "",
    confirm_password: "",
  });

  const [error, setErrors] = useState({
    name: "",
    email: "",
    phone_number: "",
    password: "",
    confirm_password: "",
  });

  const handleChange = ({ target: { name, value } }) => {
```

```
  setErrors((prev) => {
    return { ...prev, [name]: "" };
  });
  setInputs((prev) => ({ ...prev, [name]: value }));
};

const checkInputErrors = () => {
  let status = true;
  if (inputs.email.trim() === "" || !emailRegex.test(inputs.email.trim())) {
    setErrors((prev) => {
      return { ...prev, email: "Enter a valid email" };
    });
    status = false;
  }

  if (inputs.name.trim() === "") {
    setErrors((prev) => {
      return { ...prev, name: "Enter a valid name" };
    });
    status = false;
  }

  if (inputs.phone_number.trim() === "") {
    setErrors((prev) => {
      return { ...prev, phone_number: "Enter a valid phone number" };
    });
    status = false;
  }
```

```
if (inputs.confirm_password.trim() === "") {
  setErrors((prev) => {
    return { ...prev, confirm_password: "Enter a valid  password" };
  });
  status = false;
}

if (inputs.password.trim() === "") {
  setErrors((prev) => {
    return { ...prev, password: "Enter a valid password" };
  });
  status = false;
}

if (inputs.password.trim().length < 6) {
  setErrors((prev) => {
    return { ...prev, password: "Minimum 6 characters" };
  });
  status = false;
}

if (inputs.password.trim() !== inputs.confirm_password.trim()) {
  setErrors((prev) => {
    return { ...prev, confirmPassword: "Password don't match" };
  });
  status = false;
}
```

```
  return status;
};

const handleSignUp = async () => {
  if (checkInputErrors()) {
    const data = await registerUser(inputs);
    if (data.error) {
      toast({
        title: data.error,
        status: "error",
        duration: 3000,
        isClosable: true,
        variant: "left-accent",
        position: "top",
      });
      return;
    }
    setUser(data);
    toast({
      title: `Your journey starts here ${data.name}`,
      status: "success",
      duration: 3000,
      isClosable: true,
      variant: "left-accent",
      position: "top",
    });
    localStorage.setItem("user", JSON.stringify(data));
    navigate("/profile");
    }
  };

  return (
    <>
      <div>
        <button className="bg-base-300 rounded-box flex flex-row justify-
evenly items-center gap-10 px-10 py-5 w-fit mx-auto">
          <span>Sign in with Github</span>
          <img src={`github-dark.png`} alt="github" width="14%" />
        </button>
        <div className="divider max-w-xs">or</div>
        <div className="card bg-base-300 rounded-box flex flex-col justify-
center items-center gap-3 px-10 py-5 w-fit mx-auto">
          <div>
            <input
              value={inputs.name}
              type="text"
              name="name"
              placeholder="name"
              className="input input-bordered input-primary w-full"
              onChange={handleChange}
            />
            {error.name !== "" && (
              <p className="text-sm text-red-500  font-medium">{error.name}</p>
            )}
          </div>
          <div>
```

```
            <input
              value={inputs.email}
              type="text"
              name="email"
              placeholder="email"
              className="input input-bordered input-primary w-full"
              onChange={handleChange}
            />
            {error.email !== "" && (
              <p className="text-sm text-red-500  font-
medium">{error.email}</p>
            )}
          </div>
          <div>
            <input
              value={inputs.phone_number}
              type="text"
              name="phone_number"
              placeholder="phone number"
              className="input input-bordered input-primary w-full"
              onChange={handleChange}
            />
            {error.phone_number !== "" && (
              <p className="text-sm text-red-500  font-medium">
                {error.phone_number}
              </p>
            )}
          </div>
```

```
<div>
  <input
    value={inputs.password}
    type="password"
    name="password"
    placeholder="password"
    className="input input-bordered input-primary w-full"
    onChange={handleChange}
  />
  {error.password !== "" && (
    <p className="text-sm text-red-500  font-medium">
      {error.password}
    </p>
  )}
</div>
<div>
  <input
    value={inputs.confirm_password}
    type="password"
    name="confirm_password"
    placeholder="confirm password"
    className="input input-bordered input-primary w-full"
    onChange={handleChange}
  />
  {error.confirm_password !== "" && (
    <p className="text-sm text-red-500  font-medium">
      {error.confirm_password}
    </p>
```

```
          )}
         </div>
         <div className="text-center">
           <button
             onClick={handleSignUp}
             className="btn btn-sm btn-primary mb-4"
           >
             Sign Up
           </button>
         </div>
       </div>
     </div>
   </>
  );
};

export default SignUp;
```