SKILL AND JOB RECOMMENDER APPLICATION

PROJECT REPORT

A NAALAIYA THIRAN PROJECT REPORT

Submitted by

Team ID:PNT2022TMID16421

TEAM LEADER:

A.ASIBHA (6113191031013)

TEAM MEMBERS:

AISHWARYA (6113191031003)

T. ARUNDHATHI (6113191031010)

T.KANYAZHINI (6113191031045)

P.MADHUMITHA (6113191031053)

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

Department of Computer Science and Engineering

MAHENDRA ENGINEERING COLLEGE
(Autonomous)
Mahendhirapuri, Mallasamudram,
Namakkal DT -637 503

Project Report Format

1. **INTRODUCTION**

SKILL AND JOB RECOMMENDER APPLICATION

1 INTRODUCTION

PROJECT OVERVIEW

Nowadays, job search is a task commonly done on the Internet using job search engine sites like LinkedIn, Indeed and others. Commonly, a job seeker has two ways to search a job using these sites: 1) doing a query based on keywords related to the job vacancy that he/she is looking for, or 2) creating and/or updating a professional profile containing data related to his/her education, professional experience, professional skills and other, and receive personalized job recommendations based on this data. Sites providing support to the former case are more popular and have a simpler structure; however, their recommendations are less accurate than those of the sites using profile data.

Based on the person-job fit premise, we propose a framework for job recommendation based on professional skills of job seekers. We automatically extracted the skills from the job seeker profiles using a variety of text processing techniques. Therefore, we perform the job recommendation using TF-IDF and four different configurations of Word2vec over a dataset of job seeker profiles and job vacancies collected by us. Our experimental results show the performances of the evaluated methods and configurations and can be used as a guide to choose the most suitable method and configuration for job recommendation.

The remainder of this paper is organized as follows. We briefly describe the natural language processing methods we are used in our experimental setup. In Section 3 we present our proposal, including a new dataset collected by us and the framework for job recommendation.

1.2 PURPOSE

The purpose of the project is skill and job recommendation We started our journey by understanding the importance of a job recommendation system based on the skill set of the user. A system which can, not only recommend the job but also highlight the necessary skill set needed for the recommended job, which helps the user to learn more on the skill set

CHAPTER 2

LITERATURE SURVEY

2.1 Title: Job Recommendation through Progression of Job Selection

Author: Amber Nigam; Aakash Roy; Hartaran Singh;

Year: 2020

Description:

The task of job recommendation has been invariably solved using either a filter-based technique or through recommender systems where categorical features associated with jobs and candidates are used to generate recommendations. Through this paper, we are introducing a novel machine learning model which uses the candidates' job preference over time to incorporate the dynamics associated with highly volatile job market. In addition to that, our approach comprises several other smaller recommendations that contribute to problems of a) generating serendipitous recommendations b) solving the cold-start problem for new jobs and new candidates. We have used skills as embedded features to derive latent competencies from them, thereby expanding the skills of jobs and candidate to achieve more coverage in the skill domain. Our model has been developed and deployed in a real-world job recommender system and the best performance of the click-through rate metric has been achieved through a blend of machine learning and non-machine learning recommendations. The best results have been achieved through Bidirectional Long Short Term Memory Networks (Bi-LSTM) with Attention for recommending jobs through machine learning that forms a major part of our recommendation.

2.2 Title: A Recommender System for Open Educational Videos Based On Skill Requirements

Author: Mohammadreza Tavakoli; Sherzod Hakimov; Ralph Ewerth;

Year: 2020

Description:

In this paper, we suggest a novel method to help learners find relevant open educational videos to master skills demanded on the labour market. We have built a prototype, which 1) applies text classification and text mining methods on job vacancy announcements to match jobs and their required skills; 2) predicts the quality of videos; and 3) creates an open educational video recommender system to suggest personalized learning content to learners. For the first evaluation of this prototype we focused on the area of data science related jobs. Our prototype was evaluated by in-depth, semi-structured interviews. 15 subject matter experts provided feedback to assess how our recommender prototype performs in terms of its objectives, logic, and contribution to learning. More than 250 videos were recommended, and 82.8% of these recommendations were treated as useful by the interviewees. Moreover, interviews revealed that our personalized video recommender system, has the potential to improve the learning experience.

2.3 Title: GUapp: A Conversational Agent for Job Recommendation for the Italian Public Administration

Author: Vito Bellini; Giovanni Maria Biancofiore; Tommaso Di Noia; Eugenio Di Sciascio;

Year: 2021

Description:

GUapp is a platform for job-postings search and recommendation for the Italian public administration. The platform offers recommendation services with the aim of matching user skills and requests with job positions available in a given period of time. The recommender system implemented in GUapp, based on Latent Dirichlet Allocation, computes the k-nearest neighbour's job positions most similar to the user profile. Furthermore, in order to improve the user experience, GUapp implements a Chatbot whose goal is to allow users to interact with the app through natural language. Thanks to that, the search and recommendation process becomes incremental and the user can add new requirements at each stage of the interaction. In this paper we present GUapp, its recommender system, and the Chatbot developed for achieving an effective interaction with the user. In the next future, we will carry out in-vivo and in-vitro experiments for evaluating the system and its components in deep.

2.4 Title: A Staffing Recommender System based on Domain-Specific Knowledge Graph

Author: Yan Wang; Yacine Allouache; Christian Joubert

Year: 2022

Description:

In the economics environment, Job Matching is always a challenge involving the evolution of knowledge and skills. A good matching of skills and jobs can stimulate the growth of economics. Recommender System (RecSys), as one kind of Job Matching, can help the candidates predict the future job relevant to their preferences. However, RecSys still has the problem of cold start and data sparsity. The content-based filtering in RecSys needs the adaptive data for the specific staffing tasks of Bidirectional Encoder Representations from Transformers (BERT). In this paper, we propose a job RecSys based on skills and locations using a domain-specific Knowledge Graph (KG). This system has three parts: a pipeline of Named Entity Recognition (NER) and Relation Extraction (RE) using BERT; a standardization system for pre-processing, semantic enrichment and semantic similarity measurement; a domain-specific Knowledge Graph (KG). Two different relations in the KG are computed by cosine similarity and Term Frequency-Inverse Document Frequency (TF-IDF) respectively. The raw data used in the staffing RecSys include 3000 descriptions of job offers from Indeed, 126 Curriculum Vitae (CV) in English from Kaggle and 106 CV in French from Linux of Capgemini Engineering. The staffing RecSys is integrated under an architecture of Microservices. The autonomy and effectiveness of the staffing RecSys are verified through the experiment using Discounted Cumulative Gain (DCG). Finally, we propose several potential research directions for this research.

2.5 Title: Implementation K-Means Clustering Method in Job Recommendation System

Author: Betty Dewi Puspasari; Lany Lukita Damayanti; Andy Pramono;

Year: 2021

Description:

Work is important for everyone to earn income. With the large number of new graduates each year, finding job vacancies is a problem for students who have just completed their studies in higher education because they still do not have work experience so they are required to look for jobs that really match their criteria. Applications made can recommend specific job vacancies for undergraduates from universities (undergraduates) with the K-Means Clustering method. Applications in the form of websites that become third parties for companies and applicants. This application is one of the means that can provide solutions to companies and applicants in finding workers or jobs using a recommendation system. The problem to be studied is how to apply the K-Means Clustering method to the job vacancy recommendation system. The recommendation system in this application will calculate the level of match of the applicant's main skills, salary, location, and other skills with the needs of the company. The stages of making a recommendation system are making system designs and designs which include context diagrams, DFD, ERD and interface design. built with PHP, Java, jQuery, JavaScript, HTML, and CSS. Program testing is done by black box testing method. Questionnaire testing is given to applicants, companies, and admins with elements of testing based on user satisfaction, user convenience and system quality, resulting in the conclusion that the system can run well by getting a percentage of 87.6%.

2.6 Title: Convolutional Neural Network Based Career Recommender System for Pakistani Engineering Students

Author: Takreem Saeed; Muhammad Sufian; Mubashir Ali; Attique Ur Rehman

Year: 2022

Description:

In recent years, Recommender systems are utilized in a variety of areas. One reason behind why we want a recommender system in current society is that an individual has many alternatives to use because of the pervasiveness of the Internet. A recommender system seeks to estimate and predict user content preference. Old recommender systems used State-of-the-art recommender algorithms like content based filtering to predict ratings. Career Recommender system provides Engineering candidates the best possible available jobs relevant to their skills, qualification, etc. Four to six major engineering disciplines are covered in this recommender system. The proposed approach is tested using a career recommendation dataset which is collected from many students of different disciplines of various universities. A deep NLP based CNN model is used to predict the best jobs with maximum precision. 512 hidden layers are used to increase the performance of this system. Career recommendation takes care of the users and saves their cost and time spending on traditional job searching methods. Comparative study demonstrates that the proposed methodology of prediction of the best jobs achieves better results with an accuracy of 84% when matched with content based filtering technique where 81% accuracy is gained for content based career recommender system.

2.7 Title: Meta-Heuristic Algorithms for Learning Path Recommender at MOOC

Author: Ngo Tung Son; Jafreezal Jaafar; Izzatdin Abdul Aziz

Year: 2021

Description:

Online learning platforms, such as Coursera, Edx, Udemy, etc., offer thousands of courses with different content. These courses are often of discrete content. It leads the learner not to find a learning path in a vast volume of courses and contents, especially when they have no experience in advance. Streamlining the order of courses to create a well-defined learning path can help e-learners achieve their learning goals effectively and systematically. The learners usually ask the necessary skills that they expect to earn (query). The need is to develop a recommender system that can search for suitable learning paths. This study proposes a multi-objective optimization model as a knowledge-based recommender. Our model can generate an appropriate learning path for learners based on their background and job goals. The recommended studying path satisfies several learner criteria, such as the critical learning path, number of enrollments , learning duration, popularity, rating of previous learners, and cost. We have developed Metaheuristic algorithms includes the Genetic Algorithm (GA) and Ant Colony Optimization Algorithm (ACO), to solve the proposed model. Finally, we tested proposed methods with a dataset consisting of Coursera's courses and Vietnam work's jobs. The test results show the effectiveness of the proposed method.

2.8 Title: Meta-Heuristic Algorithms for Learning Path Recommender at MOOC

Author: NGO TUNG SON, JAFREEZAL JAAFAR

Year: 2021

Description:

Online learning platforms, such as Coursera, Edx, Udemy, etc., offer thousands of courses with different content. These courses are often of discrete content. It leads the learner not to find a learning path in a vast volume of courses and contents, especially when they have no experience in advance. Streamlining the order of courses to create a well-defined learning path can help e-learners achieve their learning goals effectively and systematically. The learners usually ask the necessary skills that they expect to earn (query). The need is to develop a recommender system that can search for suitable learning paths. This study proposes a multi-objective optimization model as a knowledge-based recommender. Our model can generate an appropriate learning path for learners based on their background and job goals. The recommended studying path satisfies several learner criteria, such as the critical learning path, number of enrollments, learning duration, popularity, rating of previous learners, and cost. We have developed Metaheuristic algorithms includes the Genetic Algorithm (GA) and Ant Colony Optimization Algorithm (ACO), to solve the proposed model. Finally, we tested proposed methods with a dataset consisting of Coursera's courses and Vietnam work's jobs. The test results show the effectiveness of the proposed method.

2.9 Title: Skill Analysis and Scouting Platform Using Machine Learning

Author: T. Subha; R. Ranjana; B. Aarthi; S. Pavithra;

Year: 2022

Description:

In a world where technology is rapidly advancing many firms have changed their traditional approach of recruiting the students based on their academic scores. In light of the technological advancement, improvement of placement records is a challenge for higher educational institutions because they do not adequately focus on training their students in career prospects. Therefore, the proposed study seeks to establish a Data Prediction system to analyze the technical knowledge of the students and the job seekers by predicting their ability to obtain a position in their ideal company based on their hands-on experience and skillsets. In addition, this model also proposes a recommendation system to suggest the domains that are thriving as well as the sectors that the candidate should concentrate to upgrade their skill. Many candidates will be benefitted through this model as they can analyze their skillsets and up skill themselves which in turn enhances the placement rate of the educational institutions. Many firms increasingly shortlist candidates based on their resumes, but some job seekers falsify their resume's skillsets. So as an additional feature this model also provides the recruiters with a complete see through of the candidate's technical skills and domain knowledge. The company can then take advantage of this to scout the most ideal candidate by making the right career opportunity available to the right people.

2.10 Title: SKYNET: A platform for maximizing career opportunities

Author: Aakash Kolekar

Year: 2022

Description:

Professional networking interfaces provide a significantly improved exposure and likelihood of job opportunities as well as career-related resources. Multiple availability of such platforms lead to a lack of standardization and thereby can result in a false sense of security. A massive profile pool has also statistically incremented the probability of overlooking suited candidate profiles at a global level. This paper presents a university-level individualized professional networking platform - SKYNET for enhanced prospects of job acquisition. SKYNET proved to be a successful amalgamation between career and academia which fundamentally aims on facilitating an enhanced job recommendation system based on individual professional skills. It is a Web application that smoothens the process of hiring and significantly improves an individual's chance of securing jobs and internships. It helps students to get a reality check to see where they stand among their peers as well, considering various parameters such as certifications and work experience. It will provide a customized preview of student and company profiles based on needed skills and expertise. The corresponding system also supports ranking and filtration based on the job profile and description which was achieved by a custom-designed algorithm. Results helped not only student candidates to find internships and jobs while pursuing primitive professional education which helped them work in a professional setting and apply the grasped knowledge in real-time. Furthermore, made it simple for collaborating recruiters to pick the perfect student-job-role match in a timely manner.

2.1 EXISTING SYSTEM

2.2 REFERENCE

[1] Covington, P., Adams, J., & Sargin, E. (2016, September). Deep neural networks for YouTube recommendations. In Proceedings of the 10th ACM conference on recommender systems (pp. 191-198). ACM.

[2] Gomez-Uribe, C. A., & Hunt, N. (2016). The Netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4), 13.

[3] Okura, S., Tagami, Y., Ono, S., & Tajima, A. (2017, August). Embedding-based news recommendation for millions of users. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1933-1942). ACM.

[4] Elsafty, A., Riedl, M., & Biemann, C. (2018, June). Document-based Recommender System for Job Postings using Dense Representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers) (pp. 216-224).

[5] Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge & Data Engineering, (6), 734-749.

[6] Abel, F., Deldjoo, Y., Elahi, M., & Kohlsdorf, D. (2017, August). Recsys challenge 2017: Offline and online evaluation. In Proceedings of the Eleventh ACM Conference on Recommender Systems (pp. 372-373). ACM.

[7] Chen, D., Ong, C. S., & Menon, A. K. (2019). Coldstart playlist recommendation with multitask learning. arXiv preprint arXiv:1901.06125.

[8] Jiang, M., Fang, Y., Xie, H., Chong, J., & Meng, M. (2019). User click prediction for personalized job recommendation. World Wide Web, 22(1), 325-345.

[9] Nigam, A., Sahare, P., & Pandya, K. (2019). Intent Detection and Slots Prompt in a Closed-Domain Chatbot. In IEEE 13th International Conference on Semantic Computing (ICSC) 2019 (pp. 340-343).

[10] Abel, F., Benczúr, A., Kohlsdorf, D., Larson, M., & Pálovics, R. (2016, September). Recsys challenge 2016: Job recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 425- 426). ACM.

[11] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[12] Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In International conference on machine learning (pp. 1188-1196).

[13] Zibriczky, D. (2016, September). A combination of simple models by forward predictor selection for job recommendation. In Proceedings of the Recommender Systems Challenge (p. 9). ACM.

[14] Volkovs, M., Yu, G. W., & Poutanen, T. (2017, August). Content-based neighbor models for cold start in recommender systems. In Proceedings of the Recommender Systems Challenge 2017 (p. 7). ACM.

[15] Liu, K., Shi, X., Kumar, A., Zhu, L., & Natarajan, P. (2016, September). Temporal learning and sequence modelling for a job recommender system. In Proceedings of the Recommender Systems Challenge (p. 7). ACM.

2.3 PROBLEM STATEMENT

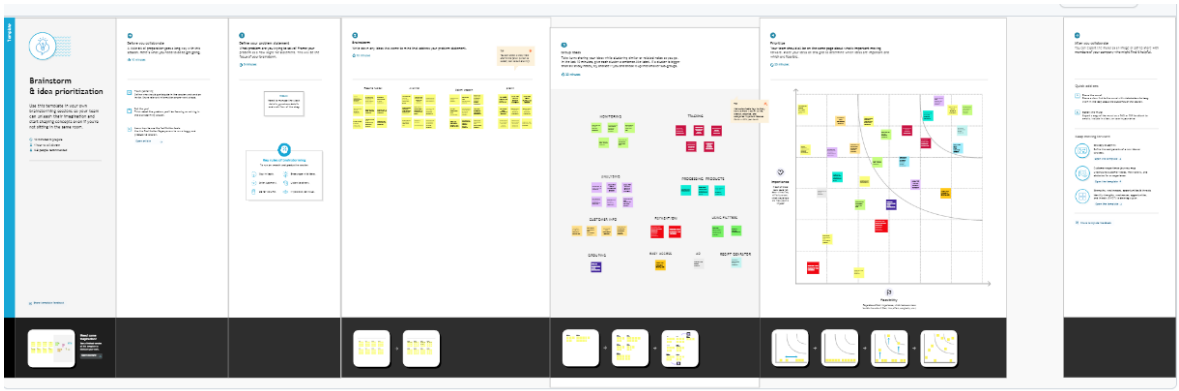
Create a problem statement to understand your customer's point of view. The customer problem statement template helps you focus on what matters to create experience people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solutions for the challenges your customers face. Throughout the process, you'll also be able to perceive your product or service.

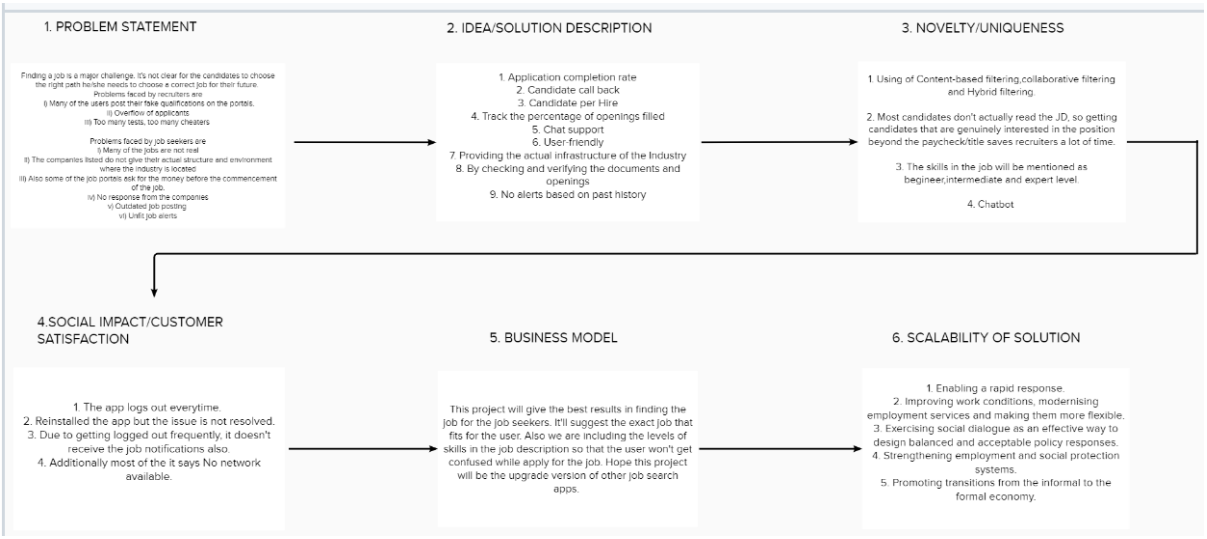
3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

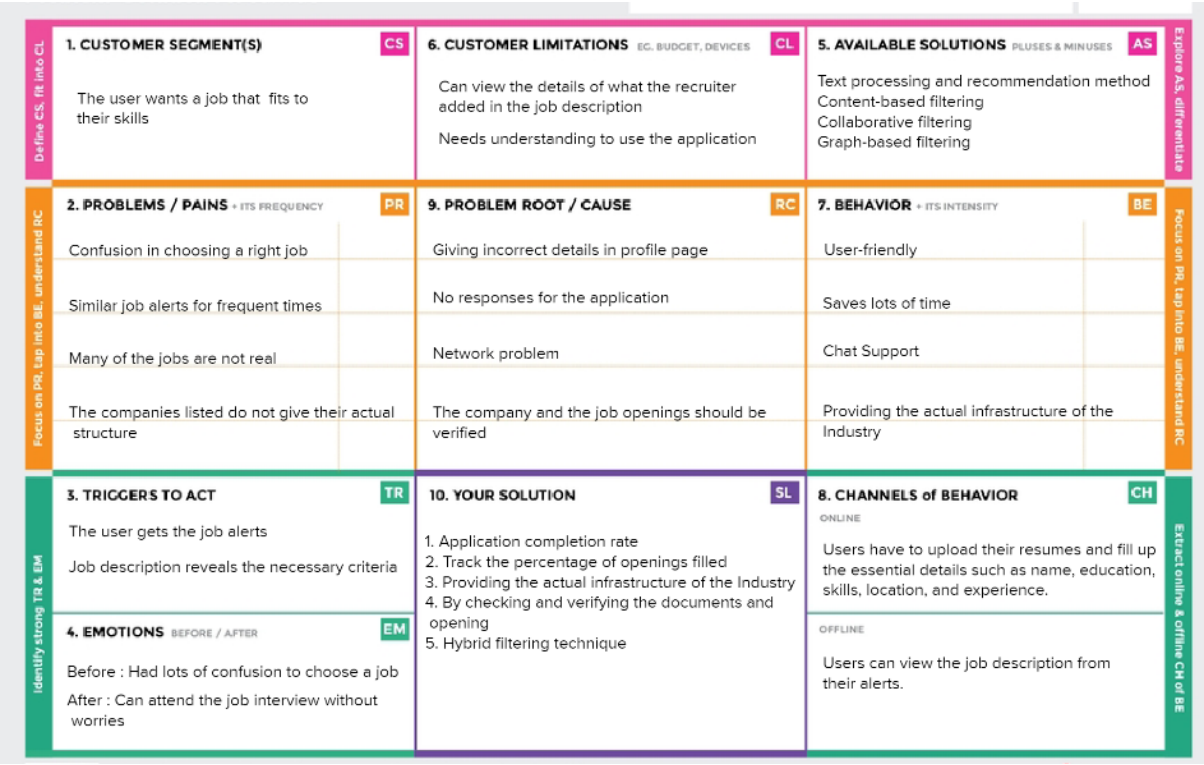
3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SYSTEM



3.4 PROBLEM SOLUTION FIT



4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Chat Bot	A Chat Bot will be there in website to solve user queries and problems related to applying a job, search for a job and much more.
FR-4	User Login	Login through Form Login through Gmail
FR-5	User Search	Exploration of Jobs based on job filters and skill recommendations.
FR-6	User Profile	Updation of the user profile through the login credentials
FR-7	User Acceptance	Confirmation of the Job.

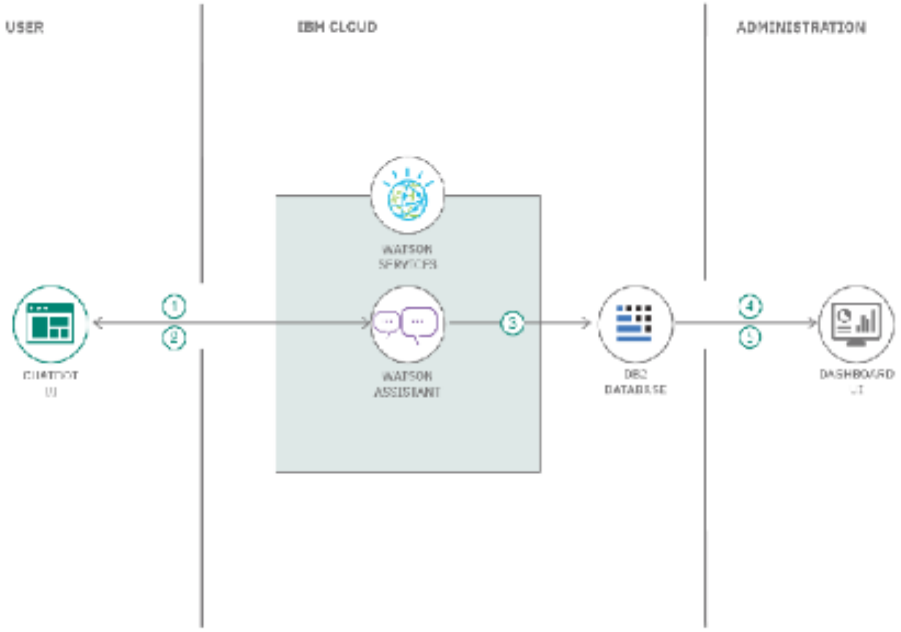
4.2 NON-FUNCTIONAL REQUIREMENTS

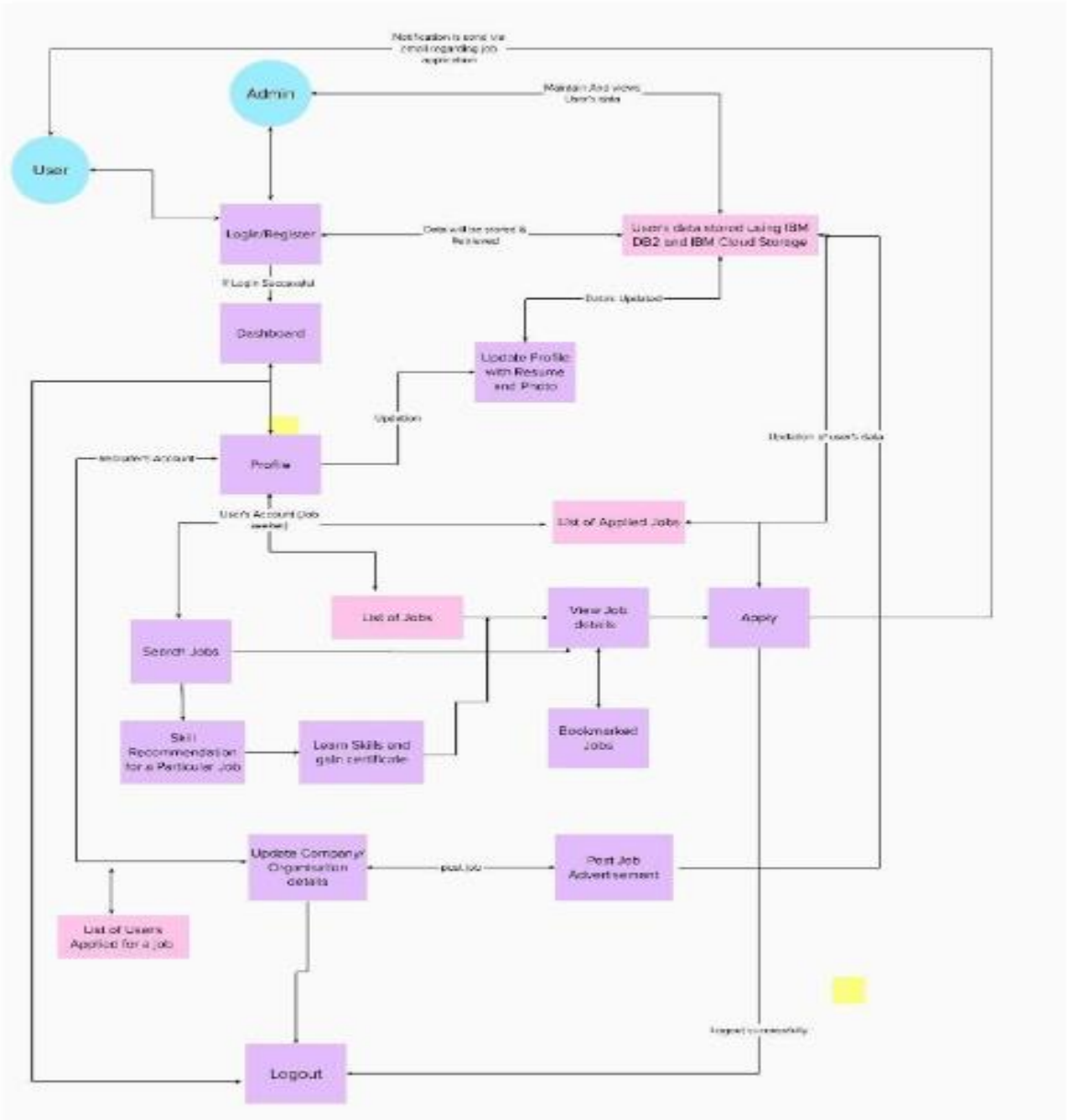
Following are the Non - functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	This application can be used by the job seekers to login and search for the job based on her Skills set.
NFR-2	**Security**	This application is secure with separate login for Job Seekers as well as Job Recruiters.
NFR-3	**Reliability**	This application is open-source and feel free to use, without need to pay anything. The enormous job openings will be provided to all the job seekers without any limitation.
NFR-4	**Performance**	The performance of this application is quicker response and takes lesser time to do any process.
NFR-5	**Availability**	This application provides job offers and recommends Skills for a Particular Job openings.
NFR-6	**Scalability**	The Response time of the application is quite faster compared to any other application.

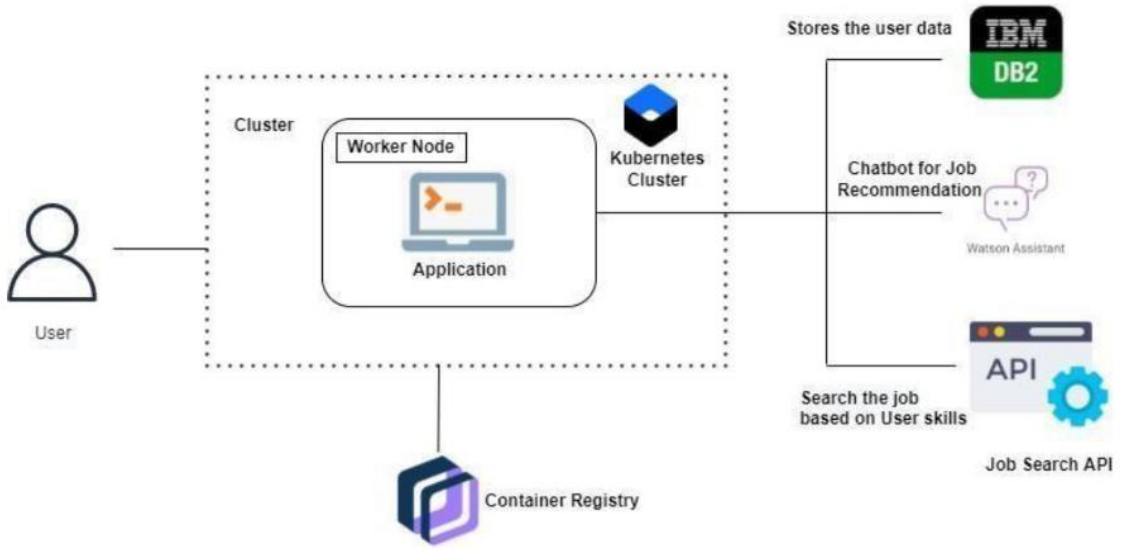
5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS




Admin
User
Login/Register
Users data stored using IBM DB2 and IBM Cloud Storage
Dashboard
Update Profile with Resume and Photo
Profile
List of Applied Jobs
List of Jobs
View Job details
Apply
Search Jobs
Skill Recommendation for a Particular Job
Learn Skills and gain certificate
Bookmarked Jobs
Update Company/ Organisation details
Post Job Advertisement
List of Users Applied for a job
Logout

5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 USER STORIES

Use the below template to list of the user stories foe the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile 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
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through online websites	I can register & access the dashboard with online website Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can receive confirmation Gmail & click confirm	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can receive confirmation email & click confirm	High	Sprint-1
	Dashboard					
Customer (Web user)		USN-6	As a user, I can able to take up the skill assessment and view the appropriate test score. Based on the skill sets I can able to get personalised job recommendations.	I can receive job recommendations	High	Sprint-1
Customer Care Executive		USN-7	As a customer care executive, we provide 24/7 chatbot support.	24/7 chatbot support	High	Sprint-1
Administrator		USN-8	As an administrator, I can able to view the progress and make required changes in the project	Deploy user specific and personalised job recommendations	High	Sprint-1

6 PROJECTS PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION:

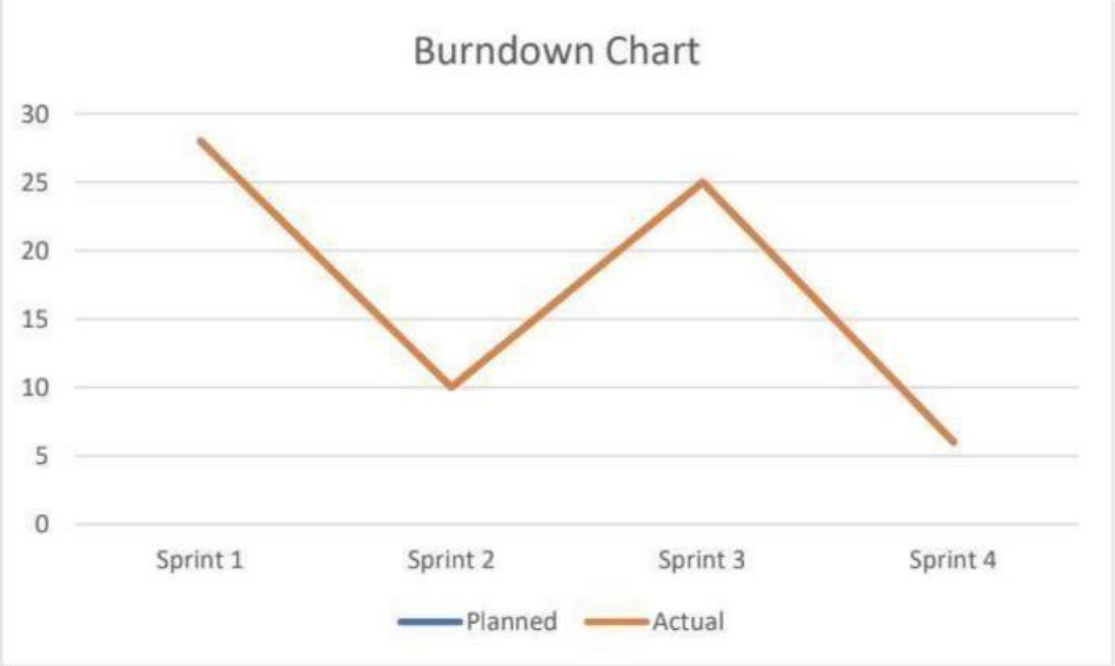
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Members
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account /dashboard	High	Asibha.A
		USN-2	As a user, I will receive confirmation emailonce I have registered for the application.	I can receive confirmation email & click confirm	High	Kanyazhini.T
		USN-3	As a user, I can register for the application through Facebook.	I can register & access thedashboard with Face bookLogin	Low	Aishwarya
		USN-4	As a user, I can register for the application through Gmail.		Medium	Madhumita.P
	Login	USN-5	As a user, I can log into the application by entering email & password.		High	Arundhathi.T
	Dashboard	USN-5	As a user, I can access my dashboard after signing in.	I can access my account / dashboard	High	Asibha.A
Customer (Web user)	Access	USN-6	As a user, I can setup a profile, and basic details by signing in.			Kanyazhini.T
		USN-7	As a user, I will upload my resume,	I can perform several task	Medium	Aishwarya

			Certificates, and other requirements.	in the application		
Customer Care Executive	Chat bot	USN-8	As a user, I can seek guidance from the customer care executive.		High	Madhumita.P
Administrator	DBMS	USN-9	As a administrator, I can keep the applications of your organization relies onrunning.	I can perform various modifications in the applications.	High	Arundhathi.T

6.2 Sprint Delivery Schedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	25 Oct 2022	31 Oct 2022	20	31 Oct 2022
Sprint-2	20	6 Days	01 Nov 2022	06 Nov 2022	18	06 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	13 Nov 2022	20	13 Nov 2022
Sprint-4	20	6 Days	13 Nov 2022	19 Nov 2022	19	19 Nov 2022

6.3 REPORTS FROM JIRA:



8 TESTING

8. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.1 TYPES OF TESTS

8.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is

the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.2 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

8.2.1 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

8.2.2 Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

8.2.3 Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.4 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Code

```
from flask import Flask, render_template, request, jsonify, session
import datetime
import re
```

```
import ibm_db

import pandas

import ibm_db_dbi

from sqlalchemy import create_engine


Engine = create_engine('sqlite://',

                       echo = False)


dsn_hostname = "b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"

dsn_uid = "bgx86936"

dsn_pwd = "LDBdZPnYhnaBy1iv"


dsn_driver = "{IBM DB2 ODBC DRIVER}"

dsn_database = "bludb"

dsn_port = "32304"

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.from_object(__name__)
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'
```

```
@app.route("/")
def homepage():

    return render_template('UserLogin.html')

@app.route("/alogin")
def alogin():

    return render_template('AdminLogin.html')

@app.route("/AdminHome")
def AdminHome():
```

```
    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')

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

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



@app.route("/NewProduct")
def NewProduct():

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

```
@app.route("/ProductInfo")
def ProductInfo():
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * from protb "
    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('ProductInfo.html',
data=engine.execute("SELECT * FROM Employee_Data").fetchall())



@app.route("/SalesInfo")
def SalesInfo():
```

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




@app.route("/Search")
def Search():

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * from protb "
    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('ViewProduct.html',
data=engine.execute("SELECT * FROM Employee_Data").fetchall())




@app.route("/viewproduct", methods=['GET', 'POST'])
def viewproduct():

    searc = request.form['subcat']




    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery =  "SELECT * from protb where SubCategory like '%" +
searc + "%' "
    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('ViewProduct.html',
data=engine.execute("SELECT * FROM Employee_Data").fetchall())

@app.route("/NewUser")
def NewUser():

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

    return render_template('index.html')


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

```
name1 = request.form['name']
gender1 = request.form['gender']
Age = request.form['age']
email = request.form['email']
address = request.form['address']
pnumber = request.form['phone']
uname = request.form['uname']
password = request.form['psw']

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

insertQuery = "INSERT INTO regtb VALUES ('" + name1 + "','" + gender1 + "','" + Age + "','" + email + "','" + pnumber + "','" + address + "','" + uname + "','" + password + "')"
insert_table = ibm_db.exec_immediate (conn, insertQuery)
print(insert_table)
```

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


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



        file = request.files['fileupload']
        file.save("static/upload/" + file.filename)



        ProductId =request.form['pid']
        Gender =request.form['gender']
        Category =request.form['cat']
        SubCategory=request.form['subcat']
        ProductType=request.form['ptype']
        Colour=request.form['color']
        Usage=request.form['usage']
        ProductTitle=request.form['ptitle']
        price = request.form['price']
        Image= file.filename
```

```
        ImageURL="static/upload/" + file.filename




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


        insertQuery = "INSERT INTO protb VALUES ('"+ ProductId +"','" + Gender + "','" + Category + "','" + SubCategory + "','" + ProductType + "','" + Colour + "','"+Usage +"','"+ProductTitle+"','"+ Image +"','"+ ImageURL +"','"+ price +"')"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)


        data1 = 'Record Saved!'
        return render_template('goback.html', data=data1)




@app.route("/userlogin", methods=['GET', 'POST'])
```

```
def userlogin():
    error = None
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['password']
        session['uname'] = request.form['uname']



        conn = ibm_db.connect(dsn, "", "")
        pd_conn = ibm_db_dbi.Connection(conn)

        selectQuery = "SELECT * from regtb where uname='" + 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 uname='" + 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('index.html',
data=engine.execute("SELECT * FROM Employee_Data").fetchall())




@app.route("/adminlogin", methods=['GET', 'POST'])
def adminlogin():
    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 admintb 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 "
            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('AdminHome.html',
data=engine.execute("SELECT * FROM Employee_Data").fetchall())




@app.route("/Remove", methods=['GET'])
def Remove():



   pid = request.args.get('id')
   conn = ibm_db.connect(dsn, "", "")
   pd_conn = ibm_db_dbi.Connection(conn)


   insertQuery = "Delete from protb  where id='"+ pid +"'"
   insert_table = ibm_db.exec_immediate(conn, insertQuery)
```

```
    selectQuery = "SELECT * from protb "
    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('ProductInfo.html',
data=engine.execute("SELECT * FROM Employee_Data").fetchall())



@app.route("/fullInfo")
def fullInfo():
    pid = request.args.get('pid')
    session['pid'] = pid

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

```
    pd_conn = ibm_db_dbi.Connection(conn)


    selectQuery = "SELECT * FROM protb where ProductId='" + pid + "'
"
    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('ProductFullInfo.html',
data=engine.execute("SELECT * FROM Employee_Data").fetchall())




@app.route("/Book", methods=['GET', 'POST'])
def Book():
```

```
if request.method == 'POST':


    uname = session['uname']
    pid = session['pid']

    qty = request.form['qty']

    ctype = request.form['ctype']
    cardno = request.form['cardno']
    cvno = request.form['cvno']


    Bookingid = ''
    ProductName =''
    UserName= uname
    Mobile=''
    Email=''
    Qty = qty
    Amount=''
```

```
        CardType = ctype
        CardNo = cardno
        CvNo = cvno
        date = datetime.datetime.now().strftime('%d-%b-%Y')



        conn = ibm_db.connect(dsn, "", "")
        pd_conn = ibm_db_dbi.Connection(conn)
        selectQuery = "SELECT * FROM protb where ProductId='" + pid +
"' "
        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()
        for item in data:
            ProductName = item[8]
            price = item[11]
            print(price)
            Amount = float(price) * float(Qty)

            print(Amount)
```

```
        selectQuery1 ="SELECT  *  FROM  regtb where  uame='" + uname +
"'"

        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]




        selectQuery = "SELECT  *  FROM  booktb"

        dataframe = pandas.read_sql(selectQuery, pd_conn)


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

        data2 = engine.execute("SELECT * FROM booktb").fetchall()

        count = 0


        for item in data2:
```

```
        count+=1



    Bookingid="BOOKID00" + str(count)



    insertQuery = "INSERT INTO booktb VALUES ('" + Bookingid +
"','"+ ProductName +"','" + price + "','" + uname + "','" + Mobile + "','"
+ Email + "','" + str(Qty) + "','" + str(Amount) + "','"+ str(CardType)
+"','"+ str(CardNo) +"','"+ str(CvNo) +"','"+ str(date) +"')"
    insert_table = ibm_db.exec_immediate(conn, insertQuery)


    sendmsg(Email,"order received delivery in one week ")
```

```
        selectQuery = "SELECT * FROM booktb where  uname= '" + uname
+ "' "
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        dataframe.to_sql('booktb1', con=engine, if_exists='append')
        data = engine.execute("SELECT * FROM booktb1").fetchall()

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



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
    s.quit()


@app.route("/UOrderInfo")
def UOrderInfo():

    uname = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
```

```
    selectQuery = "SELECT * FROM booktb where uname= '" + uname +
"' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()



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



@app.route("/UserHome")
def UserHome():

    uname = session['uname']



    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM regtb where  uname= '" + uname + "'
"
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
```

```
    data = engine.execute("SELECT * FROM booktb1").fetchall()



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



@app.route("/ASalesInfo")
def ASalesInfo():
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM booktb  "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb").fetchall()


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


def main():
    app.run(debug=True, use_reloader=True)
@app.route("/UReviewInfo")
def ureview():
    return render_template('NewReview.html')
```

```
if __name__ == '__main__':
    main()
```

```
html
<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Design by foolishdeveloper.com -->
    <title>job and skill</title>

    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&display=swap" rel="stylesheet">
    <!--Stylesheet-->
    <link rel="stylesheet" href="{{ url_for('static', filename='css/login.css') }}"  />

</head>
<body>
```

```
<div class="background">
    <div class="shape"></div>
    <div class="shape"></div>
</div>
<form id="form1" runat="server" method="post"
action="/userlogin">
    <h3>Login Here</h3>

    <label for="username">Username</label>
    <input type="text" placeholder="user name" name="uname"
id="username">

    <label for="password">Password</label>
    <input type="password" placeholder="Password" name="password"
id="password">

    <button>Log In</button>
    <div class="social">
      <div class="go"><a href="/NewUser"> Register</a></div>
      <div class="fb"><a href="/alogin" > Admin </a></div>
    </div>
</form>
```

</body>

</html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>job search</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta charset="utf-8">

<meta name="keywords" content="Cat Club Responsive web template, Bootstrap Web Templates, Flat Web Templates, Android Compatible web template,

Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyEricsson, Motorola web design" />

<script type="application/x-javascript"> addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false); function hideURLbar(){ window.scrollTo(0,1); } </script>

<link href='//fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='static/css'>

```
<link
href="//fonts.googleapis.com/css?family=Raleway:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">
<link
href='//fonts.googleapis.com/css?family=Roboto+Condensed:400,700italic,700,400italic,300italic,300' rel='stylesheet' type='static/static/text/css'>
<script src="static/js/jquery-1.11.1.min.js"></script>
<script src="static/js/bootstrap.js"></script>
<script type="text/javascript">
   jQuery(document).ready(function ($) {
      $(".scroll").click(function (event) {
         event.preventDefault();
         $('html,body').animate({ scrollTop: $(this.hash).offset().top }, 1000);
      });
   });
</script>
   <script src="https://kit.fontawesome.com/a076d05399.js"
crossorigin="anonymous"></script>
   <style>
      a{
         text-decoration: none;
         color: black;
      }
```

```
 nav{
    background: grey;
    height: 80px;
    width: 100%;
 }
nav ul{
    float: right;
   margin-right: 20px;
 }
 nav ul li{
    display: inline-block;
    line-height: 60px;
    margin: 0 5px;

 }
 nav ul li a{
    color: white;
    font-size: 17px;
    padding: 7px 13px;
    border-radius: 3px;
    text-transform: uppercase;
 }
```

```
a.active,a:hover{
    background: #1b9bff;
    transition: .5s;
}
.checkbtn{
    font-size: 30px;
    color: white;
    float: right;
    line-height: 80px;
    margin-right: 40px;
    cursor: pointer;
    display: none;
}
#check{
    display: none
}
@media (max-width: 952px){
    nav ul li a{
        font-size: 16px;
    }
}
@media (max-width: 858px){
```

```
.checkbtn{
    display: block;
}
ul{
    position: fixed;
    width: 100%;
    height: 100vh;
    background: #2c3e50;
    top: 80px;
    left: -100%;
    text-align: center;
    transition: all .5s;
}
nav ul li{
    display: block;
    margin: 50px 0;
    line-height: 30px

}
nav ul li a{
    font-size: 20px;
```

```
        }
        a:hover,a.active{
            background: none;
            color: #0082e6;
        }
        #checkchecked ~ ul{
            left: 0;
        }
    }

  </style>
  </head>
<body style="background-color: #080710; color: white;">
<h1 align= 'center'>
                                                <a style="color: white; "
href="/">JOB SEARCH</a>
                                            </h1>
  <nav>  <input type="checkbox" id="check" >
    <label for="check" class="checkbtn">
    <i class="fas fa-bars"></i></label>
    <ul>
```

**
Home
**

**
Admin Login
**

**
User Login
**

**
New User
**

** </nav>**

<form id="form" name="form" method="post" action="/RNewUser">

** <h2> New User Registration </h2> **

<table width="35%" border="0" align="center">

<tr>

<td></td>

</tr>

<tr>

<td width="45%" height="48">Name</td>

```
    <td width="55%"><input name="name" type="text" id="name"
required pattern="[A-Za-z ]{3,32}"/>
    </td>
  </tr>
  <tr>
    <td height="50">Gender</td>
    <td><input name="gender" type="radio" value="male" required />
      Male
      <input name="gender" type="radio" value="female" />
      Female</td>

  </tr>
  <tr>
    <td height="49">Age</td>
    <td>
      <input name="age" type="text" id="age" required size="3" />
    </td>
  </tr>
  <tr>
    <td height="48">Email Id</td>
    <td><input name="email" type="email" id="email" required /></td>
  </tr>
```

```
        <tr>
    <td height="46">Phone Number </td>
    <td><input name="phone" type="text" id="phone"   required
size="10" pattern="[0-9]{10}"/></td>
  </tr>



        <tr>
    <td height="50">Address</td>
    <td><textarea name="address" id="address"
required></textarea></td>
     </tr>

  <tr>
    <td height="40">User Name</td>
    <td><input name="uname" type="text" id="uname"  required/></td>
  </tr>



  <tr>
   <td height="53">Passwrod</td>
    <td><input name="psw" type="password" id="psw"  required/></td>
```

```
  </tr>


      <tr>
    <td height="72"> </td>
    <td><input name="btn" type="submit" id="btn" value="Submit" />
    <input type="reset" name="Submit2" value="Reset" /></td>
  </tr>
 </table>
       </form>
    <!-- copyright -->
      <div class="copyright">
          <div class="container">
              <p>©  All rights reserved | Design by <a href="#">JOB AND SKILL</a></p>
          </div>
      </div>
      <!-- //copyright -->
      <script src="static/js/responsiveslides.min.js"></script>
      <script src="static/js/SmoothScroll.min.js"></script>
<script type="text/javascript" src="static/js/move-top.js"></script>
<script type="text/javascript" src="static/js/easing.js"></script>
      <!-- here stars scrolling icon -->
```

```
        <script type="text/javascript">
            $(document).ready(function () {
                /*
                var defaults = {
                containerID: 'toTop', // fading element id
                containerHoverID: 'toTopHover', // fading element hover id
                scrollSpeed: 1200,
                easingType: 'linear'
                };
                */

                $().UItoTop({ easingType: 'easeOutQuart' });

            });
        </script>
<!-- //here ends scrolling icon -->
</body>
</html>



<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
<title>job search</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta charset="utf-8">
<meta name="keywords" content="Cat Club Responsive web template, Bootstrap Web Templates, Flat Web Templates, Android Compatible web template,
Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyEricsson, Motorola web design" />
<script type="application/x-javascript"> addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false); function hideURLbar(){ window.scrollTo(0,1); } </script>

<link href='//fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='static/css'>
<link href="//fonts.googleapis.com/css?family=Raleway:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">
<link href='//fonts.googleapis.com/css?family=Roboto+Condensed:400,700italic,700,400italic,300italic,300' rel='stylesheet' type='static/static/text/css'>
<script src="static/js/jquery-1.11.1.min.js"></script>
<script src="static/js/bootstrap.js"></script>
<script type="text/javascript">
    jQuery(document).ready(function ($) {
```

```
        $(".scroll").click(function (event) {
            event.preventDefault();
            $('html,body').animate({ scrollTop: $(this.hash).offset().top }, 1000);
        });
    });
</script>
    <script src="https://kit.fontawesome.com/a076d05399.js"
crossorigin="anonymous"></script>
    <style>
        a{
            text-decoration: none;
            color: black;
        }
        nav{
            background: grey;
            height: 80px;
            width: 100%;
        }
       nav ul{
            float: right;
          margin-right: 20px;
        }
```

```
nav ul li{
    display: inline-block;
    line-height: 60px;
    margin: 0 5px;

}
nav ul li a{
    color: white;
    font-size: 17px;
    padding: 7px 13px;
    border-radius: 3px;
    text-transform: uppercase;
}
a.active,a:hover{
    background: #1b9bff;
    transition: .5s;
}
.checkbtn{
    font-size: 30px;
    color: white;
    float: right;
    line-height: 80px;
```

```
    margin-right: 40px;
    cursor: pointer;
    display: none;
}
#check{
    display: none
}
@media (max-width: 952px){
    nav ul li a{
        font-size: 16px;
    }
}
@media (max-width: 858px){
    .checkbtn{
        display: block;
    }
    ul{
        position: fixed;
        width: 100%;
        height: 100vh;
        background: #2c3e50;
        top: 80px;
```

```
        left: -100%;
        text-align: center;
        transition: all .5s;
    }
    nav ul li{
        display: block;
        margin: 50px 0;
        line-height: 30px

    }
    nav ul li a{
        font-size: 20px;

    }
    a:hover,a.active{
        background: none;
        color: #0082e6;
    }
    #checkchecked ~ ul{
        left: 0;
    }
}
```

```
    </style>
    </head>
<body style="background-color: #080710; color: white;">
<h1 align= 'center'>
                                                <a style="color: white; "
href="/">JOB SEARCH</a>
                                            </h1>
    <nav>  <input type="checkbox" id="check" >
        <label for="check" class="checkbtn">
        <i class="fas fa-bars"></i></label>
        <ul>

                                                                <li><a
href="/">Home</a></li>
                                                                <li><a
href="/adminlogin">Admin Login</a></li>
                                                                <li><a
href="/UserLogin">User Login</a></li>
                                                                <li><a
href="/RNewUser">New User</a></li>
```

```
                                                    </ul> </nav>

        <form id="form" name="form" method="post"
action="/RNewUser">

            <div align="center" ><h2>  <strong>New User
Registration</strong>  </h2> </div>

             <table width="35%" border="0" align="center">


    <tr>

        <td></td>

    </tr>

    <tr>

      <td width="45%" height="48">Name</td>

      <td width="55%"><input name="name" type="text" id="name"
required pattern="[A-Za-z ]{3,32}"/>

      </td>

    </tr>

    <tr>

      <td height="50">Gender</td>

      <td><input name="gender" type="radio" value="male" required />

        Male

          <input name="gender" type="radio" value="female" />

          Female</td>
```

```
    </tr>
    <tr>
      <td height="49">Age</td>
      <td>
        <input name="age" type="text" id="age" required size="3" />
      </td>
    </tr>
    <tr>
      <td height="48">Email Id</td>
      <td><input name="email" type="email" id="email" required /></td>
    </tr>
          <tr>
      <td height="46">Phone Number </td>
      <td><input name="phone" type="text" id="phone"   required size="10" pattern="[0-9]{10}"/></td>
    </tr>


          <tr>
      <td height="50">Address</td>
```

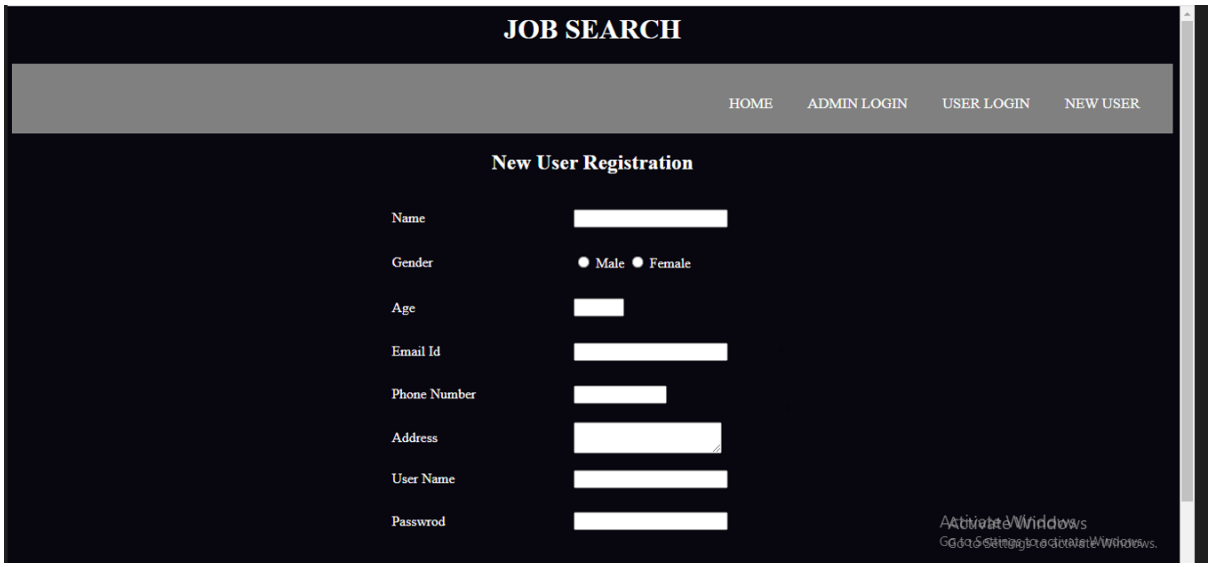
```
    <td><textarea name="address" id="address"
required></textarea></td>
     </tr>

   <tr>
    <td height="40">User Name</td>
    <td><input name="uname" type="text" id="uname"  required/></td>
   </tr>

   <tr>
    <td height="53">Passwrod</td>
    <td><input name="psw" type="password" id="psw"  required/></td>
   </tr>

       <tr>
    <td height="72"> </td>
    <td><input name="btn" type="submit" id="btn" value="Submit" />
    <input type="reset" name="Submit2" value="Reset" /></td>
   </tr>
  </table>
        </form>
```

```
    <!-- copyright -->
        <div class="copyright">
            <div class="container">
                <p>© All rights reserved | Design by <a href="#">JOB AND SKILL</a></p>
            </div>
        </div>
        <!-- //copyright -->
        <script src="static/js/responsiveslides.min.js"></script>
        <script src="static/js/SmoothScroll.min.js"></script>
<script type="text/javascript" src="static/js/move-top.js"></script>
<script type="text/javascript" src="static/js/easing.js"></script>
        <!-- here stars scrolling icon -->
        <script type="text/javascript">
            $(document).ready(function () {
                /*
                var defaults = {
                containerID: 'toTop', // fading element id
                containerHoverID: 'toTopHover', // fading element hover id
                scrollSpeed: 1200,
                easingType: 'linear'
                };
```

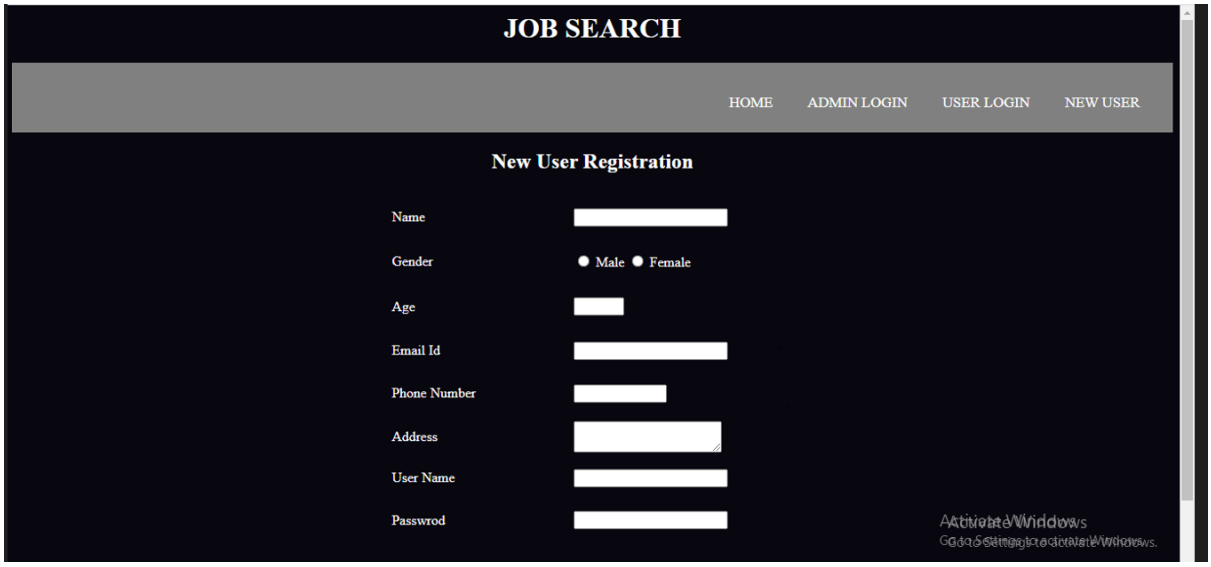
```
            */

            $().UItoTop({ easingType: 'easeOutQuart' });

        });
    </script>
<!-- //here ends scrolling icon -->
</body>
</html>
```

9 RESULT

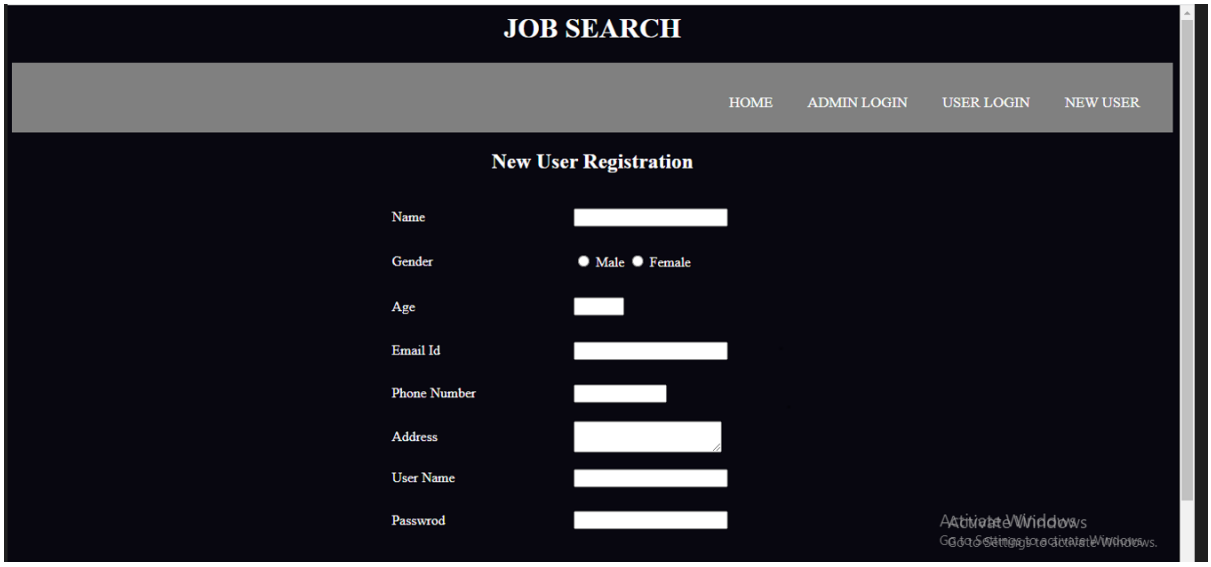
Registration



Recommented jobs



Perfect job



10 ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

- It provides the user-friendly account
- We can provide job recommendation
- Its user friendly

DISADVANTAGES:

- Its not deployed in real time implementation
- It’s not support when job seakers high

11 CONCLUSIONS:

a novel blended approach that leverages progression of job selection by candidates and attempts to make job recommendations serendipitous. Using blended methods, recommendations suggested to candidates are based on their interaction history with jobs, along with jobs that are a) similar to the other jobs applied by the candidate and b) Figure 4: Bi-LSTM model with Attention applied by similar candidates. Our approach naturally solves the candidate and job cold-start problem in the absence of interaction data. We also demonstrated the use of latent competency groups which expand the job skill requirements and the candidate skills thereby attempting to reveal latent competencies and achieve more coverage in the skill domain. Using our methodology, we see a relative increase in clickthrough rates of candidates visiting our portal and applying for jobs.

12 FUTURE SCOPE:

In future scope we implemented real time implementation A letter of recommendation is **a letter from a professional contact in your network—past or present—endorsing you for a job or position**. This letter is a testament on behalf of the writer that you possess the necessary skills, positive demeanor, and potential to be successful in the role you're seeking. In this paper, we presented a job recommender model aiming to extract meaningful data from job postings using text-clustering methods. As a result, job offers are divided into job clusters based on their common features and job offers are matched to job seekers according to their interactions. Our future Work

will focus on training and evaluating our model using Word2vec method and k-means clustering algorithms used to capture and represent the context of job profiles. Subsequently, it will be easy to match set of job offers to a given job seeker based on its past interactions toward specific job offers. The dataset that will be used is built from scraping job search websites.

SOURCE CODE:

GITHUB & PROJECT DEMO LINK: https://github.com/IBM-EPBL/IBM-Project-36363-1660294508