1. INTRODUCTION

1.1. Project Overview

When the whole world is coming back on its feet, those businesses affected by this pandemic disease slowly tries to gain back the momentum it lost. Now is the time when the companies or businesses seek to invest in human resources, which would help them to gain the momentum it lost during this period. To serve the constant cycle of the hiring process in the job applicant's perspective, many job companies have come up with solutions for providing the job board. Here a seeker looks up for the job he would find relevant to him and apply for it.

Aim is to come up with a job recommender system, which takes the skills from LinkedIn and jobs from Indeed and throws the best jobs available for you according to your skills.

1.2. Purpose

As there are many job boards, applicants tend to use the tool that provides better services to them, services such as writing a CV, creating a job profile, and recommending new jobs to a job seeker. Job applicants have become more persistent and proactive in searching for new opportunities that fit their skills. However, companies that are targeting these job seekers are finding it challenging to identify the job seeker's skill and provide personalized job recommendations.

2. LITERATURE SURVEY

Here we will take a look at all the previous solutions, attempts and implementations to the skill/job recommender application or anything that is vaguely related to it.

Job applicants have become more persistent and proactive in searching for new opportunities that fit their skills. However, companies that are targeting these job seekers are finding it challenging to identify the job seeker's skill and provide them job.

2.1. Existing Problem

Hirist.com and Linkedin are some popular websites that allow job seekers to search or recommend jobs based on their needs.

These websites get details from the users and based on the users skills and experience, they recommend jobs to users.They also allow users to post job posting and hire people as their need.

TITLE	PROPOSED WORK	TOOLS USED / ALGORITHMS	TECHNOLOGY	ADVANTAGES / DISADVANTAGES
Job Recommendation based on Extracted Skill Embeddings	Developing job recommendation systems can significantly help both employers and job seekers in speeding up this process and finding the best jobs for them.	collaborative filtering, content based filtering, Hybrid Job Recommender System	Python, Flask, Machine Learning, Html, Bootstrap, Data Science	Best Job Matches can be found for the Job Seekers, but it is not always easy for SaaS providers to know what customers are experiencing.

2.2. References

- https://www.researchgate.net/publication/325697854_Job_Recommendation_based_on_Job_Seeker_Skills_An_Empirical_Study
- https://www.researchgate.net/publication/272802616_A_survey_of_job_recommender_systems

2.3. Problem Statement Definition

Mrs. Anjali has just completed her college and now she is looking for a job that matches her skills. But She couldn't find the job according to her skills and interest.

- Mrs. Anjali wants to know best recommendation of jobs based on her interest and skills.
- She wasted so much time in looking for the Jobs, still couldn't find one according to her need.
- This situation is usually faced by many graduates nowadays
- This causes frustation and depression among the students.

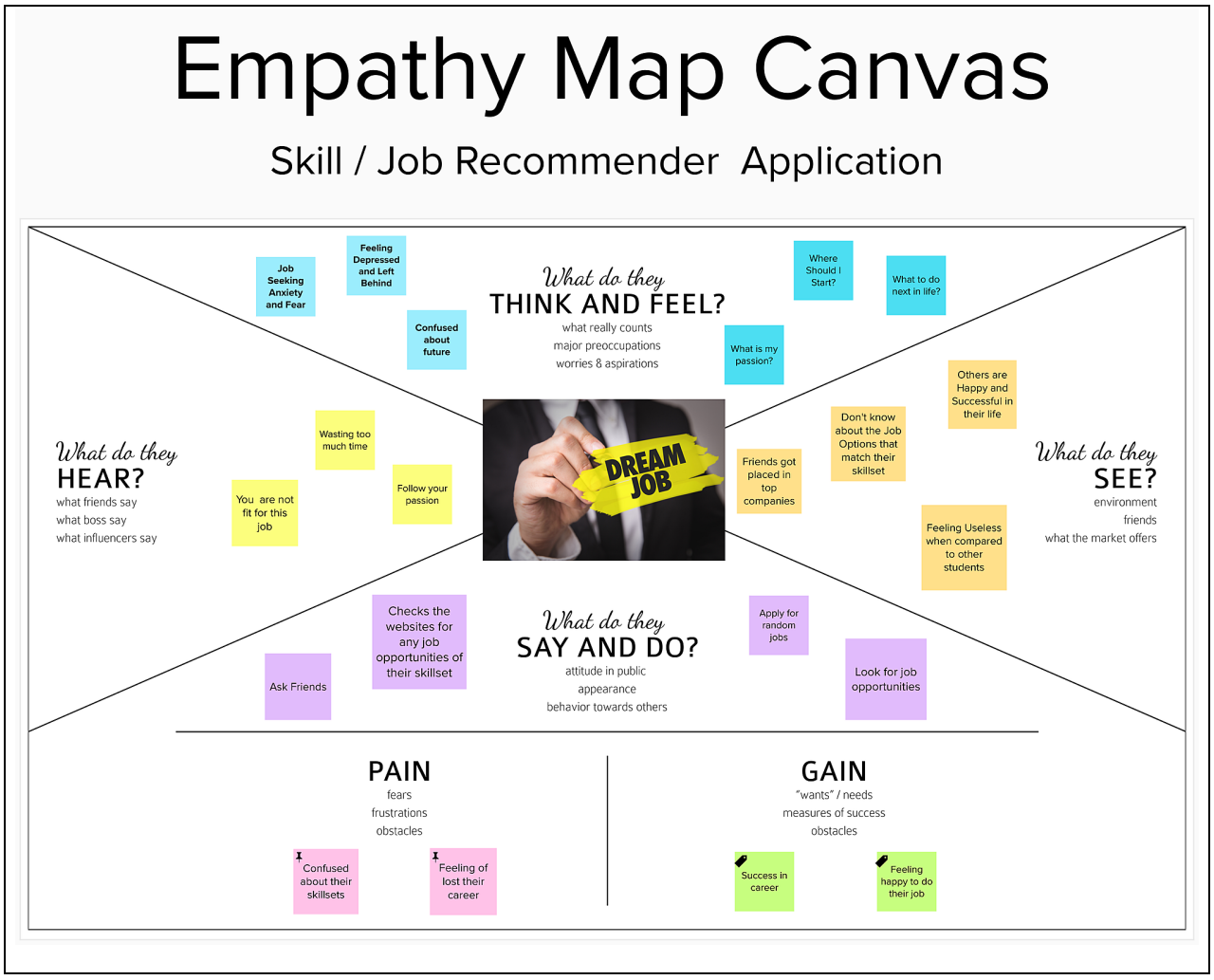
Who does the problem affect?	Graduated Students and Unemployed People
What are the boundaries of the problem?	People who are seeking for jobs based on their passion and interest
What is the issue?	People get frustated and depressed if they don't get a job of their interest and skillsset after graduation.
When does the issue occur?	After completion of studies and confused about career.

Where does the issue occur?	These issue occurs in rural areas where the people don't even have the knowledge of vast amount of job categories available in the market.
Why is it important that we fix the problem?	It is important so that our younger genration can be guided in the correct path and they can grow constantly in there field of interest, which will also be helpful for our nations growth.
What solution to solve this issue?	A recommendation algorithm based app that can recommend the available job postings the matches the user interests.
What methodology used to solve the issue?	Machine Learning based filtering algorithms can be used to filter and recommend jobs.

3. IDEATION & PROPOSED SOLUTION

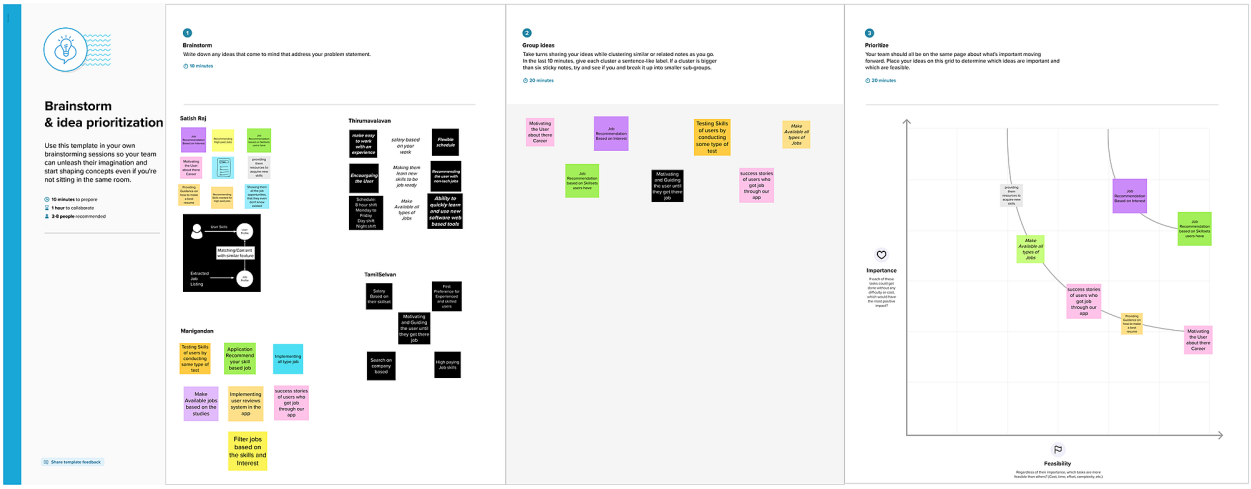
3.1. Empathy Map Canvas

An empathy map canvas helps brands provide a better experience for users by helping teams understand the perspectives and mindset of their customers. Using a template to create an empathy map canvas reduces the preparation time and standardizes the process so you create empathy map canvases of similar quality.



3.2. Ideation & Brainstorming

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity.



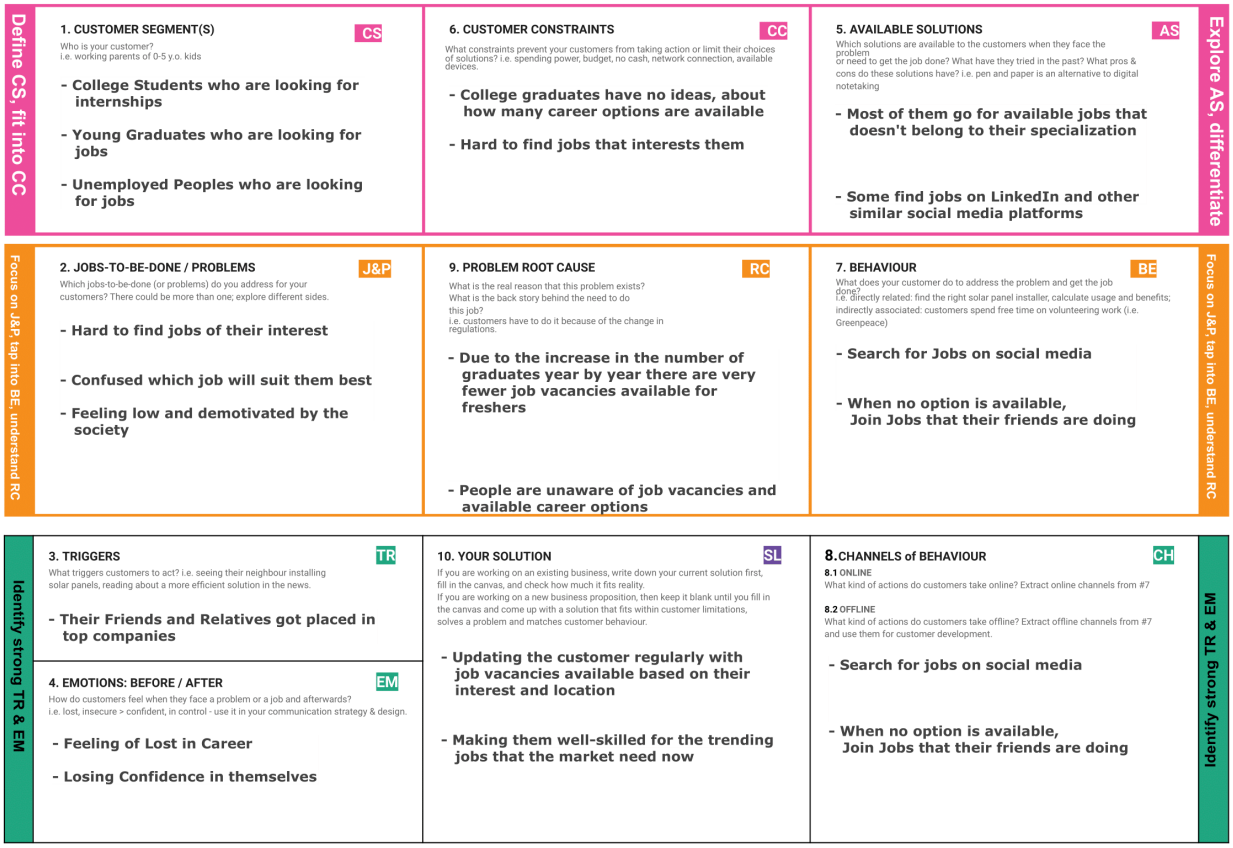
3.3. Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The dataset used for this research are sourced from Stack overflow survey data which is modelled as the user data for this research. Another dataset was created by web scrapping the Job board Using R programming language to fulfill the road map of this dissertation. The research question proposed by this research is "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?". To answer the research question, below are the objectives that need to be satisfied with going forward.
2.	Idea / Solution description	A Flask web application backed by

		powerful deep learning model and ESCO-API to recommend job based on user skill set along with recommending necessary skill set needed for the recommended job. Building more powerful AI model and better prediction accuracy. Creating a database for easy identification of skill set for recommended job.
3.	Novelty / Uniqueness	Definition of novelty is a crucial role and basis for novel recommendation. From the analysis of research status described above, researchers for the definition of novelty are not the same according to research field and data characteristics, and algorithm design of novel recommendation is also different.
4.	Social Impact / Customer Satisfaction	Customer service representatives receive and place telephone calls and help maintain solid relationships with customers by answering questions and concerns with speed and professionalism. But cashiers, salespeople, management, and billing departments can also be regarded as customer service jobs since they interact with customers.
5.	Business Model (Revenue Model)	Revenue modeling is the process of defining how a business plans to make money. A company's revenue model identifies the product or service, why it is valuable, how it can generate income and who it serves. Revenue modeling often involves incorporating multiple revenue streams into a central view of a company's finances.
6.	Scalability of the Solution	It navigates huge collections of items/jobs data and it up to the actual datasets, deals efficiency and analysis to accelerate for massive datasets.

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.



4. REQUIREMENT ANALYSIS

4.1. Functional Requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	SubRequirement (Story/ Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation via Email
FR-3	Updating UsersSkills & Interest	Getting UserSkills and Interest through form
FR-4	Job Recommendation	Recommending Job by querying data from DB2 Database

4.2. Non-Functional Requirement

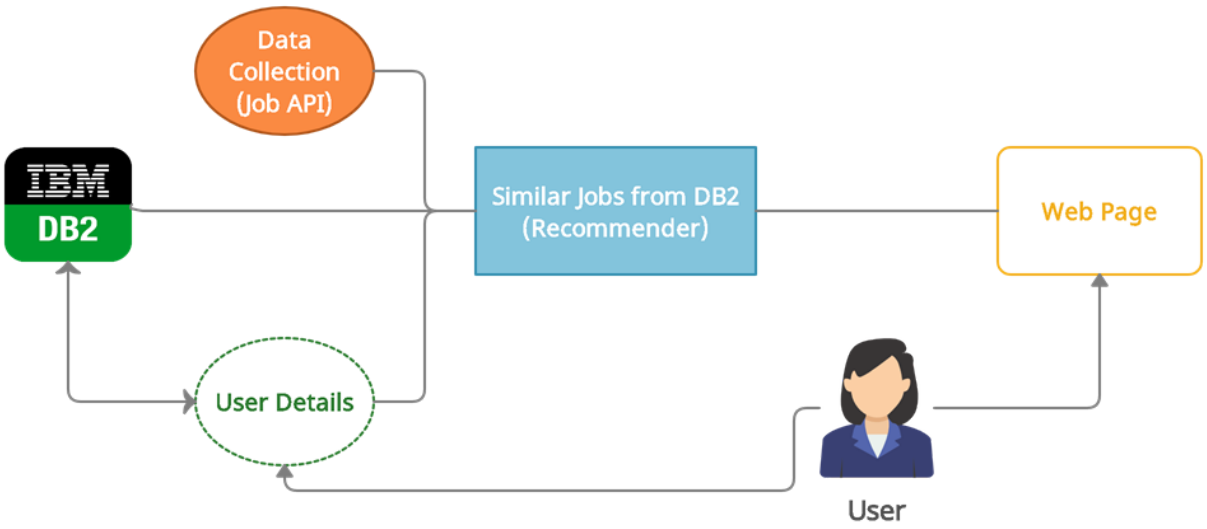
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Simple andEasy to use interface
NFR-2	**Security**	Using SSL Certificatie for DB2
NFR-3	**Reliability**	Data cannotbe leaked easily
NFR-4	**Performance**	Well Optimized application
NFR-5	**Availability**	Chatbot available to help 24/7hrs
NFR-6	**Scalability**	Can easily extend theapp as we are using kubernetes and docker containers

5. PROJECT DESIGN

5.1. Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data entersand leaves the system, what changes the information, and where data is stored.



Data Flow :

- User Registers into a form with his Interestand Skills
- Collected user details are stored in datbase
- A Job API is also used to get data of the available jobs in the market now
- User Detailsand Data from Job API is comapredand similar data are displayed on the web page as a recommendation

5.2. Solution & Technical Architecture

TechnicalArchitecture:

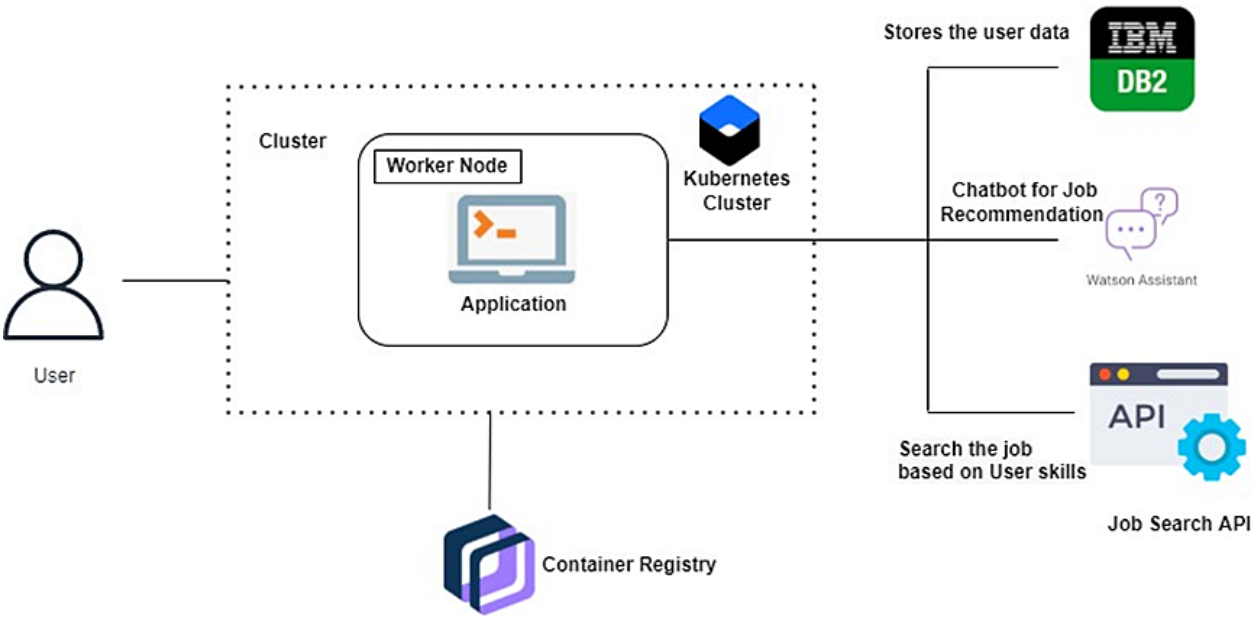


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g.Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Bootstrap etc.
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBMWatson STT service
4.	Application Logic-3	Logic for a process in the application	IBMWatson Assistant
5.	Database	Data Type, Configurations etc.	SqlAlchemy, Sqlite etc.
6.	Cloud Database	Database Service on Cloud	IBMDB2, IBM Cloudant etc.

7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	JOBAPI, etc.
9.	Infrastructure (Server/ Cloud)	Application Deployment on Local System / Cloud LocalServer Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Flask, Bootstrap, Kubernetes
2.	Security Implementations	List all the security / access controls implemented,use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier,Micro-services)	Kubernetes, docker
4.	Availability	Justify the availability of application (e.g.use of loadbalancers, distributed serversetc.)	Distributed Servers
5.	Performance	Design consideration for the performance of theapplication (number of requests per sec, use of Cache, use of CDN's)etc.	Use of CDN

-

5.3. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story/ Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email& password	I can receive confirmation email & click confirm		Sprint-1
Customer	Recommendation	USN-2	As a user, I have entered my skills and interest during the registration	Now I get job recommendat ion based on my skills and interest		Sprint-2

6. PROJECT PLANNING & SCHEDULING

6.1. Sprint Planning & Estimation

Project Tracker, Velocity& Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Points Completed (as onPlanned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		19 Nov 2022

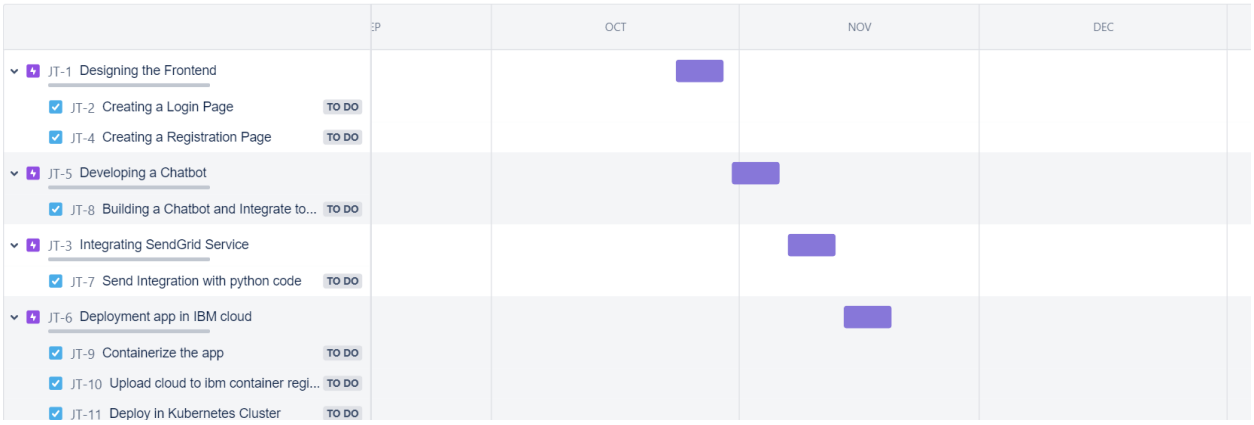
6.2. Sprint Delivery Schedule

Use the below template to create productbacklog and sprintschedule:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email,password, and confirming my password.	2	High	Satish Raj, Thirumavalavan

Sprint-2		USN-2	As a user, I will receive confirmationemail once I have registered for the application	1	High	Manigandan, Tamilselvan
Sprint-3		USN-3	As a user, I can recommend job according to user data	2	High	Satish Raj
Sprint-4	Dashboard	USN-4	As a user I canshow userdata	1	High	Satish Raj, Manigand an

6.3. Report from JIRA



7. CODING & SOLUTIONING

7.1. Feature 1

Allows job search by key skills, Companies, Destinations and Designations:

The search option is the most important feature in any job seeking apps. So while developing a job search apps make sure that there is a search option provided for the users which allows them to find out the desired job by their key skills, destination, designation and the company. There is an additional feature which you can provide the users like filtering the search which helps them to get a more productive and relevant job. The more viable and transparent the app is, the more users will be inclined. One of the transparent feature is that showing the salary structure in the search results.

Get Relevant Job Alerts:

One of the fundamental features of any applications is to get relevant job alerts. Though people are seeing and hunting for jobs doesn't mean that they can spend time on some useless stuff or unnecessary information which does not match their profile. So it is a must have feature to send the relevant jobs for the users and help them to get their dream job as they desired.

7.2. Feature 2

In-app communication:

Communication is the key factor in any domain and it is applicable even in job search applications. We know the fact that use of technology increases our comfort in fact the in-app communication make the communication more easier. Users always wanted to be heard, so while developing a job search application one should be aware that it is very important to include messaging platform to the users which helps you to increase the expansion of the product.

By conveying the communication as the key factor, we need to understand the ways of connecting to the job hunters which are

- **SMS**
- **Emails**
- **Push notifications**

These will make users updated regarding their desired jobs, new job postings, evolving trends, job openings, and many more. Once the application is submitted or in progress the job seeker is also allowed to contact the recruiters and admin with these in-app communication.

Privacy by Cloud management:

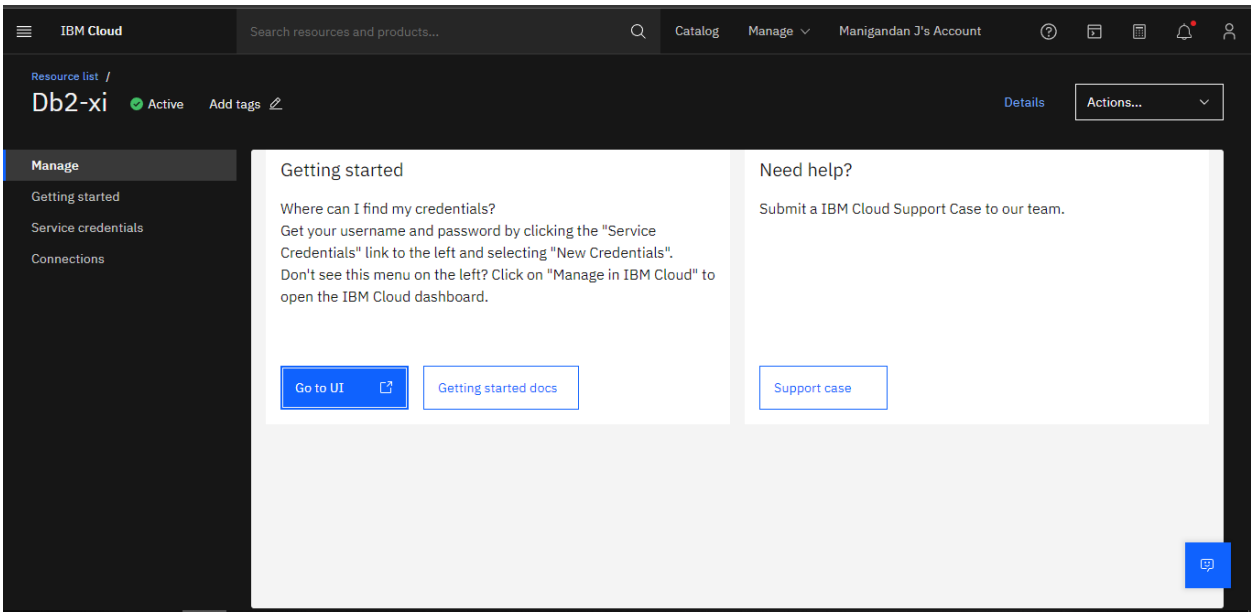
While developing the job search applications, developers should keep in mind the most vital factor which is nothing but privacy of the users. The data is managed securely by implementing the cloud technology. The data which need to protected is as follows

- **Information of the job seekers**
- **Resumes of the job seekers**
- **Profiles of Recruiters**

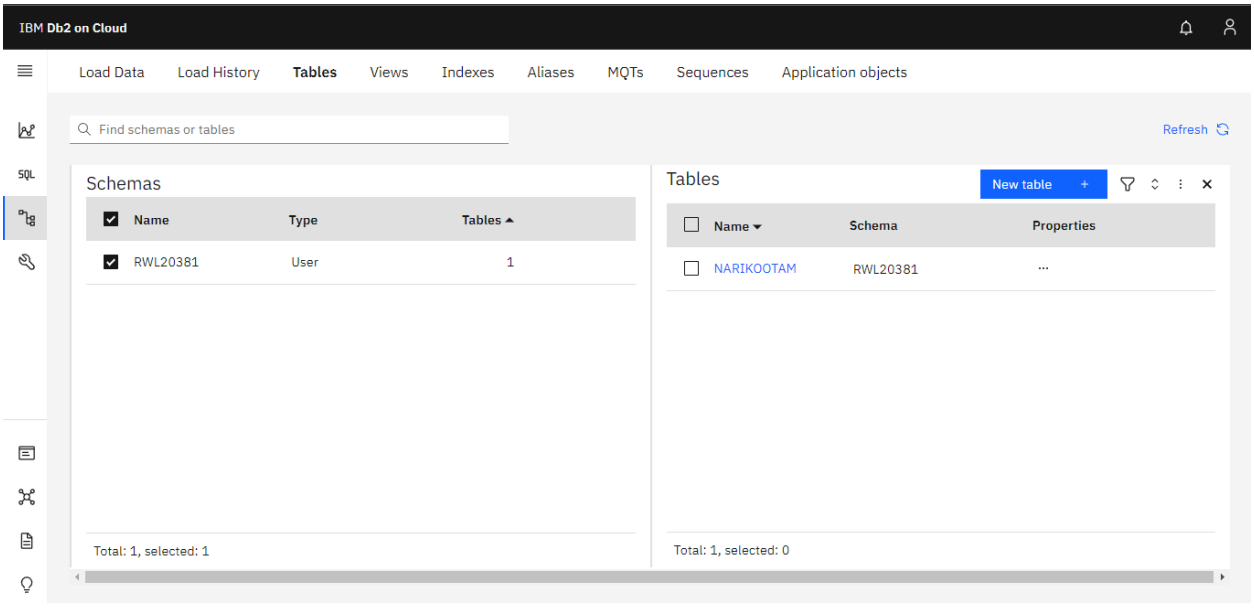
So by adopting cloud management the information is stored securely therefore protecting the entire business integrity.

7.3. Database Schema

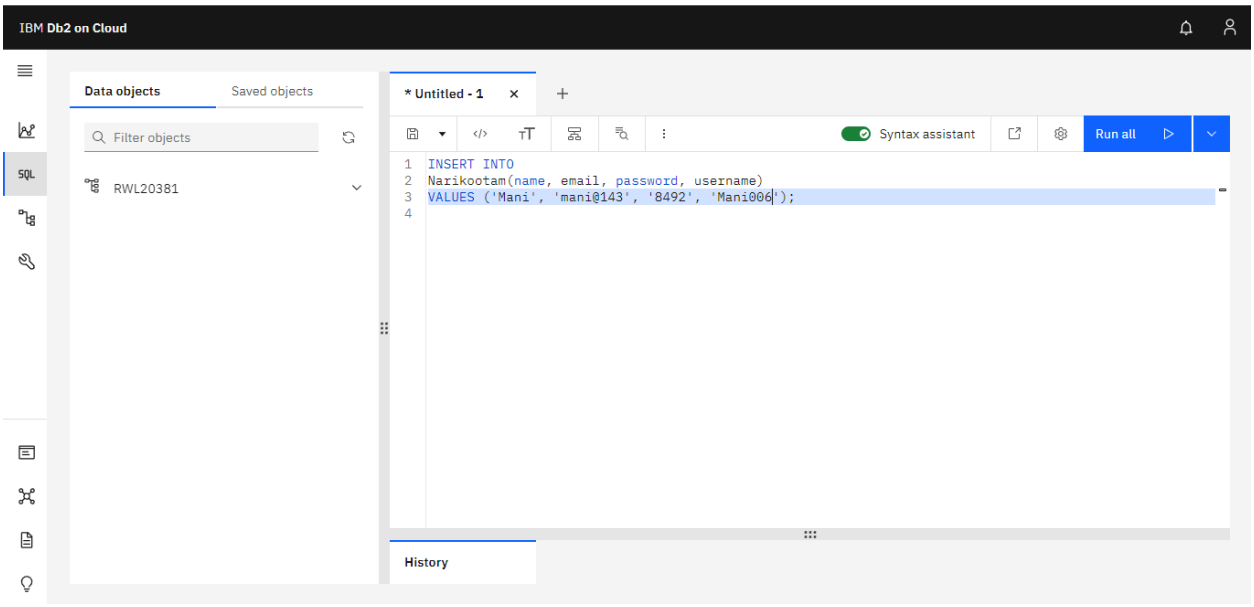
Create a Database on IBM Cloud:



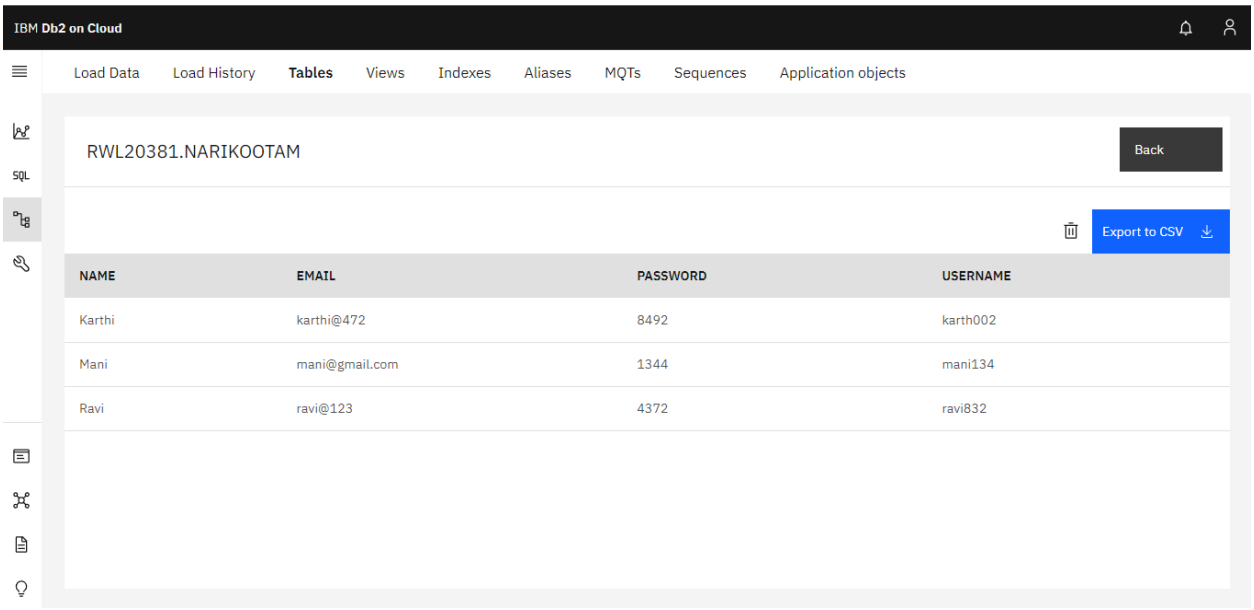
Create a new table:



INSERT INTO the Values:



View the table:



8. TESTING

8.2. User Acceptance Testing

Purpose of Document:

The purpose of this document is to briefly explain the test coverage and open issues of the Skills and Job Recommendation project at the time of the release to User Acceptance Testing (UAT).

Defect Analysis:

This report shows the number of resolved or closed bugs at each severity level, and howthey were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	1	3	1	6
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	80

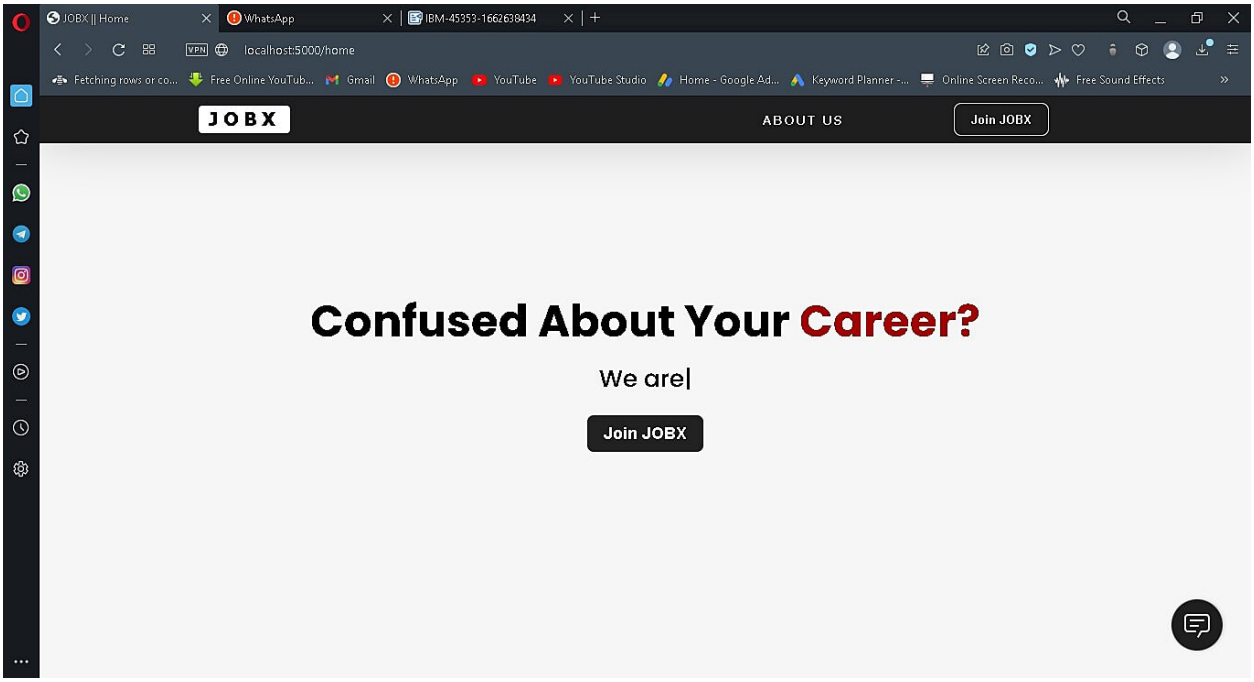
Test Case Analysis:

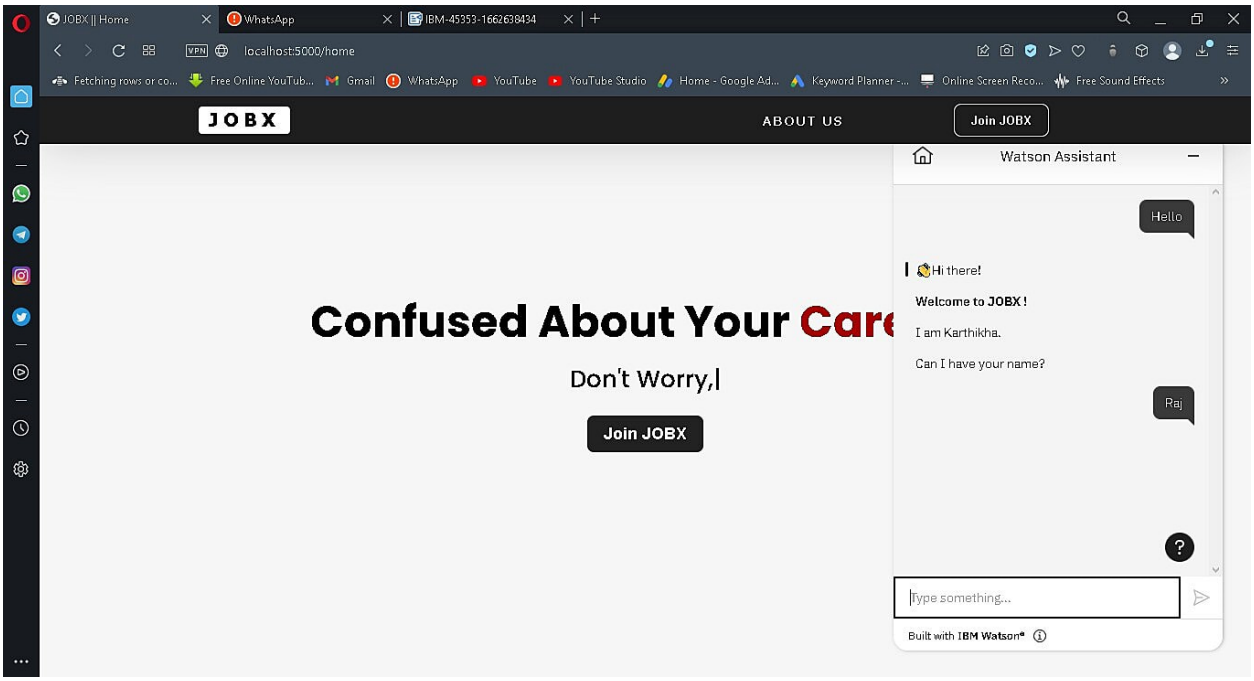
This report shows the number of test cases that have passed, failed,and untested

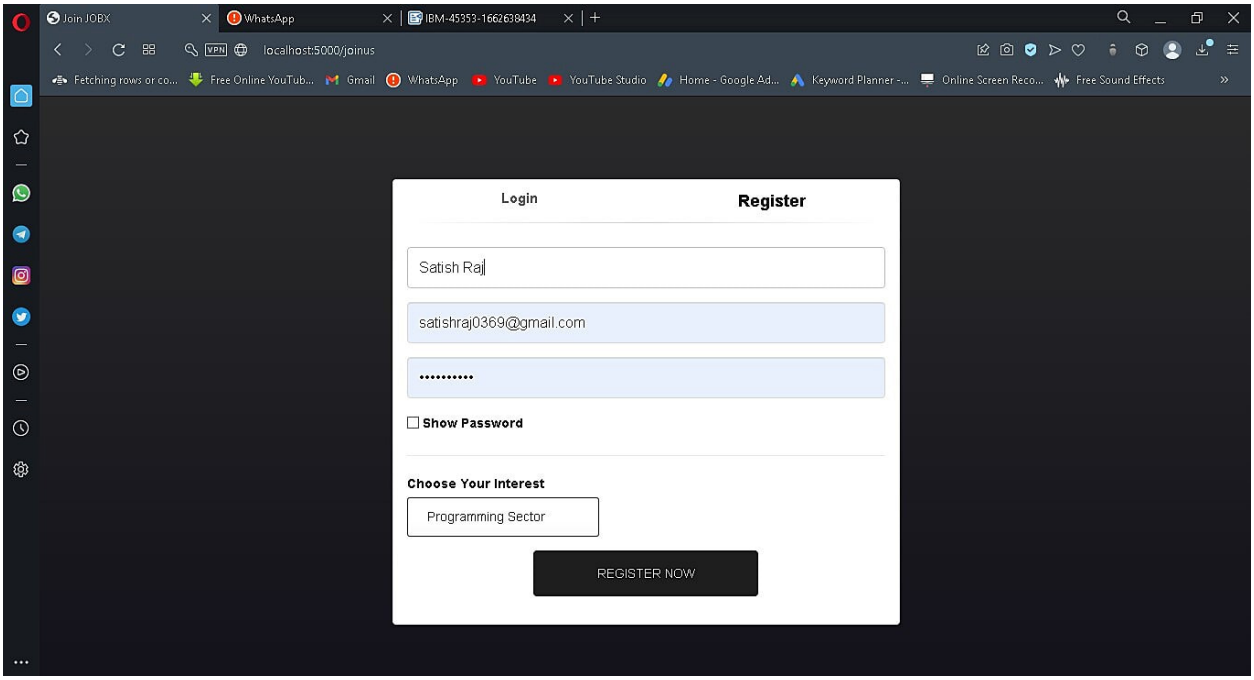
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	1	7
Client Application	51	0	1	51
Security	2	0	2	2
Outsource Shipping	3	0	1	3
Exception Reporting	9	0	1	9
Final Report Output	4	0	1	4
Version Control	2	0	0	2

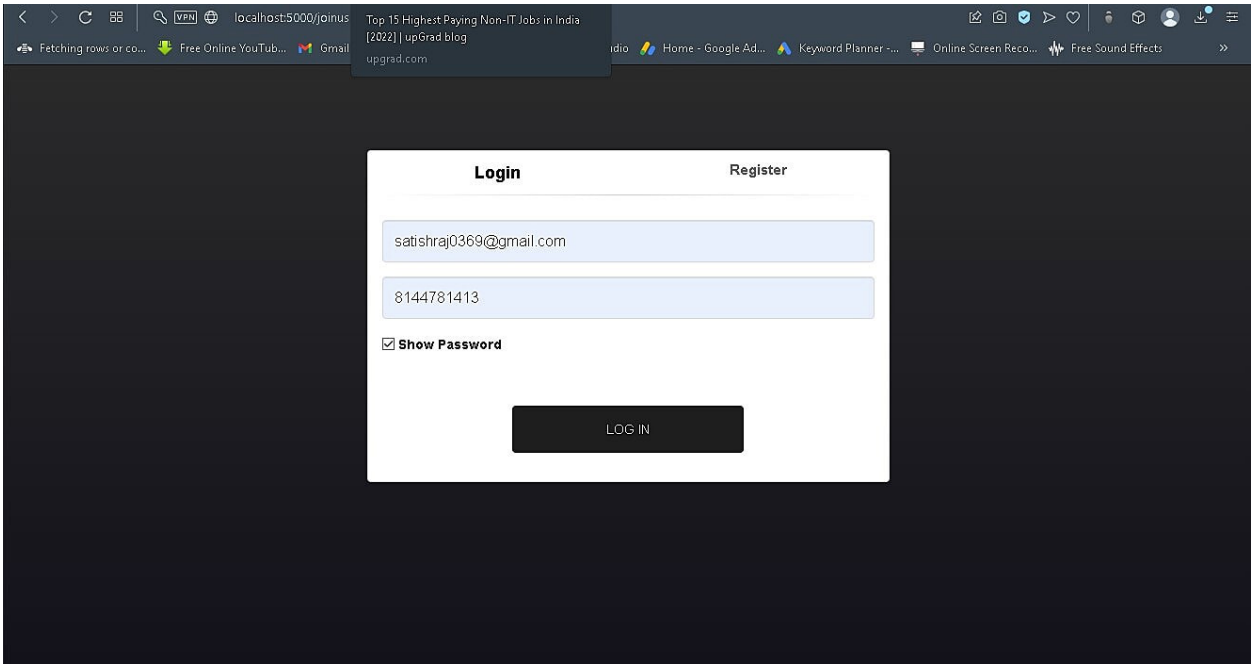
9. RESULTS

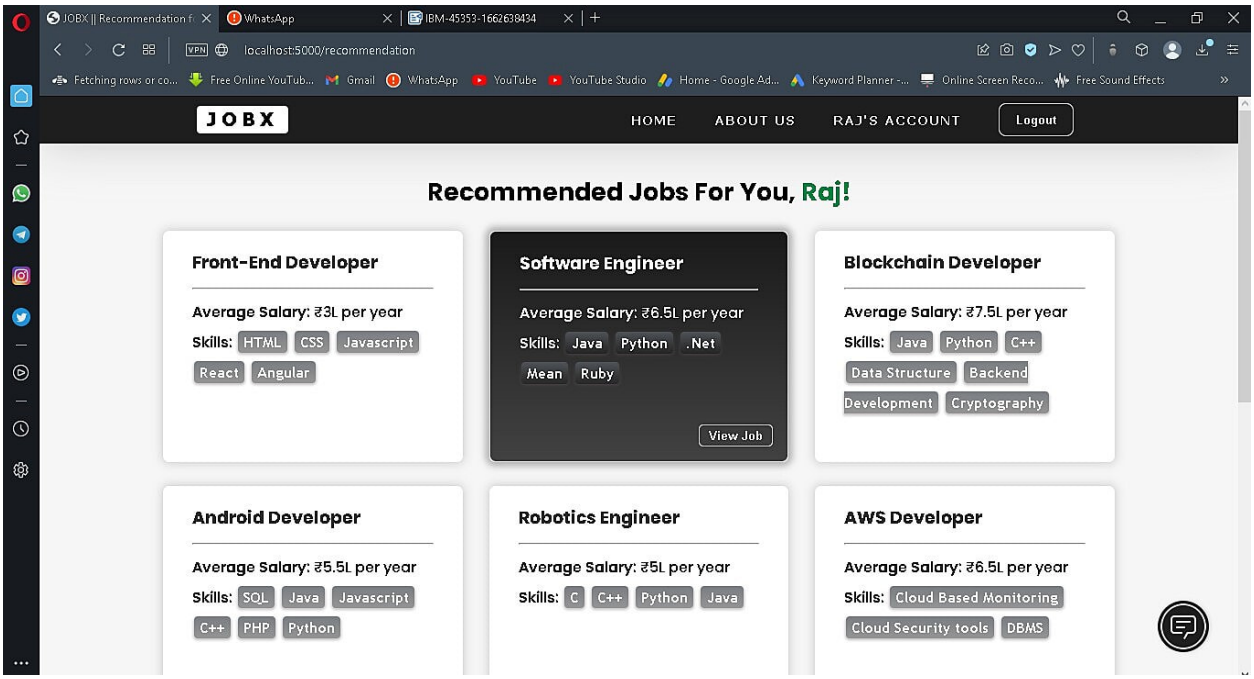
9.1. Performance Testing

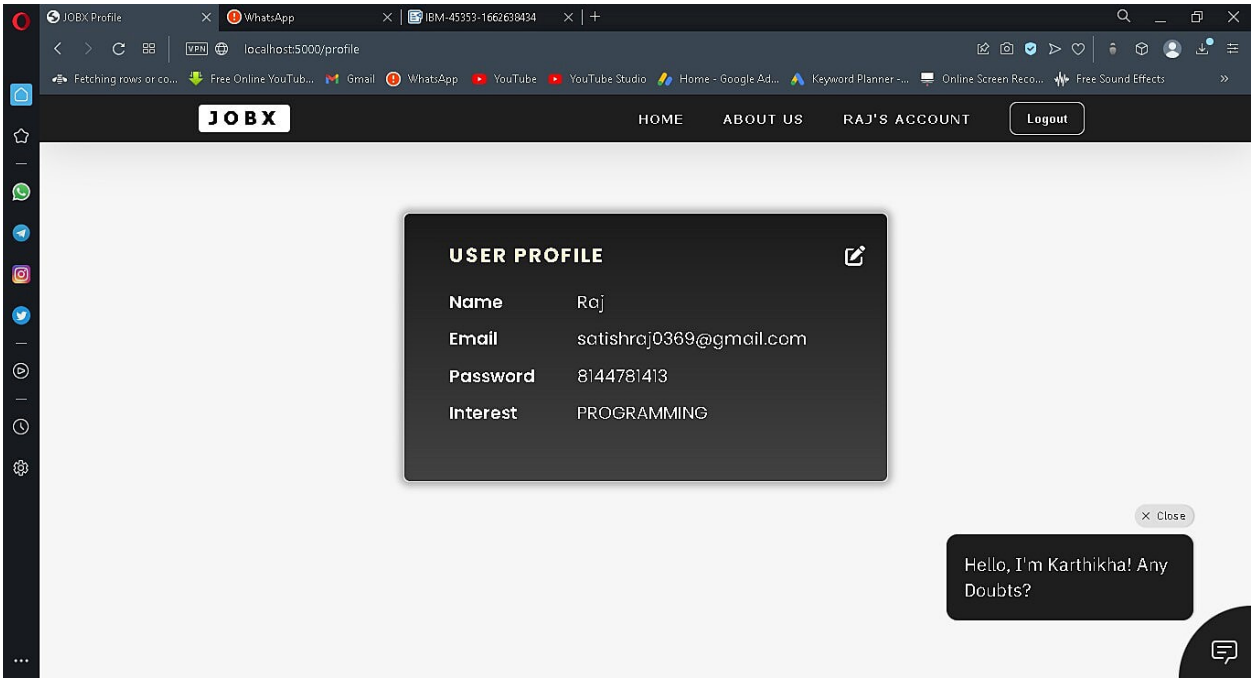


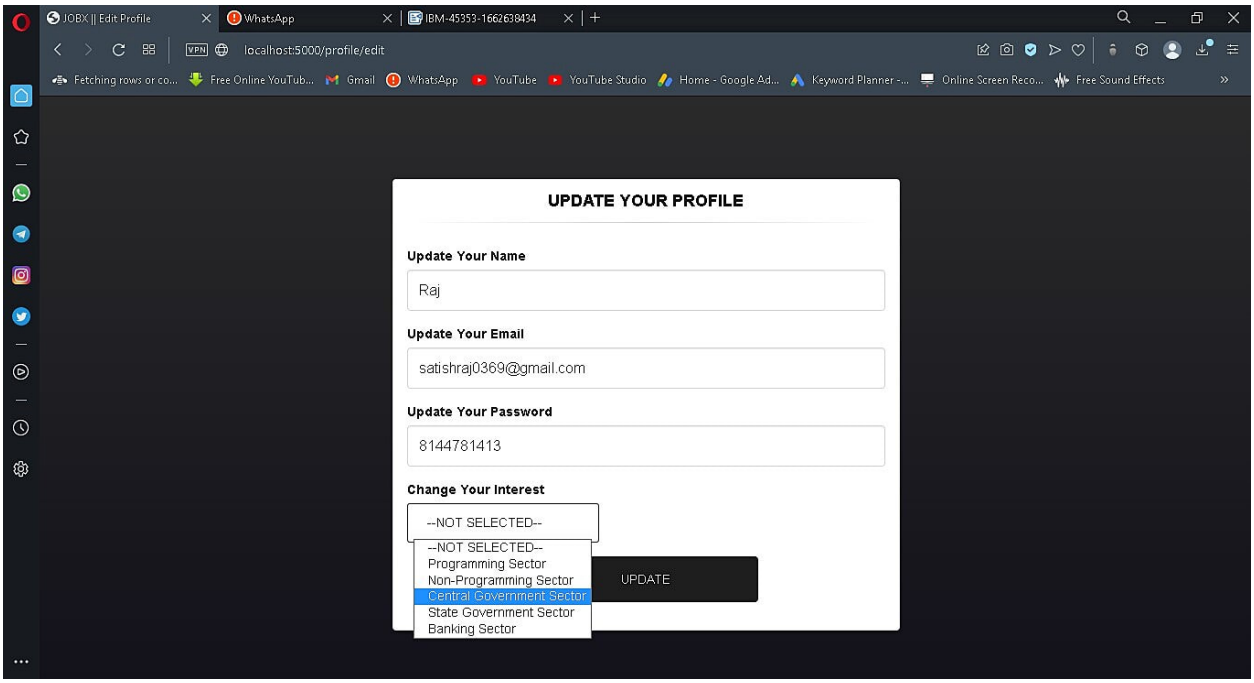



Join JOBX
WhatsApp
IBM-45353-1662638434
localhost:5000/joinus
Fetching rows or co...
Free Online YouTub...
Gmail
WhatsApp
YouTube
YouTube Studio
Home - Google Ad...
Keyword Planner -...
Online Screen Reco...
Free Sound Effects
Login
Register
Satish Raj
satishraj0369@gmail.com
Show Password
Choose Your Interest
Programming Sector
REGISTER NOW


localhost:5000/joinus
Top 15 Highest Paying Non-IT Jobs in India [2022] | upGrad blog
upgrad.com
Fetching rows or co...
Free Online YouTub...
Gmail
Home - Google Ad...
Keyword Planner -...
Online Screen Reco...
Free Sound Effects
Login
Register
satishraj0369@gmail.com
8144781413
Show Password
LOG IN


JOBX || Recommendation f
WhatsApp
IBM-45353-1662638434
localhost:5000/recommendation
JOBX
HOME
ABOUT US
RAJ'S ACCOUNT
Logout
Recommended Jobs For You, Raj!
Front-End Developer
Average Salary: ₹3L per year
Skills: HTML CSS Javascript React Angular
Software Engineer
Average Salary: ₹6.5L per year
Skills: Java Python .Net Mean Ruby
View Job
Blockchain Developer
Average Salary: ₹7.5L per year
Skills: Java Python C++ Data Structure Backend Development Cryptography
Android Developer
Average Salary: ₹5.5L per year
Skills: SQL Java Javascript C++ PHP Python
Robotics Engineer
Average Salary: ₹5L per year
Skills: C C++ Python Java
AWS Developer
Average Salary: ₹6.5L per year
Skills: Cloud Based Monitoring Cloud Security tools DBMS


JOBX Profile
WhatsApp
IBM-45353-1662638434
localhost:5000/profile
JOBX
HOME
ABOUT US
RAJ'S ACCOUNT
Logout
USER PROFILE
Name Raj
Email satishraj0369@gmail.com
Password 8144781413
Interest PROGRAMMING
Close
Hello, I'm Karthikha! Any Doubts?


JOBX || Edit Profile
WhatsApp
IBM-45353-1662638434
localhost:5000/profile/edit
Fetching rows or co...
Free Online YouTub...
Gmail
WhatsApp
YouTube
YouTube Studio
Home - Google Ad...
Keyword Planner -...
Online Screen Reco...
Free Sound Effects
UPDATE YOUR PROFILE
Update Your Name
Raj
Update Your Email
satishraj0369@gmail.com
Update Your Password
8144781413
Change Your Interest
--NOT SELECTED--
--NOT SELECTED--
Programming Sector
Non-Programming Sector
Central Government Sector
State Government Sector
Banking Sector
UPDATE

10. ADVANTAGES & DISADVANTAGES

Advantages:

- Users will only get recommendations related to their preferences in their profile, and recommender engine may never recommend any item with other characteristics.
- As it is based on similarity among items and users, it is not easy to find the neighbour users.
- Use ontology to categorize jobs and as a knowledge base to define features (attenuate cold-start problem).
- Use two levels in skills matching (constrains and preferences).
- Use linguistic variables to determine skill levels.
- Various information retrieval techniques are used.
- Use integration-based similarity in skills matching (explicit and implicit preferences).

Disanvantages:

- In other words, the model has limited ability to expand on the users' existing interests.
- Key words search method.
- The model can only make recommendations based on existing interests of the user.
- Knowledge acquisition and knowledge engineering problems.
- Tools and technologies skills excluded.
- Scalability, ramp-up, and data sparsity problems.
- No relational aspects are included.

11. CONCLUSION

we proposed a framework for job recommendation task. This framework facilitates the understandingof job recommendation process as well as it allows the use of a variety of text processing and recommendationmethods according to the preferences of the job recommender system designer. Moreover, we also contribute making publicly available a new dataset containing job seekers pro les and job vacancies.

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.

12. FUTURE SCOPE

For this system to be hybrid, content-based filtering is required, which can only recommend jobs based on the user's current profile. It cannot deliver anything surprising based on the user's past searches. This paper also uses collaborative filtering which faces well-known problems of privacy breaches and cold start. The system has a broad scope that can be used to make it more robust and foolproof. Firstly, automating the crawling process is required, when a new company is added to the database. In other words, removing the one-time configuration step/process to fetch jobs of a particular new company can be done. These models can implement techniques such as KNN in collaborative filtering. Implementing NLP in content-based filtering for better and more accurate search matching can be done. Along with this, testing and collecting more user data for better performance of the collaborative filtering module is required. Lastly, improving the cleansing process of the job description and using natural language processing are required. While using collaborative filtering, this work can be improved by giving different weights to different users based on their LinkedIn skills.

13. APPENDIX

Source code:

app.py

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
import ibm_db
import random
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
from dotenv import load_dotenv
from data import *
from mailtemplate import *

app = Flask(__name__)

load_dotenv()
# Environment Variables
FROM_EMAIL = os.getenv('FROM_EMAIL')
SENDGRID_API_KEY = os.getenv('SENDGRID_API_KEY')
IBM_DB_URL = os.getenv('IBM_DB_URL')
SECRET_KEY = os.getenv('SECRET_KEY')

# Setting Secret Key for Sessions
app.secret_key = SECRET_KEY

# Creating IBM DB Connection
conn = ibm_db.connect(IBM_DB_URL,'','')
```

```
# Home Route
@app.route("/")
@app.route("/home")
def home():
    return render_template('index.html', title= "JOBX || Home")

# Recommendation Route
@app.route("/recommendation")
def recommendation():

    if "name" in session and 'interest' in session:
        name = session["name"]
        interest = session['interest']
return  render_template('recommendation.html',  title="JOBX  ||  Recommendation
for you.", name = name, user_interest = interest, random = random,job=job)
    else:
        flash("You are not Logged In!")
        return redirect(url_for("joinus"))

# Joinus Route
@app.route("/joinus")
def joinus():
    if "name" in session:
        flash("Already Logged In!")
        return redirect(url_for('recommendation'))
    return render_template('joinus.html', title = "Join JOBX")

# Login Route
```

```
@app.route("/login", methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        email = request.form["email"]
        password = request.form["password"]

        sql = "SELECT * FROM USERS WHERE email=?"
        prep_stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(prep_stmt, 1, email)
        ibm_db.execute(prep_stmt)

        # Checking if we are getting rows or not
        noData = False
        try:
            # Getting the Row as dictionary
            dictionary = ibm_db.fetch_assoc(prep_stmt)
        except:
            pass

        if dictionary is False:
            noData = True
        else:
            noData = False

        if noData is False:
            if password == dictionary['PASSWORD']:
                session['logged_in'] = True
                if "name" in session:
```

```
            session.pop("name", None)
            session.pop("email", None)
            session.pop("interest", None)

            name = dictionary['NAME']
            session["name"] = name

            email = dictionary['EMAIL']
            session["email"] = email

            interest = dictionary['INTERESTS']
            session["interest"] = interest

            return redirect(url_for('recommendation'))
        else:
            name = dictionary['NAME']
            session["name"] = name

            interest = dictionary['INTERESTS']
            session["interest"] = interest

            email = dictionary['EMAIL']
            session["email"] = email

            return redirect(url_for('recommendation'))
    else:
        error = "Invalid Password"
        return render_template("joinus.html",error= error)
else:
```

```
            error = "Invalid Email"
            return render_template("joinus.html",error= error)

    else:
        if "name" in session:
            flash("Already Logged In!")
            return redirect(url_for('recommendation'))
        return redirect(url_for("joinus"))



# Register Route
@app.route("/register", methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        uname = request.form["name"]
        email = request.form["email"]
        password = request.form["password"]
        interest = request.form["interest"]

        if uname[0].isupper():
            name = uname
        else:
            name = uname.capitalize()

        # Initializing Session
        session["name"] = name
        session["email"] = email
        session["interest"] = interest
        session['logged_in'] = True
```

```
# Checking If User Email Already Exists
sql = "SELECT * FROM USERS WHERE email=?"
prep_stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(prep_stmt, 1, email)
ibm_db.execute(prep_stmt)

# Checking if we are getting rows or not
noData = False
try:
    # Getting the Row as dictionary
    dictionary = ibm_db.fetch_assoc(prep_stmt)
except:
    pass

if dictionary is False:
    noData = True
else:
    noData = False

if noData is True:
    sql = "INSERT INTO USERS VALUES (?,?,?,?)"
    try:
        prep_stmt = ibm_db.prepare(conn, sql)
    except:
        pass
    ibm_db.bind_param(prep_stmt, 1, name)
    ibm_db.bind_param(prep_stmt, 2, email)
    ibm_db.bind_param(prep_stmt, 3, password)
```

```
            ibm_db.bind_param(prep_stmt, 4, interest)
            try:
                ibm_db.execute(prep_stmt)
            except:
                pass

            # Sending Register Successful Mail to the user
            message = Mail(
            from_email= FROM_EMAIL,
            to_emails= email,
            subject='Welcome to JOBX',
            html_content= mailtemplate )

            sg = SendGridAPIClient(
            api_key= SENDGRID_API_KEY)
            try:
                response = sg.send(message)
            except Exception:
                pass
            print("Mail Sent and response code is ",response.status_code)

        else:
            error = "User Already Exists!"
            return render_template("joinus.html",error= error)

    return redirect(url_for('recommendation'))


#Upload
```

```
@app.route("/upload")
def upload():
    return render_template("upload.html", title= "JOBX || Admin Panel")

#Profile
@app.route("/profile")
def profile():
    if session['logged_in'] == True:
        email = session['email']
        sql = "SELECT * FROM USERS WHERE email=?"
        prep_stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(prep_stmt, 1, email)
        ibm_db.execute(prep_stmt)
        # Checking if we are getting rows or not
        try:
            # Getting the Row as dictionary
            dictionary = ibm_db.fetch_assoc(prep_stmt)
        except:
            pass
    else:
        flash("Please Login")
        return redirect(url_for("joinus"))



    return render_template("profile.html", title = "JOBX Profile", data = dictionary)

#Edit Profile
@app.route('/profile/edit')
def edit():
```

```
    if session['logged_in'] == True:
        email = session['email']
        # Fetching Data from DB
        sql = "SELECT * FROM USERS WHERE email=?"
        prep_stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(prep_stmt, 1, email)
        ibm_db.execute(prep_stmt)
        try:
            # Getting the Row as dictionary
            dictionary = ibm_db.fetch_assoc(prep_stmt)
        except:
            pass

            return render_template('edit.html', title = "JOBX || Edit Profile", dictionary =
dictionary)
```

#Update Profile

```
@app.route("/update", methods=['GET', 'POST'])
def update():
    email = session['email']
    # Getting Data From Update Form
    if request.method == 'POST':
        u_uname = request.form["name"]
        u_email = request.form["email"]
        u_password = request.form["password"]
        u_interest = request.form["interest"]
        if u_uname[0].isupper():
            u_name = u_uname
        else:
```

```
        u_name = u_uname.capitalize()

    # Checking if data is update or not
    sql = "SELECT * FROM USERS WHERE email=?"
    prep_stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(prep_stmt, 1, email)
    ibm_db.execute(prep_stmt)
    try:
        # Getting the Row as dictionary
        acnt = ibm_db.fetch_assoc(prep_stmt)
    except:
        pass

                if acnt['NAME'] == u_uname and acnt['EMAIL'] == u_email and
acnt['PASSWORD'] == u_password and acnt['INTERESTS'] == u_interest:
        flash("Nothing Updated")
        return redirect(url_for('home'))
    else:
        # Updating Database
          sql = "UPDATE USERS SET NAME = ?, EMAIL= ?, PASSWORD= ?, INTERESTS= ?
WHERE EMAIL = ?;"
        prep_stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(prep_stmt, 1, u_name)
        ibm_db.bind_param(prep_stmt, 2, u_email)
        ibm_db.bind_param(prep_stmt, 3, u_password)
        ibm_db.bind_param(prep_stmt, 4, u_interest)
        ibm_db.bind_param(prep_stmt, 5, email)
        try:
            ibm_db.execute(prep_stmt)
```

```
            except Exception as e:
                pass
        # Sending Register Successful Mail to the user
        message = Mail(
        from_email= FROM_EMAIL,
        to_emails= u_email,
        subject='JOBX Profile Updated',
                        html_content= f"<h3 style='background-color: rgb(28, 28, 28);color:aliceblue;font-family: Verdana, Geneva, Tahoma, sans-serif;'>PROFILE UPDATED</h3><table style='width:100%;margin:0 auto'><tr><td>Name </td><td>{u_name}</td></tr><tr><td>Email </td><td>{u_email}</td></tr><tr><td>Password </td><td>{u_password}</td></tr><tr><td>Interests </td><td>{u_interest}</td></tr></table>" )
        sg = SendGridAPIClient(
        api_key= SENDGRID_API_KEY)
        try:
            response = sg.send(message)
        except Exception:
            pass
        print("Mail Sent and response code is ",response.status_code)
        # Loggin out user to login again with new details
        flash(f"Login with your updated details, {u_uname}")
        session.pop("name", None)
        session.pop("email", None)
        session.pop("interest", None)
        session['logged_in'] = False
        return redirect(url_for("home"))
```

 # Logout

```
@app.route("/logout")
def logout():
    if "name" in session and "interest" in session:
        name = session["name"]
        flash(f"You have been logged out, {name}")
        session.pop("name", None)
        session.pop("email", None)
        session.pop("interest", None)
        session['logged_in'] = False
        return redirect(url_for("home"))
    else:
        flash(f"You are not Logged In!")
        return redirect(url_for("home"))
# About Route
@app.route("/about")
def about():
    return render_template('about.html', title="About JOBX")
# Handling Errors
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html',title = "Page Not Found!"), 404

# Running the App
if __name__ == "__main__":
    app.run(debug=True)
```

about.html

```
{% extends 'layout.html' %}
{% block content %}
<!-- User Welcome -->
<div class="text-container">
                <h1>Confused About Your <span style="color:rgb(165, 0, 0)
;">Career?</span></h1>
   <p><span class="auto-input"></span></p>
<!-- Showing JOIN button if not logged in-->
   {% if session['logged_in'] %}
                    <button id="btn"><a href="{{url_for('recommendation')}}">Your
Recommendations</a></button>
   {% else %}
   <button id="btn"><a href="{{url_for('joinus')}}">Join JOBX</a></button>
   {% endif %}
</div>
<!-- Auto Type Text Script -->
<script src="https://cdn.jsdelivr.net/npm/typed.js@2.0.12"></script>
<script>
   var typed = new Typed(".auto-input", {
      strings: ["Don't Worry,", "We are here for you ;)", "We Show Who you are?",
         "We Show Who you could be?"
      ],
      typeSpeed: 100,
      backSpeed: 100,
      loop: true,
   });
</script>
{% endblock content %}
```

joinus.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   {% if title %}
   <title>{{ title }}</title>
   {% else %}
   <title>JOBX</title>
   {% endif %}

   <!-- BootStrap CDN -->
      <link  href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
                                                                  <script                defer
src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/js/bootstrap.min.js"></script>

   <!-- CSS -->
   <link rel="stylesheet" href="{{url_for('static',filename='css/joinus.css')}}">

   <!-- Script -->
   <script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
   <script defer src="{{url_for('static',filename='JS/joinus.js')}}"></script>
</head>

<body>
```

```
<div class="container">
  <div class="row">
    <div class="col-md-6 col-md-offset-3">
      <div class="panel panel-login">
        <div class="panel-heading">
          <div class="row">
            <div class="col-xs-6">
              <a href="#" class="active" id="login-form-link">Login</a>
            </div>
            <div class="col-xs-6">
              <a href="#" id="register-form-link">Register</a>
            </div>
          </div>
          <hr>
        </div>
        <div class="panel-body">
          <div class="row">
            <div class="col-lg-12">
              <form id="login-form" action="/login" method="post" role="form"
style="display: block;">
                <div class="form-group">
                  <input type="email" name="email" id="email" tabindex="1"
class="form-control"
                    placeholder="Email" required>
                </div>
                <div class="form-group">
                  <input type="password" name="password" id="password-
login" tabindex="2"
                    class="form-control" placeholder="Password" required>
```

```
                    </div>
                    <div class="form-group">
                                    <label><input id="checkbox1" type="checkbox"
onclick="showPassLogin()"> Show
                        Password</label>
                    </div>
                    <!-- <div class="form-group text-center">
                                    <input type="checkbox" tabindex="3" class=""
name="remember" id="remember">
                      <label for="remember"> Remember Me</label>
                    </div> -->
                    <div style="text-align:center;" class="form-group">
                                    <label style="font-size: 18px; color:rgb(198, 0,
0)">{{error}}</label>
                      <!-- Message Box -->
                      <div class="msg-box">
                        {% with messages = get_flashed_messages() %}
                        {% if messages %}
                        {% for msg in messages %}
                                    <label style="font-size: 18px; color:rgb(231, 134,
0)">{{msg}}</label>
                        {% endfor %}
                        {% endif %}
                        {% endwith %}
                      </div>
                    </div>
                    <div class="form-group">
                      <div class="row">
                        <div class="col-sm-6 col-sm-offset-3">
```

```
                                    <input type="submit" name="login-submit" id="login-
submit" tabindex="4"
                                        class="form-control btn btn-login" value="Log In">
                                </div>
                            </div>
                        </div>
                        <!-- <div class="form-group">
                            <div class="row">
                                <div class="col-lg-12">
                                    <div class="text-center">
                                                <a href="#" tabindex="5" class="forgot-
password">Forgot
                                            Password?</a>
                                    </div>
                                </div>
                            </div>
                        </div> -->
                    </form>
                        <form id="register-form" action="/register" method="post"
role="form"
                        style="display: none;">
                        <div class="form-group">
                            <input type="text" name="name" id="name" class="form-
control" placeholder="Name"
                            required autocomplete="off">
                        </div>
                        <div class="form-group">
                            <input type="email" name="email" id="email" class="form-
control"
```

```
                                                placeholder="Email Address" required
autocomplete="off">
                        </div>
                        <div class="form-group">
                              <input type="password" name="password" id="password-
reg" class="form-control"
                            placeholder="Password" required autocomplete="off">
                        </div>
                        <div class="form-group">
                                        <label><input id="checkbox2" type="checkbox"
onclick="showPassReg()"> Show
                            Password</label>
                        </div>
                        <!-- Interest Section-->
                        <hr>
                        <div class="form-group">
                          <label for="interest-select">Choose Your Interest</label>
                          <div class="select">
                            <select name="interest" id="interest-select" required>
                                  <option value="programming" selected>Programming
Sector</option>
                                <option value="non-programming">Non-Programming
Sector</option>
                                          <option value="central-government">Central
Government Sector</option>
                                  <option value="state-government">State Government
Sector</option>
                                <option value="banking">Banking Sector</option>
                            </select>
```

```
                    </div>
                </div>


                <!--
                <div class="form-group">
                    <label for="interest">Choose Your Interest</label>
                </div>
                <div class="form-group">
                    <select name="interest" id="interest">
                            <option value="programming" selected>Programming
Sector</option>
                            <option value="non-programming">Non-Programming
Sector</option>
                    </select>
                </div> -->

                <div class="form-group">
                    <div class="row">
                        <div class="col-sm-6 col-sm-offset-3">
                                <input type="submit" name="register-submit"
id="register-submit"
                            tabindex="4" class="form-control btn btn-register"
                            value="Register Now">
                        </div>
                    </div>
                </div>
            </form>
        </div>
```

```
                </div>
            </div>
        </div>
    </div>
</div>

<script>
    passLogin = document.getElementById("password-login");
    passReg = document.getElementById("password-reg");
    checkbox1 = document.getElementById("checkbox1");
    checkbox2 = document.getElementById("checkbox2");

    const showPassLogin = function () {
        if (checkbox1.checked) {
            if (passLogin.type === "password") {
                passLogin.type = "text";
            } else {
                passLogin.type = "password";
            }
        } else {
            passLogin.type = "password";
        }
    }
    const showPassReg = function () {
        if (checkbox2.checked) {

            if (passReg.type === "password") {
                passReg.type = "text";
```

```
            } else {
                passReg.type = "password";
            }
        } else {
            passReg.type = "password";
        }
    }
  </script>
</body>

</html>
```

layout.html:

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  {% if title %}
  <title>{{ title }}</title>
  {% else %}
  <title>JOBX</title>
  {% endif %}
  <!-- CSS -->
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.1/css/all.min.css"
                                                                integrity="sha512-MV7K8+y+gLIBoVD59lQIYicR65iaqukzvf/nwasF0nqhPay5w/9lJmVM2hMDcnK1OnMGCdVK+iQrJ7lzPJQd1w=="
```

```
      crossorigin="anonymous" referrerpolicy="no-referrer" />
   <link rel="stylesheet" href="{{ url_for('static', filename='css/index.css')}}">
</head>

<body>
   <div id="preloader"></div>
   <!-- NavBar -->
   <nav>
      <div class="logo">
         <h4><a href="{{ url_for('home') }}">JOBX</a></h4>
      </div>
      <ul class="nav-links">
         <!-- Not showing Home button in home -->
         {% if "Home" in title %}
         <li></li>
         {% else %}
         <li><a href="{{ url_for('home') }}">Home</a></li>
         {% endif %}

         <li><a href="{{ url_for('about') }}">About Us</a></li>

         <!-- Showing Login/Logout button -->
         {% if session['logged_in'] %}
         <!-- Logged in -->
         <li><a href="{{ url_for('profile') }}">{{ session['name'] }}'s Account</a></li>
                                    <li><a href="{{ url_for('logout') }}"> <button
id="btn">Logout</button></a></li>
         {% else %}
         <!-- Not Logged in -->
```

```
                        <li><a href="{{ url_for('joinus') }}"> <button id="btn">Join JOBX</button></a></li>
                    {% endif %}
                </ul>
                <div class="burger">
                    <div class="line1"></div>
                    <div class="line2"></div>
                    <div class="line3"></div>
                </div>
            </nav>
            <!-- Message Box -->
            <div class="msg-box">
                {% with messages = get_flashed_messages() %}
                {% if messages %}
                {% for msg in messages %}
                <p>{{ msg }}</p>
                {% endfor %}
                {% endif %}
                {% endwith %}
            </div>
            {% block content %}

            {% endblock content %}

            <!-- Script for Preloader -->
            <script>
                const loader = document.getElementById('preloader')
                window.addEventListener('load', function () {
                    loader.style.display = "none"
```

```
    })
  </script>
  <!-- Script for Navbar -->
  <script src="{{url_for('static', filename='JS/navbar.js')}}"></script>
  <!-- Script for chatbot -->
  <script>
    window.watsonAssistantChatOptions = {
        integrationID: "8a1c12e4-59cb-47c6-9190-7822515b0841", // The ID of this integration.
        region: "jp-tok", // The region your integration is hosted in.
        serviceInstanceID: "fcbf8e08-813e-433f-af7d-38592fcc96e3", // 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>
</body>

</html>
```

recommendation.html:

```
{% extends 'layout.html' %}
{% block content %}

<!-- Recommender Section -->
<h2 class="heading">Recommended Jobs For You, <a href="{{url_for('profile')}}"
      style="color:rgb(0, 135, 61) ;">{{name}}!</a></h2>
<div class="card-container">



   {% for interest,data in job.items() %}
   {% if user_interest == interest %}
   <span style="display:none;">{{ random.shuffle(data) }}</span>
   {% for job in data %}
   <div class="card-items">
      <h3 class="job-title">{{job['title']}}</h3>
      <hr>
      <p class="job-details">
         <label>Average Salary: </label>{{job['salary']}}
      </p>
      <p class="job-details">
            {% if interest == 'central-government' or interest == 'state-government' or
interest == 'banking' %}
         <label>Exams: </label>
         {% else %}
         <label>Skills: </label>
         {% endif %}
```

```
            {% for skill in job['skills'] %}
            <span class="tag">{{skill}}</span>
            {% endfor %}
          </p>
          <button><a href="{{job['link']}}" target="_blank">View Job</a></button>
    </div>
    {% endfor %}
    {% endif %}
    {% endfor %}
</div>
{% endblock content %}
```

edit.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    {% if title %}
    <title>{{ title }}</title>
    {% else %}
    <title>JOBX</title>
    {% endif %}

    <!-- BootStrap CDN -->
      <link  href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
```

```
                                                    <script defer
src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/js/bootstrap.min.js"></script>

    <!-- CSS -->
    <link rel="stylesheet" href="{{url_for('static',filename='css/joinus.css')}}">

    <!-- Script -->
    <script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
    <script defer src="{{url_for('static',filename='JS/joinus.js')}}"></script>
</head>

<body>
    <div class="container">
        <div class="row">
            <div class="col-md-6 col-md-offset-3">
                <div class="panel panel-login">
                    <div class="panel-heading">
                        <div class="row">
                                <a href="#" class="active" id="login-form-link">UPDATE YOUR
PROFILE</a>

                        </div>
                        <hr>
                    </div>
                    <div class="panel-body">
                        <div class="row">
                            <div class="col-lg-12">
                                        <form id="login-form" action="{{url_for('update')}}"
method="post" role="form"
```

```
                    style="display: block;">
                    <div class="form-group">
                        <label for="name">Update Your Name</label>
                            <input type="text" name="name" id="name" class="form-
control" placeholder="Name"
                                value="{{dictionary['NAME']}}" required>
                    </div>
                    <div class="form-group">
                        <label for="email">Update Your Email</label>
                            <input type="text" name="email" id="email" class="form-
control"
                                    placeholder="Email" value="{{dictionary['EMAIL']}}"
required>
                    </div>
                    <div class="form-group">
                        <label for="password">Update Your Password</label>
                            <input type="text" name="password" class="form-control"
placeholder="Password"
                                value="{{dictionary['PASSWORD']}}" required>
                    </div>

                    <div class="form-group">
                        <label for="interest-select">Change Your Interest</label>
                        <div class="select">
                            <select name="interest" id="interest-select" required>
                                    <option value="{{dictionary['INTERESTS']}}">--NOT
SELECTED--</option>
                                        <option value="programming">Programming
Sector</option>
```

```
                                <option value="non-programming">Non-Programming
Sector</option>
                                                <option value="central-government">Central
Government Sector</option>
                                    <option value="state-government">State Government
Sector</option>
                                  <option value="banking">Banking Sector</option>
                                </select>
                              </div>
                            </div>

                            <div class="form-group">
                              <div class="row">
                                <div class="col-sm-6 col-sm-offset-3">
                                    <input type="submit" name="login-submit" id="login-
submit"
                                        class="form-control btn btn-login" value="UPDATE">
                                </div>
                              </div>
                            </div>

                          </form></div>
                        </div>
                      </div>
                    </div>
                  </div>
                </body>
</html>
```

data.py:

```
job = {
    'programming':[
        {
        'title':'Front-End Developer',
        'salary': '₹3L per year',
        'skills':['HTML', 'CSS', 'Javascript', 'React', 'Angular'],
        'link': 'https://www.naukri.com/front-end-developer-jobs'
        },
        {
        'title': 'Full Stack Developer',
        'salary': '₹6.5L per year',
        'skills':['Frontend Languages', 'Backend Technologies', 'DBMS'],
        'link': 'https://www.naukri.com/full-stack-developer-jobs'
        },
        {
        'title': 'Android Developer',
        'salary': '₹5.5L per year',
        'skills':['SQL', 'Java', 'Javascript', 'C++', 'PHP', 'Python'],
        'link': 'https://www.naukri.com/full-stack-developer-jobs'
        },
        {
        'title': 'Big Data Engineer',
        'salary': '₹7.8L per year',
        'skills':['Scripting Languages', 'Java', 'Python', 'MongoDB', 'redis'],
        'link': 'https://www.naukri.com/full-stack-developer-jobs'
        },
        {
        'title': 'AWS Developer',
```

'salary': '₹6.5L per year',
'skills':['Cloud Based Monitoring', 'Cloud Security tools', 'DBMS'],
'link': 'https://www.naukri.com/full-stack-developer-jobs'
},
{
'title': 'Robotics Engineer',
'salary': '₹5L per year',
'skills':['C', 'C++', 'Python', 'Java'],
'link': 'https://www.naukri.com/full-stack-developer-jobs'
},
{
'title': 'Software Engineer',
'salary': '₹6.5L per year',
'skills':['Java', 'Python', '.Net', 'Mean', 'Ruby'],
'link': 'https://www.naukri.com/full-stack-developer-jobs'
},
{
'title': 'Blockchain Developer',
'salary': '₹7.5L per year',
'skills':['Java', 'Python', 'C++', 'Data Structure', 'Backend Development','Cryptography'],
'link': 'https://www.naukri.com/full-stack-developer-jobs'
},
{
'title': 'Machine Learning Engineer',
'salary': '₹9L per year',
'skills':['Python', 'C++', 'Data modelling', 'A.I'],
'link': 'https://www.naukri.com/full-stack-developer-jobs'
},

```
        {
        'title': 'Data Scientists',
        'salary': '₹11L per year',
                'skills':['Machine learning', 'Deep learning', 'Data Visualization', 'Data Wrangling'],
        'link': 'https://www.naukri.com/full-stack-developer-jobs'
        },
    ],
    'non-programming': [
        {
        'title': 'Business Analyst',
        'salary': '₹6L per year',
        'skills':['MS Office', 'Interpersonal skills', 'Consultative skills'],
        'link': 'https://www.naukri.com/business-analyst-jobs'
        },
        {
        'title': 'UI/UX Designer',
        'salary': '₹5.1L per year',
         'skills':['Prototyping', ' wireframing', 'mockups', 'user flows','Industry knowledge', 'Focus on a specialty'],
        'link': 'https://www.naukri.com/business-analyst-jobs'
        },
        {
        'title': 'System Admin',
        'salary': '₹6.9L per year',
          'skills':['Technical Skills', 'Operating System', 'Organizational Skills','Attention to Detail','Networking'],
        'link': 'https://www.naukri.com/business-analyst-jobs'
        },
```

```
        {
        'title': 'Technical Writer',
        'salary': '₹6.5L per year',
                    'skills':['Clear understanding', 'User persona targeting', 'Technical documentation essentials', 'Basic graphic', ' web design', 'Research skills'],
        'link': 'https://www.naukri.com/business-analyst-jobs'
        },
        {
        'title': 'QA Manual Testing',
        'salary': '₹6.3L per year',
        'skills':['Attention to detail', 'Strong organizational skills'],
        'link': 'https://www.naukri.com/business-analyst-jobs'
        },
        {
        'title': 'Human Resource',
        'salary': '₹6L per year',
        'skills':['Employee relations', 'Project management', 'Advising', 'Coaching'],
        'link': 'https://www.naukri.com/business-analyst-jobs'
        },
        {
        'title': 'Devops Engineer',
        'salary': '₹7.5L per year',
        'skills':['Saas', 'Plateform tools(Chef, Puppet, Docker)'],
        'link': 'https://www.naukri.com/business-analyst-jobs'
        },
    ],
    'central-government' : [
        {
        'title': 'IAS',
```

'skills': ['UPSC'],
'salary': '₹15L per year',
'link': 'https://www.upsc.gov.in/'
},
{
'title': 'IPS',
'skills':['UPSC'],
'salary': '₹12L per year',
'link': 'https://www.upsc.gov.in/'
},
{
'title': 'IFOS',
'skills':['UPSC'],
'salary': '₹11L per year',
'link': 'https://www.upsc.gov.in/'
},
{
'title': 'Inspector',
'skills':['SSC'],
'salary': '₹9L per year',
'link' : 'https://ssc.nic.in/'
},
{
'title': 'Sub Inspector',
'skills':['SSC'],
'salary': '₹7.5L per year',
'link' : 'https://ssc.nic.in/'
},
{

'title': 'Inspector of Income Tax',
'skills':['SSC'],
'salary': '₹9L per year',
'link' : 'https://ssc.nic.in/'
},
{
'title': 'Auditor',
'skills':['SSC'],
'salary': '₹11L per year',
'link' : 'https://ssc.nic.in/'
},
{
'title': 'Station Master',
'skills':['RRB'],
'salary': '₹5.9L per year',
'link' : 'https://indianrailways.gov.in/'
},
{
'title': 'Senior Time Keeper',
'skills':['RRB'],
'salary': '₹4L per year',
'link' : 'https://indianrailways.gov.in/'
},
{
'title': 'Junior Account Assistant Cum Typist',
'skills':['RRB'],
'salary': '₹3L per year',
'link' : 'https://indianrailways.gov.in/'
},

```
    {
    'title': 'Goods Guard',
    'skills':['RRB'],
    'salary': '₹6L per year',
    'link' : 'https://indianrailways.gov.in/'
    },
],
'state-government' : [
    {
    'title': 'Deputy Collector',
    'skills':['TNPSC Group 1'],
    'salary': '₹12L per year',
    'link' : 'https://www.tnpsc.gov.in/'
    },
    {
    'title': 'Deputy Superintendent of Police',
    'skills':['TNPSC Group 1'],
    'salary': '₹10L per year',
    'link' : 'https://www.tnpsc.gov.in/'
    },
    {
    'title': 'Assistant Commissioner',
    'skills':['TNPSC Group 1'],
    'salary': '₹9L per year',
    'link' : 'https://www.tnpsc.gov.in/'
    },
    {
    'title': 'Deputy Registrar of Co-operative Societies',
    'skills':['TNPSC Group 1'],
```

'salary': '₹5L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{
'title': 'Assistant Director of Rural Development',
'skills':['TNPSC Group 1'],
'salary': '₹5L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{
'title': 'District Officer',
'skills':['TNPSC Group 1'],
'salary': '₹4L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{
'title': 'Industrial Co-operative Officer',
'skills':['TNPSC Group 2'],
'salary': '₹6L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{
'title': 'Municipal Commissioner',
'skills':['TNPSC Group 2'],
'salary': '₹4L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{
'title': 'Audit Inspector',

'skills':['TNPSC Group 2'],
'salary': '₹5L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{
'title': 'Assistant Inspector',
'skills':['TNPSC Group 2'],
'salary': '₹4L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{
'title': 'Station Fire Officer',
'skills':['TNPSC Group 3'],
'salary': '₹4L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{
'title': 'Village Administrative Officer',
'skills':['TNPSC Group 4'],
'salary': '₹4L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{
'title': 'Typist',
'skills':['TNPSC Group 4'],
'salary': '₹3L per year',
'link' : 'https://www.tnpsc.gov.in/'
},
{

```
    'title': 'Bill Collector',
    'skills':['TNPSC Group 4'],
    'salary': '₹3L per year',
    'link' : 'https://www.tnpsc.gov.in/'
    },
    {
    'title': 'Steno-Typist ',
    'skills':['TNPSC Group 4'],
    'salary': '₹2.5L per year',
    'link' : 'https://www.tnpsc.gov.in/'
    },
    {
    'title': 'Junior Assistant in Tamil Nadu Housing Board',
    'skills':['TNPSC Group 4'],
    'salary': '₹4L per year',
    'link' : 'https://www.tnpsc.gov.in/'
    },
    {
    'title': 'Junior Assistant',
    'skills':['TNPSC Group 4'],
    'salary': '₹4L per year',
    'link' : 'https://www.tnpsc.gov.in/'
    },
],
'banking' : [
    {
    'title': 'State Bank of India',
    'skills':['IBPS'],
    'salary': '₹4L per year',
```

```
        'link' : 'https://www.ibps.in/'
        },
        {
        'title': 'Reserve Bank of India',
        'skills':['IBPS'],
        'salary': '₹3.5L per year',
        'link' : 'https://www.ibps.in/'
        },
        {
        'title': 'Bank of India',
        'skills':['IBPS'],
        'salary': '₹3.6L per year',
        'link' : 'https://www.ibps.in/'
        },
        {
        'title': 'Indian Overseas Bank',
        'skills':['IBPS'],
        'salary': '₹3.3L per year',
        'link' : 'https://www.ibps.in/'
        },
        {
        'title': 'Corporation Bank',
        'skills':['IBPS'],
        'salary': '₹3.1L per year',
        'link' : 'https://www.ibps.in/'
        },
    ],
}
```

GITHUB & PROJECT DEMO LINK

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-45353-1660729600

YouTube Link: https://www.youtube.com/embed/fpI4yTnj5-s