

SKILL AND JOB RECOMMENDER

A PROJECT REPORT

Submitted by

PRAIMATHI.K (813819104064)

PRIYADHARSHINI.N.K (813819104069)

SAHANA PARVEEN.M (813819104080)

SHRI VARDHINI.M (813819104094)

CONTENTS

1. INTRODUCTION

1. Project Overview
2. Purpose

2. LITERATURE SURVEY

1. Existing problem
2. References
3. Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

1. Empathy Map Canvas
2. Ideation & Brainstorming
3. Proposed Solution
4. Problem Solution fit

4. REQUIREMENT ANALYSIS

1. Functional requirement
2. Non-Functional requirements

5. PROJECT DESIGN

1. Data Flow Diagrams
2. Solution & Technical Architecture
3. User Stories

6. PROJECT PLANNING & SCHEDULING

1. Sprint Planning & Estimation
2. Sprint Delivery Schedule
3. Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

1. Feature 1
2. Feature 2
3. Database Schema (if Applicable)

8. TESTING

1. Test Cases
2. User Acceptance Testing

9. RESULTS

1. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1.INTRODUCTION

1.1 Project Overview

Finding jobs that best suits the interests and skill set is quite a challenging task for the job seekers. The difficulties arise from not having proper knowledge on the organization's objective, their work culture and current job openings. In addition, finding the right candidate with desired qualifications to fill their current job openings is an important task for the recruiters of any organization. Job recommendation system has certainly made job seeking convenient to job seekers. This is the solution where recruiter as well as the job seeker meets aiming at fulfilling their individual requirement. To develop end-to-end web users are the cheapest as well as the fastest source of communication reaching wide range of audience on just a single click irrespective of their geographical distance. As simplified recruitment process which makes way convenience for job seekers to access job portal on the go, as our app script is built to support numerous business models as per the industry requirements. In this decade people are using their smart phone rather than web portals for job seeking online. So it is the right way for career portal which balances the gap between recruitment board and job searcher candidates. On the job seeker panel, the user can register their profile using system. Here, a job seeker searches for the positions that interest him and submits an application. Due to the abundance of job boards, candidates typically choose the one that offers the best services to them, including skills, creating a job profile, and suggesting new positions to job seekers.

1.2 Purpose

Job recommendation is primarily aimed at supporting the discovery of jobs that may interest the user. It should be dynamic in order to cater to the changing preferences of the user. The proposed system will help the user to overcome these difficulties by matching their skills and other details with appropriate companies suitable for respective user. The proposed system consists of user dataset with various attributes and company dataset with company details

2.LITERATURE REVIEW

2.1 Existing problem

2.1.1 TITLE OF THE PAPER : Job Recommendation based on Job Seeker Skills: An Empirical Study.

YEAR OF PUBLICATION: 2018

AUTHORS OF THE PAPER:

- Jorge Valverde-Rebaza
- Ricardo Puma
- Paul Bustios
- Nathalia C. Silva

THEME OF THE PAPER: Job recommender systems have become popular since they successfully reduce information overload by generating personalized job suggestions. Although in the literature exists a variety of techniques and strategies used as part of job recommender systems, most of them fail to recommending job vacancies that properly to the job seekers profiles. Thus, the contributions of this work are threefold, we:

- i) made publicly available a new dataset formed by a set of job seekers profiles and a set of job vacancies collected from different job search engine sites
 - ii) put forward the proposal of a framework for job recommendation based on professional skills of job seekers
 - iii) carried out an evaluation to quantify empirically the recommendation abilities of two state-of-the-art methods, considering different configurations, within the proposed framework. We thus present a general panorama of job recommendation task aiming to facilitate research and real-world application design regarding this important issue.
- Keywords: Job matching, job seeking, job search, job recommender Systems, word embedding.

2.1.2 TITLE OF THE PAPER : Implicit Skills Extraction Using Document Embedding and Its Use in Job Recommendation

YEAR OF PUBLICATION : 2014

AUTHORS OF THE PAPER :

- Akshay Gugnani
- Hemant Misra

THEME OF THE PAPER : Formal job search and application typically involves matching one's profile or curriculum vitae (CV) with the available job descriptions (JD), and then applying for those job opportunities whose JDs are the closest match to one's CV, and also considering his/her needs, constraints, and aspirations. The fundamental premise this paper builds upon is that skills are one of the most important aspects while matching CVs to JDs, and play a major role in recommending JDs which are the best match for a CV. The following are the important contributions of this paper : • A new methodology which combines several natural language processing (NLP) techniques for robust skill extraction from natural occurring texts in CVs and JDs is proposed • An approach for inferring implicit skills in a JD (skills not explicitly mentioned in the JD) is introduced, and a method to extract these implicit skills from other similar JDs is presented • A bi-directional matching algorithm to match skills between CVs and JDs is suggested to obtain the most relevant job recommendation for each CV.

2.2 REFERENCES

1. F. M. Javed Mehedi Shamrat, Implementation of an Intelligent Online Job Portal Using Machine Learning Algorithms, 2020.
2. Zamiwe Tembo, Designing And Implementation Of A Graduate Job Portal System, 2019.
3. Ankit Bhatnagar¹ ,Nitish Kajla² , Mahesh Kumar Gupta³,Recruitment And Selection Process With Reference Using Job Portal Framework, 2021.
4. Aradhana Patra,Munjarin Rahman,Shared Values of E-Recruitment Portal: Determinant Factors of Job-Seekers' Intention to use Job Portals, 2020.
5. Gökçe Karaoglu, Eszter Hargittai & Minh Hao Nguyen,Inequality in online job searching in the age of social media, 2021.

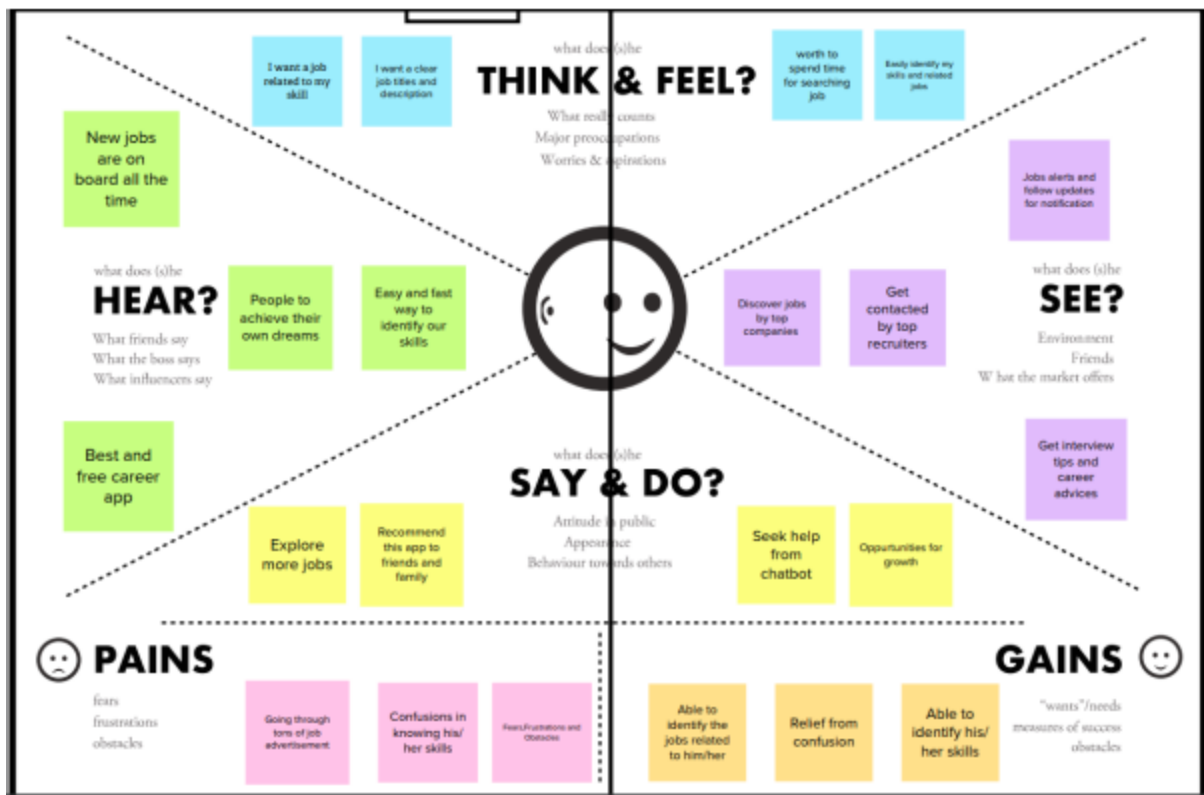
2.3 PROBLEM STATEMENT DEFINITION

The existing system is handled manually. The system follows large number of paper work for maintaining job details and user can be difficult to search the part time jobs in manual process. In current system the student or user don't know about part time jobs details or company/office details and location. In this existing system takes lots of time for searching particular jobs information.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

3.3 Proposed Solution

The proposed system is developed after a detailed study about the requirements requested by the user. Proposed system is a computerized one, where all the limitations of manual system are compensated. jobs details of web application for skill based Job application system have simplified the working information and make a user friendly environment, where the user is provided with much flexibility to manage effectively. It helps the admin to generate desirable interface more quickly and also to produce better results.

3.4 Problem Solution fit

Dealing with the enormous amount of recruiting information on the Internet, a job seeker always spends hours to find useful ones. Many times, people who lack industry knowledge are unclear about what exactly they need to learn in order to get a suitable job for them. We address the problem of recommending suitable jobs to people who are seeking a new job.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

- **Create interface** : This module offered a framework for job platform application to the user, to get answers without any human assistance. Admin can train keywords with answers for future processing. Chatbots are such kind of computer programs that interact with users using natural languages.
- **Registration** : There is registration form available where new user can create their account by providing required information to the system. The registration form details are like name, email, gender, mobile number, address, and etc. These details are stored in the database. And then can getting to the username and password in the system. After the login process, the user can login in the system using username and password.
- **Update job details** : The company can register to this application, the registered details like company name, id, email address; mobile number etc. after the registration process, the company can update the job details.
- **Update skills** : The user can upload the skill details to this application. And the user will interact with the Chabot and can get the recommendations based on their skills.
- **Recommend job with alert** :After updating the skills details, the system will recommend the job openings based on the user skills.
- **Apply job** : After get the job alert, the user can apply the job through this application.

4.2 NON FUNCTIONAL REQUIREMENTS

Usability

The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy

Availability

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

Scalability

Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

Security

A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied

Performance

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the requestsubmittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 secondsto appear on the screen.

Reliability

The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week. 24 hours a day.

5. PROJECT DESIGN

5.1 Data Flow Diagrams

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

LEVEL 0

The Level 0 DFD shows how the system is divided into 'sub-systems' (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

LEVEL 1

The next stage is to create the Level 1 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, to describe the system was using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.

LEVEL 2

A Data Flow Diagram (DFD) tracks processes and their data paths within the business or system boundary under investigation. A DFD defines each domain boundary and illustrates the logical movement and transformation of data within the defined boundary. The diagram shows 'what' input data enters the domain, 'what' logical processes the domain applies to that data, and 'what' output data leaves the domain. Essentially, a DFD is a tool for process modelling and one of the oldest.

5.2 Solution & Technical Architecture

5.3 User Stories

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day).

$$\text{AV} = \text{Sprint duration} / \text{Velocity} = 20 / 10 = 2$$

Average Velocity for Sprint 1 $\Rightarrow 20 / 6 = 3.3$

Average Velocity for Sprint 2 $\Rightarrow 20 / 6 = 3.3$

Average Velocity for Sprint 3 $\Rightarrow 20 / 6 = 3.3$

Average Velocity for Sprint 4 $\Rightarrow 20 / 6 = 3.3$

6.3 Reports from JIRA

7. CODING & SOLUTIONING

7.1 Feature 1

sendgrid notification :

```
#send grid

def sendmsg(Mailid,message):
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    from email.mime.base import MIMEBase
    from email import encoders

    fromaddr = "sampletest685@gmail.com"
    toaddr = Mailid

    # instance of MIMEMultipart
    msg = MIMEMultipart()

    # storing the senders email address
    msg['From'] = fromaddr

    # storing the receivers email address
    msg['To'] = toaddr

    # storing the subject
    msg['Subject'] = "Alert"

    # string to store the body of the mail
    body = message

    # attach the body with the msg instance
    msg.attach(MIMEText(body, 'plain'))

    # creates SMTP session
    s = smtplib.SMTP('smtp.gmail.com', 587)

    # start TLS for security
    s.starttls()

    # Authentication
    s.login(fromaddr, "hneucvnontsuwgpj")

    # Converts the Multipart msg into a string
    text = msg.as_string()
```

```
# sending the mail
s.sendmail(fromaddr, toaddr, text)

# terminating the session

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

7.2 Feature 2

```
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "ba94e962-028d-4687-be00-33151ab90d90", // The ID of this integration.
    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceID: "a5732b76-b5f7-4073-bf52-222d580f1669", // 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>
<script src="static/js/responsiveslides.min.js"></script>
<script>
  // You can also use "$(window).load(function() {"
  $(function () {
    // Slideshow 3
    $("#slider3").responsiveSlides({
      auto:false,
      pager: true,
      nav: false,
      speed: 500,
      namespace: "callbacks",
      before: function () {
        $('.events').append("<li>before event fired.</li>");
      },
      after: function () {
        $('.events').append("<li>after event fired.</li>");
      }
    });

  });
</script>
```

7.3 Database Schema

A table is a data structure that organizes information into rows and columns. It can be used to both store and display data in a structured format. For example, databases store data in tables so that information can be quickly accessed from specific rows. Websites often use tables to display multiple rows of data on page. Spreadsheets combine both purposes of a table by storing and displaying data in a structured format.

Databases often contain multiple tables, with each one designed for a specific purpose. For example, a company database may contain separate tables for employees, clients, and suppliers. Each table may include its own set of fields, based on what data the table needs to store. In database tables, each field is considered a column, while each entry (or record), is considered a row. A specific value can be accessed from the table by requesting data from an individual column and row.

Company table

Field	Type
companyname	nvarchar(50)
regno	nvarchar(50)
mobile	nvarchar(50)
email	nvarchar(50)
website	nvarchar(50)
address	nvarchar(50)
username	nvarchar(50)
password	nvarchar(50)

Job table

Field	Type
Company name	nvarchar(50)
Contact no	nvarchar(50)
address	nvarchar(50)
location	nvarchar(50)
vacancy	nvarchar(50)
job	nvarchar(50)
department	nvarchar(50)
Website	nvarchar(50)
cname	nvarchar(50)

Registration table

Field	Type
Name	nvarchar(50)
gender	nvarchar(50)
age	nvarchar(50)
Email	nvarchar(50)
Phone	nvarchar(50)
Address	nvarchar(50)
degree	nvarchar(50)
department	nvarchar(50)
Username	nvarchar(50)
password	nvarchar(50)

8. TESTING

8.1 TEST CASES

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on “HOW” to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary.

S. N O	Scenario	Input	Excepted output	Actual output
1	Admin Login Form	User name and password	Login	Login success.
2	User Registration Page	User Basic Details	Registered successfully	User registration details are stored in database.
3	User Login Form	User name and password	Login	Login success.
4	Update Skills Details	Skills Details	Updated successfully	Skills details are stored in database.

8.2 USER ACCEPTANCE TESTING

This is a type of testing done by users, customers, or other authorised entities to determine application/software needs and business processes. Acceptance testing is the most important phase of testing as this decides whether the client approves the application/software or not. It may involve functionality, usability, performance, and U.I of the application. It is also known as user acceptance testing (UAT), operational acceptance testing (OAT), and end-user testing.

Acceptance Testing UAT Execution & Report Submission

Date	03 November 2022
Team ID	PNT2022TMD32880
Project Name	SKILL and JOB RECOMMENDER
Maximum Marks	4 Marks

1. Purpose of Document

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

2. Defect Analysis

Outsource Shipping	2	0	0	2	
This report shows the number of resolved or closed bugs at each severity level, and how they were resolved					
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	8	3	4	5	20
Duplicate	2	0	2	0	4
External	1	0	4	0	5
Fixed	13	3	1	17	34
Not Reproduced	0	1	1	0	2
Skipped	2	0	1	1	4
Won't Fix	0	5	2	3	10
Totals	26	12	15	26	79

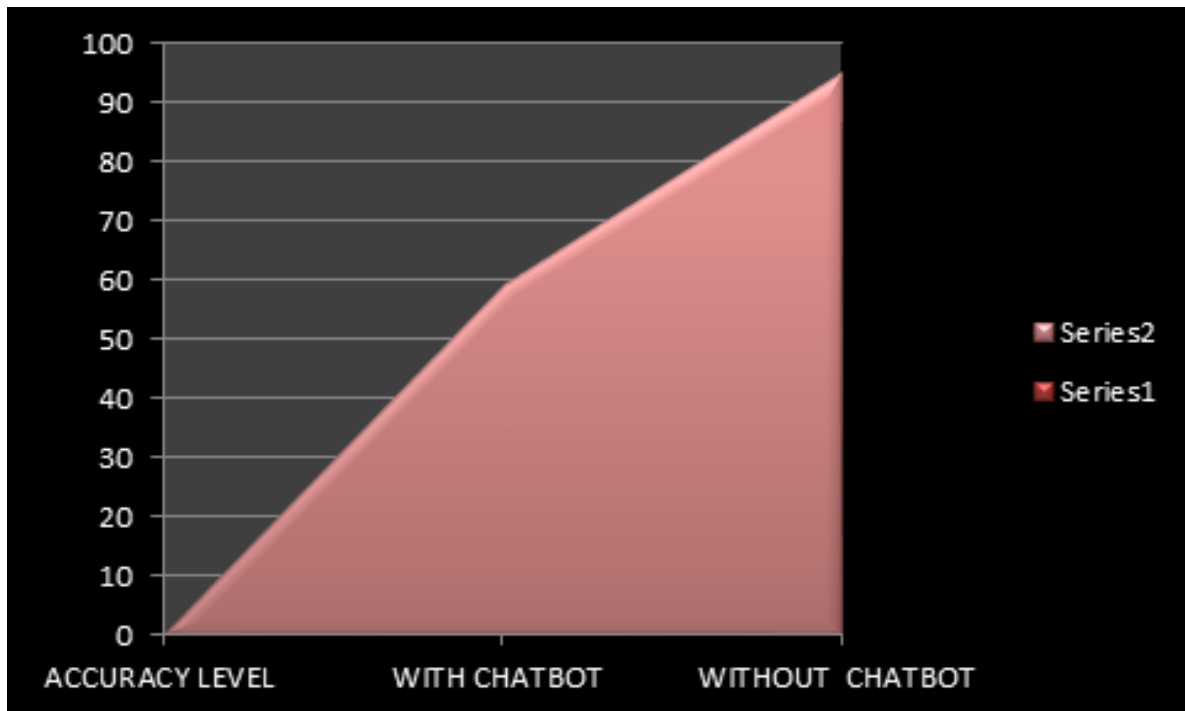
2. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested				
Exception Reporting	9	0	0	9
Final Report Output	3	0	0	3
Version Control	5	0	0	5

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	72	0	0	72
Security	6	0	0	6

9. RESULTS

9.1 PERFORMANCE METRICS



10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. User can easily know about the company details.
2. Automation of existing manual information systems.
3. Reduction of manual processing
4. Users will interact with the Chabot and can get the recommendations based on their skills.
5. Keep track of daily information exchange at the server by the administrator.
6. Increase in processing and transfer speeds of information over the network.
7. Decrease in processing time

DISADVANTAGES

1. Poor communication between user and company officer, so here intimating about new job is a hard task.
2. Know the company job vacancy information is very difficult
3. Immediate response to the queries is difficult.
4. More stationary use so they are expensive.
5. Manual system is takes more time.
6. Existing system is manually, so it increases the chances of errors.

11.CONCLUSION

In this essay, we suggested a structure for the duty of job recommendations. The use of a variety of text processing and recommendation methods in accordance with the preferences of the job recommender system creator is permitted by this framework, which also makes it easier to comprehend the job suggestion process. Furthermore, we make a new dataset with profiles of job seekers and open positions publicly accessible. The coding is done in a simplified and easy to understandable manner so that other team trying to enhance the project can do so without facing much difficulty. The documentation will also assist in the process as it has also been carried out in a simplified and concise way.

12. FUTURE SCOPE

In future we can develop this project in android application. This system is developed such a way that additional enhancement can be done without much difficulty. The renovation of the project would increase the flexibility of the system. Also the features are provided in such a way that the system can also be made better and efficient functionality

- Try to all user contact with online.
- Add more features in site future.

13. APPENDIX

SOURCE CODE

App.py

```
from flask import Flask, render_template, flash, request, session
from flask import render_template, redirect, url_for, request
import json
from json2html import *
import requests
import ibm_db
import pandas
import ibm_db_dbi
from sqlalchemy import create_engine
engine = create_engine('sqlite://', echo = False)
dsn_hostname="98538591-7217-4024-b027-
8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud"
dsn_uid = "tvd24047"
dsn_pwd = "C0fhAXeLsuoQuvel"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"
dsn_port = "30875"
dsn_protocol = "TCPIP"
dsn_security = "SSL"
dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
```



```

    "PWD={6};"

    "SECURITY={7};").format(dsn_driver, dsn_database, dsn_hostname, dsn_port,
        dsn_protocol, dsn_uid, dsn_pwd,dsn_security)
try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ",
        dsn_hostname)
except:
    print ("Unable to connect: ", ibm_db.conn_errormsg() )
app = Flask(__name__)
app.config['DEBUG']
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'
@app.route("/")
defhomepage():
    return render_template('index.html')

@app.route("/Home")
defHome():
    return render_template('index.html')

@app.route("/AdminLogin")
defAdminLogin():
    return render_template('AdminLogin.html')

@app.route("/NewUser")
defNewUser():
    return render_template('NewUser.html')

@app.route("/NewCompany")

```

```

defNewCompany():
    return render_template('NewCompany.html')

@app.route("/UserLogin")
defStudentLogin():
    return render_template('UserLogin.html')

@app.route("/CompanyLogin")
defCompanyLogin():
    return render_template('CompanyLogin.html')

@app.route("/Search")
defSearch():
    return render_template('Search.html')

@app.route("/AdminHome")
@app.route("/AdminHome")
defAdminHome():
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * from regtb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()
    return render_template('AdminHome.html', data=data)

@app.route("/ACompanyInfo")
defACompanyInfo():

    conn = ibm_db.connect(dsn, "", "")

```

```
pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * from companytb "
dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM Employee_Data").fetchall()
return render_template('ACompanyInfo.html', data=data)
```

```
@app.route("/AjobInfo")
defAjobInfo():
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * from jobtb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()

    return render_template('AjobInfo.html', data=data)
```

```
@app.route("/SCompanyInfo")
defSCompanyInfo():

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * from jobtb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
```

```
data = engine.execute("SELECT * FROM Employee_Data").fetchall()
```

```
return render_template('SCompanyInfo.html', data=data)
```

```
@app.route("/CompanyHome")
```

```
defCompanyHome():
```

```
    return render_template('CompanyHome.html')
```

```
@app.route("/UserHome")
```

```
defUserHome():
```

```
    uname= session['uname']
```

```
    conn = ibm_db.connect(dsn, "", "")
```

```
    pd_conn = ibm_db_dbi.Connection(conn)
```

```
    selectQuery = "SELECT * FROM regtb where Username='"+ uname +"' "
```

```
    dataframe = pandas.read_sql(selectQuery, pd_conn)
```

```
    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
```

```
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()
```

```
    return render_template('UserHome.html', data=data)
```

```
@app.route("/CJobInfo")
```

```
defCJobInfo():
```

```
    cname= session['cname']
```

```
    conn = ibm_db.connect(dsn, "", "")
```

```

pd_conn = ibm_db_dbi.Connection(conn)
selectQuery = "SELECT * FROM jobtb where Cname='"+ cname +"' "
dataframe = pandas.read_sql(selectQuery, pd_conn)
dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
data = engine.execute("SELECT * FROM Employee_Data").fetchall()

return render_template('CJobInfo.html', data=data)

@app.route("/adminlogin", methods=['GET', 'POST'])
defadminlogin():
    error = None
    if request.method == 'POST':
        if request.form['uname'] == 'admin'or request.form['password'] == 'admin':
            conn = ibm_db.connect(dsn, "", "")
            pd_conn = ibm_db_dbi.Connection(conn)
            selectQuery = "SELECT * FROM regtb "
            dataframe = pandas.read_sql(selectQuery, pd_conn)
            dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
            data = engine.execute("SELECT * FROM Employee_Data").fetchall()

            return render_template('AdminHome.html', data=data)

        else:
            return render_template('index.html', error=error)
@app.route("/userlogin", methods=['GET', 'POST'])
defuserlogin():
    error = None
    if request.method == 'POST':

```

```

username = request.form['uname']
password = request.form['password']

conn = ibm_db.connect(dsn, "", "")

pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * from regtb where UserName='" + username + "' and
password='" + password + "'"
dataframe = pandas.read_sql(selectQuery, pd_conn)

if dataframe.empty:
    data1 = 'Username or Password is wrong'
    return render_template('goback.html', data=data1)
else:
    print("Login")
    selectQuery = "SELECT * from regtb where UserName='" + username + "' and
password='" + password + "'"
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('Employee_Data',
                    con=engine,
                    if_exists='append')

# run a sql query
print(engine.execute("SELECT * FROM Employee_Data").fetchall())

    return render_template('UserHome.html', data=engine.execute("SELECT * FROM
Employee_Data").fetchall())

```

```

@app.route("/companylogin", methods=['GET', 'POST'])
def companylogin():
    error = None
    if request.method == 'POST':

        uname = request.form['uname']
        password = request.form['password']
        session['cname'] = uname

        conn = ibm_db.connect(dsn, "", "")

        pd_conn = ibm_db_dbi.Connection(conn)

        selectQuery = "SELECT * from companytb where UserName='" + uname + "' and
password='" + password + "'"
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        if dataframe.empty:
            data1 = 'Username or Password is wrong'
            return render_template('goback.html', data=data1)
        else:
            print("Login")
            selectQuery = "SELECT * from companytb where UserName='" + uname + "' and
password='" + password + "'"
            dataframe = pandas.read_sql(selectQuery, pd_conn)

            dataframe.to_sql('Employee_Data',
                            con=engine,
                            if_exists='append')

```

```

# run a sql query
print(engine.execute("SELECT * FROM Employee_Data").fetchall())

return render_template('CompanyHome.html', data=engine.execute("SELECT *
FROM Employee_Data").fetchall())

@app.route("/NewStudent1", methods=['GET', 'POST'])
defNewStudent1():
    if request.method == 'POST':

        name = request.form['name']
        gender = request.form['gender']
        Age = request.form['Age']
        email = request.form['email']
        pnumber = request.form['pnumber']
        address = request.form['address']
        Degree = request.form['Degree']
        depat = request.form['depat']
        uname = request.form['uname']
        passw = request.form['passw']

    conn = ibm_db.connect(dsn, "", "")

    insertQuery = "insert into regtb values('" + name + "','" + gender + "','" + Age + "','" +
email + "','" + pnumber + "','" + address + "','" + Degree + "','" + depat + "','" + uname +
','" + passw + "')"
    insert_table = ibm_db.exec_immediate(conn, insertQuery)

```



```

sendmsg(email, "Successfully registered this website")

data1 = 'Record Saved!'
return render_template('goback.html', data=data1)
@app.route("/newcompany", methods=['GET', 'POST'])
defnewcompany():
    if request.method == 'POST':

        cname = request.form['cname']
        regno = request.form['regno']
        mobile = request.form['mobile']

        email = request.form['email']
        Website = request.form['Website']
        address = request.form['address']
        uname = request.form['uname']
        passw = request.form['passw']

        conn = ibm_db.connect(dsn, "", "")
        insertQuery = "insert into companytb
values('"+cname+"','"+regno+"','"+mobile+"','"+email+"','"+Website+"','"+address+"','"+u
name+"','"+passw+"')"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)

        data1 = 'Record Saved!'
        return render_template('goback.html', data=data1)
@app.route("/newjob", methods=['GET', 'POST'])
defnewjob():
    if request.method == 'POST':

```

```
cnn = session['cname']
cname = request.form['cname']
cno = request.form['cno']
Address = request.form['Address']
JobLocation = request.form['JobLocation']
Vacancy = request.form['Vacancy']
Job = request.form['Job']
Department = request.form['depat']
website = request.form['website']
```

```
conn = ibm_db.connect(dsn, "", "")
```

```
insertQuery = "insert into jobtb values('" + cname + "','" + cno + "','" + Address + "','" +
JobLocation + "','" + Vacancy + "','" + Job + "','" + Department + "','" + website +
','+"cnn+"")"
```

```
insert_table = ibm_db.exec_immediate(conn, insertQuery)
```

```
conn = ibm_db.connect(dsn, "", "")
```

```
pd_conn = ibm_db_dbi.Connection(conn)
```

```
selectQuery1 = "SELECT * FROM regtb where Department='" + Department + "'"
```

```
dataframe = pandas.read_sql(selectQuery1, pd_conn)
```

```
dataframe.to_sql('regtb', con=engine, if_exists='append')
```

```
data1 = engine.execute("SELECT * FROM regtb").fetchall()
```

```
for item1 in data1:
```

```
    Mobile = item1[5]
```

```
    Email = item1[4]
```

```
    sendmsg(Email, "Jop Title"+Job + " More Info Visit Website")
```

```

data = 'Record Saved!'

return render_template("goback.html", data=data)

@app.route("/jobsearch", methods=['GET', 'POST'])
def jobsearch():
    if request.method == 'POST':
        jobname = request.form['name']

        url = "https://linkedin-jobs-search.p.rapidapi.com/"

        payload = {
            "search_terms": jobname,
            "location": "india",
            "page": "1"
        }
        headers = {
            "content-type": "application/json",
            "X-RapidAPI-Key": "b045b9af95msha8d7c3160785729p1674cdjsnbdf4adbf9868",
            "X-RapidAPI-Host": "linkedin-jobs-search.p.rapidapi.com"
        }

        response = requests.request("POST", url, json=payload, headers=headers)
        print(response.text)
        infoFromJson = json.loads(response.text)
        df = pandas.json_normalize(infoFromJson)
        df.to_sql('regtb', con=engine, if_exists='append')

```

```
        data1 = engine.execute("SELECT * FROM regtb").fetchall()
    return render_template('Search.html',data=data1)
#send grid
defsendmsg(Mailid,message):
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    from email.mime.base import MIMEBase
    from email import encoders
    fromaddr = "sampletest685@gmail.com"
    toaddr = Mailid
# instance of MIMEMultipart
    msg = MIMEMultipart()

    # storing the senders email address
    msg['From'] = fromaddr

    # storing the receivers email address
    msg['To'] = toaddr

    # storing the subject
    msg['Subject'] = "Alert"

    # string to store the body of the mail
    body = message

    # attach the body with the msg instance
    msg.attach(MIMEText(body, 'plain'))
```

```

# creates SMTP session
s = smtplib.SMTP('smtp.gmail.com', 587)

# start TLS for security
s.starttls()

# Authentication
s.login(fromaddr, "hneucvnontsuwgpj")

# Converts the Multipart msg into a string
text = msg.as_string()

# sending the mail
s.sendmail(fromaddr, toaddr, text)

# terminating the session

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug='TRUE')

```

job.py

```

import requests
import json
import pandas as pd
from json2html import *
url = "https://linkedin-jobs-search.p.rapidapi.com/"

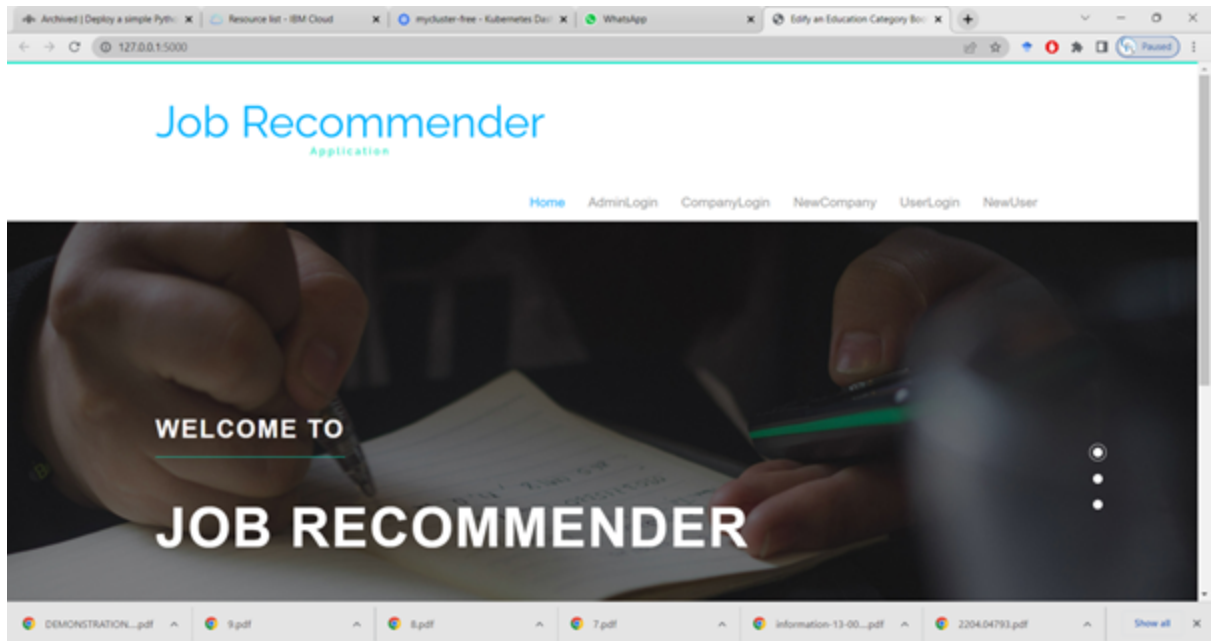
payload = {
    "search_terms": "python programmer",
    "location": "india",
    "page": "1"

```

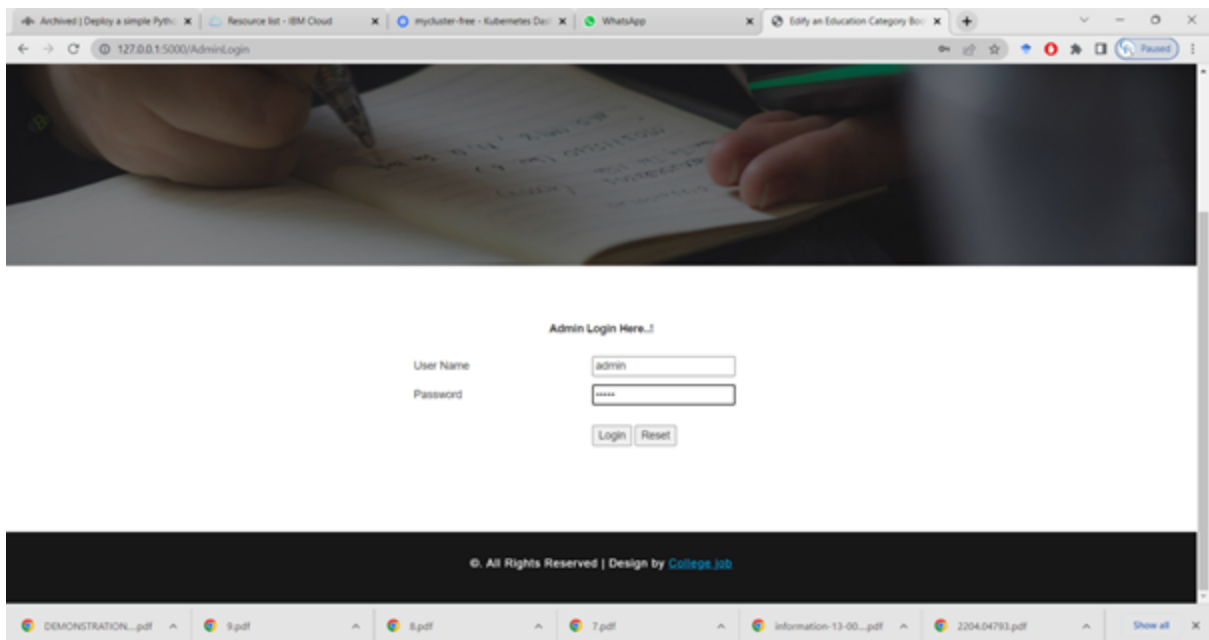
```
}  
headers = {  
    "content-type": "application/json",  
    "X-RapidAPI-Key": "b045b9af95msha8d7c3160785729p1674cdjsnbdf4adbf9868",  
    "X-RapidAPI-Host": "linkedin-jobs-search.p.rapidapi.com"  
}  
response = requests.request("POST", url, json=payload, headers=headers)  
print(response.text)  
  
infoFromJson = json.loads(response.text)  
print(json2html.convert(json = infoFromJson))  
#data = json.loads(elevations)  
df = pd.json_normalize(infoFromJson)  
print(df)
```

SCREENSHOTS

HOMEPAGE




ADMIN LOGIN PAGE



JOB INFORMATION

Archived | Deploy a simple | Service Details - IBM Cloud | IBM Db2 on Cloud | mycluster-free - Kubernetes | WhatsApp | Edit an Education Category

127.0.0.1:5000/CompanyInfo



Jop Info

CompanyName	ContactNo	Address	Location	Vacancy	Job	Department	website
vikram	9486365553	no 7 trichy	Trichy	20	python dev	CSE	www.ranjith.com
java	9874563210	trichy	trichy	1	developer	CSE	www.fantasy.com

DEMONSTRATION...pdf 9.pdf 8.pdf 7.pdf information-13-00...pdf 2204.04793.pdf Show all

Archived | Deploy a simple | Service Details - IBM Cloud | IBM Db2 on Cloud | mycluster-free - Kubernetes | WhatsApp | Edit an Education Category

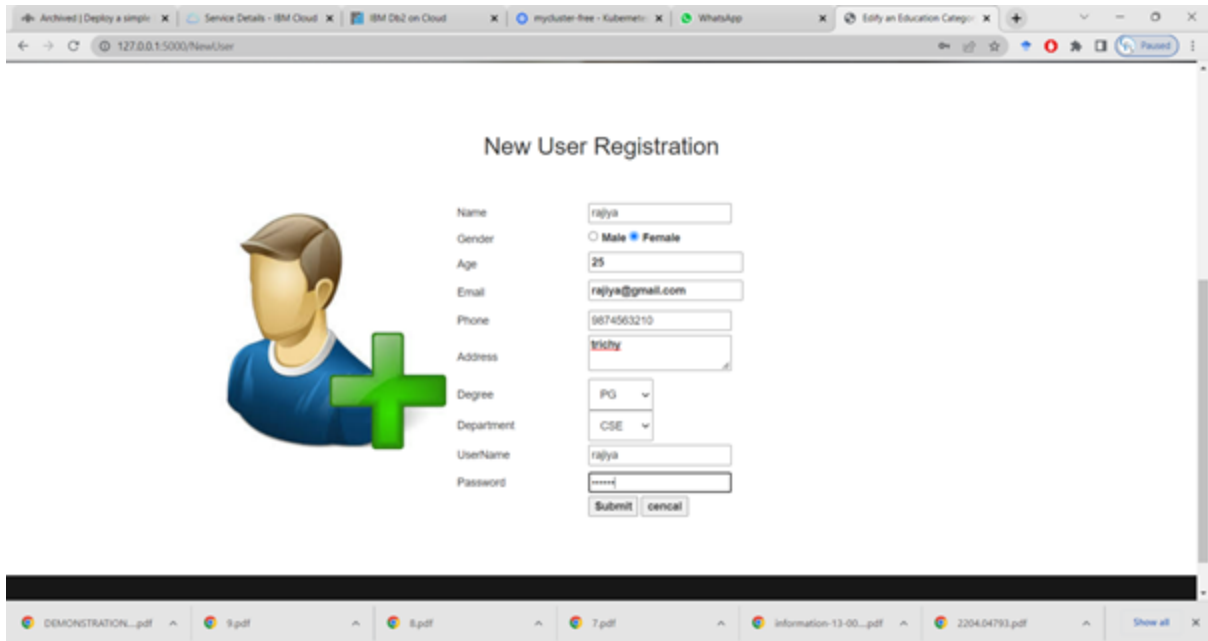
127.0.0.1:5000/jobsearch

JobName Search

job_url	company_name	company_url	job_title	job_location	posted_date
https://in.linkedin.com/jobs/view/java-application-developer-at-kpmg-india-3353857797	KPMG India	https://in.linkedin.com/company/kpmgindia	Java Application Developer	Bengaluru, Karnataka, India	2022-11-10
https://in.linkedin.com/jobs/view/java-developer-at-infosys-3363599845	Infosys	https://in.linkedin.com/company/infosys	Java Developer	India	2022-11-17
https://in.linkedin.com/jobs/view/java-at-innovation-groups-3357093084	Innovation Groups	https://ph.linkedin.com/company/innovation-groups	JAVA	New Delhi, Delhi, India	2022-11-17
https://in.linkedin.com/jobs/view/java-software-engineer-at-jio-3344044853	Jio	https://in.linkedin.com/company/jio	Java Software Engineer	Mumbai, Maharashtra, India	2022-11-09
https://in.linkedin.com/jobs/view/java-developer-at-infosys-3363622627	Infosys	https://in.linkedin.com/company/infosys	Java Developer	India	2022-11-17
https://in.linkedin.com/jobs/view/java-developer-with-rcp-ejb-at-mercedes-benz-research-and-development-india-3348032044	Mercedes-Benz Research and Development India	https://in.linkedin.com/company/mercedes-benz-research-and-development-india	Java Developer with RCP/EJB	Bengaluru, Karnataka, India	2022-11-11
https://in.linkedin.com/jobs/view/java-software-engineer-at-tech-mahindra-3357546503	Tech Mahindra	https://in.linkedin.com/company/tech-mahindra	Java Software Engineer	Chennai, Tamil Nadu, India	2022-11-11
			Senior Java	Bangalore	

DEMONSTRATION...pdf 9.pdf 8.pdf 7.pdf information-13-00...pdf 2204.04793.pdf Show all

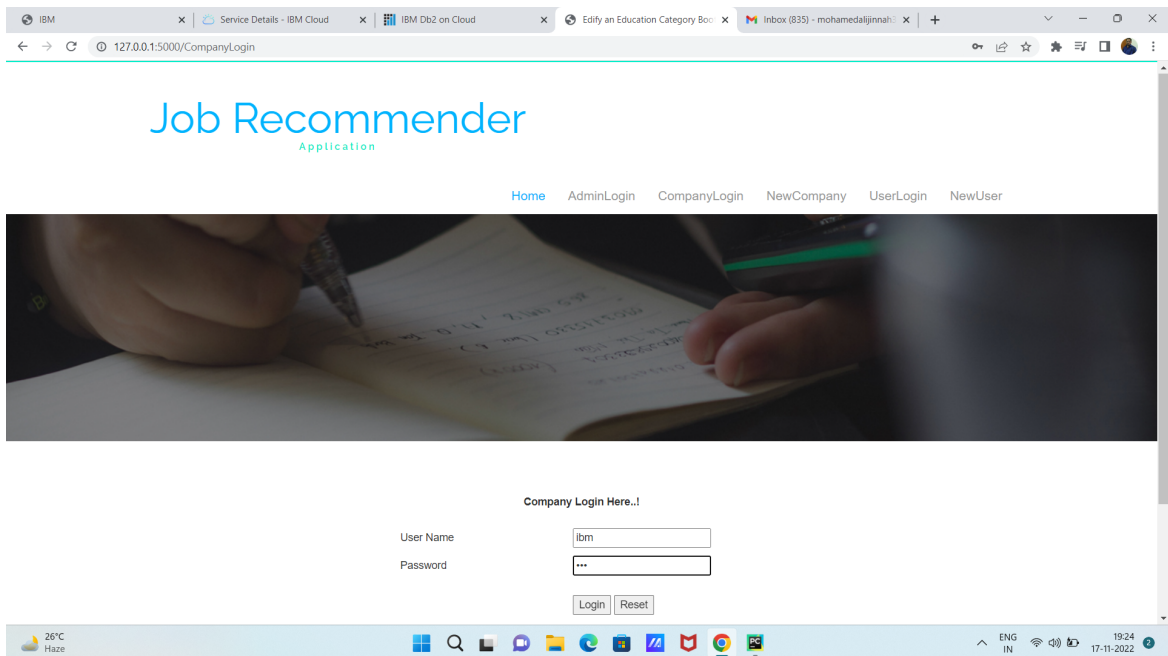
USER REGISTER PAGE



New User Registration

☐ Male ☒ Female

USER LOGIN PAGE



Job Recommender
Application


[Home](#) [AdminLogin](#) [CompanyLogin](#) [NewCompany](#) [UserLogin](#) [NewUser](#)

Company Login Here..!

NEW COMPANY REGISTER PAGE

Archived | Deploy a simple Pyth... | mycluster-free - IBM Cloud | mycluster-free - Kubernetes De... | WhatsApp | Edit an Education Category Bo... | 127.0.0.1:5000/NewCompany

New Company Details



CompanyName	<input type="text" value="fantasy"/>
RegisterNo	<input type="text" value="520535"/>
Mobile	<input type="text" value="9874563210"/>
Email	<input type="text" value="fantasy@gmail.com"/>
Website	<input type="text" value="www.fantasy.com"/>
Address	<input type="text" value="Kichy"/>
Username	<input type="text" value="fantasy"/>
Password	<input type="password" value="*****"/>

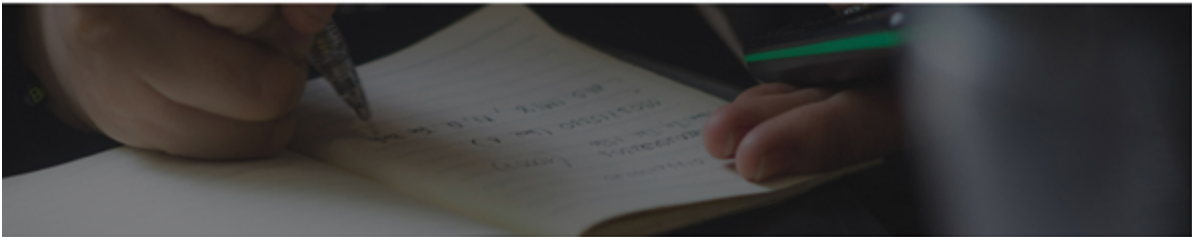
DEMONSTRATION...pdf 9.pdf 8.pdf 7.pdf Information-13-00...pdf 2204.04793.pdf Show all

COMPANY LOGIN PAGE

Archived | Deploy a simple Pyth... | mycluster-free - IBM Cloud | mycluster-free - Kubernetes De... | WhatsApp | Edit an Education Category Bo... | 127.0.0.1:5000/CompanyLogin

Application

[Home](#) [AdminLogin](#) [CompanyLogin](#) [NewCompany](#) [UserLogin](#) [NewUser](#)

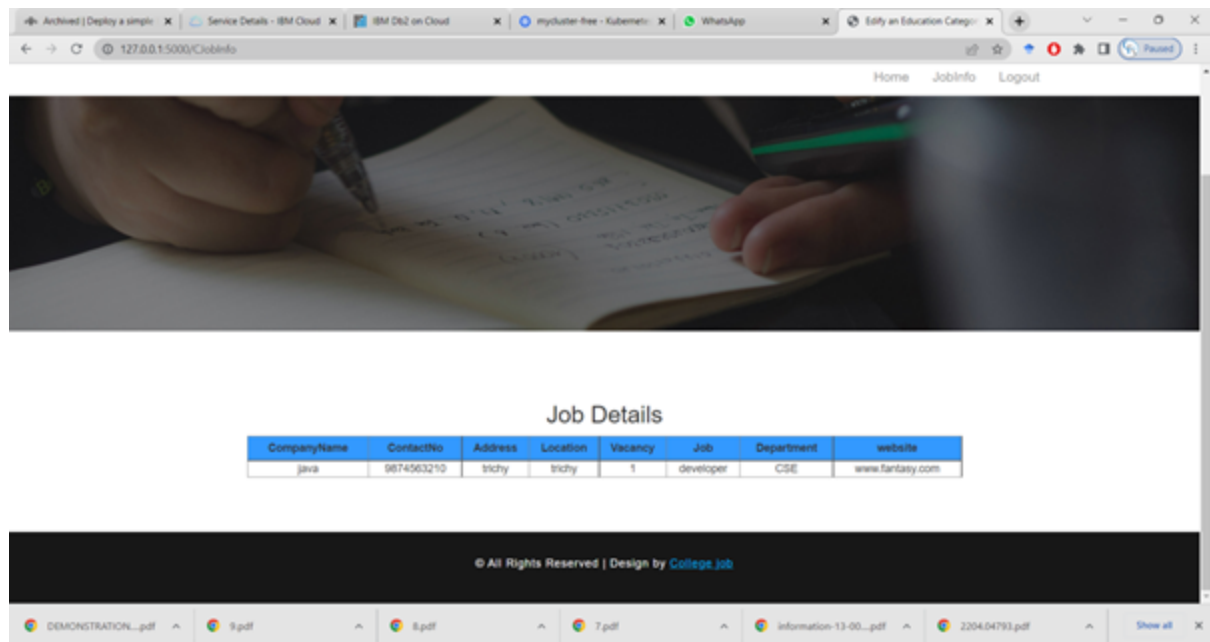


Company Login Here..!

User Name	<input type="text" value="fantasy"/>
Password	<input type="password" value="*****"/>

DEMONSTRATION...pdf 9.pdf 8.pdf 7.pdf Information-13-00...pdf 2204.04793.pdf Show all

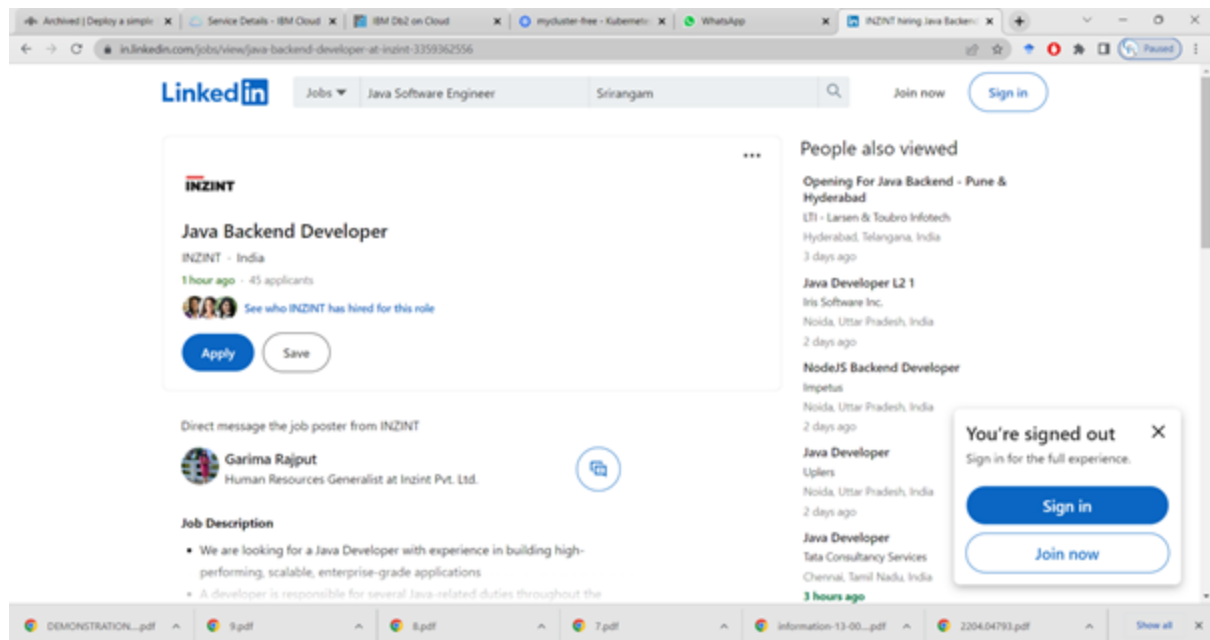
JOB DETAILS



The screenshot shows a web browser with multiple tabs. The active tab displays a web application with a header containing 'Home', 'JobInfo', and 'Logout' links. Below the header is a large image of hands writing on a notepad. Underneath the image, the title 'Job Details' is centered. Below the title is a table with the following data:

CompanyName	ContactNo	Address	Location	Vacancy	Job	Department	website
java	9874563210	trichy	trichy	1	developer	CSE	www.fantasy.com

Below the table, a footer bar contains the text '© All Rights Reserved | Design by College Job'. At the bottom of the browser window, a taskbar shows several PDF files: 'DEMONSTRATION...pdf', '9.pdf', '8.pdf', '7.pdf', 'Information-13-00...pdf', and '2204.04793.pdf'.



The screenshot shows a LinkedIn job posting for a 'Java Backend Developer' at 'INZINT'. The job was posted 1 hour ago and has 45 applicants. The job poster is 'Garima Rajput', a Human Resources Generalist at Inzint Pvt. Ltd. The job description states: 'We are looking for a Java Developer with experience in building high-performing, scalable, enterprise-grade applications' and 'A developer is responsible for several Java-related duties throughout the...'. On the right side, a list of 'People also viewed' includes: 'Opening For Java Backend - Pune & Hyderabad', 'Java Developer L2 1', 'NodeJS Backend Developer', 'Java Developer', and 'Java Developer'. A 'You're signed out' notification is visible on the right, with 'Sign in' and 'Join now' buttons. The browser's taskbar at the bottom shows the same PDF files as the first screenshot.

GITHUB & PROJECT DEMO LINK

<https://github.com/IBM-EPBL/IBM-Project-17016-1659626790>

DEMO LINK: <https://youtu.be/3Rz412OUbzM>

