**B.Tech Project Report (CSPE 40)**

**on**

**Job Recommendation System**

**using user-skills**

**by**

**Ajay Kumar (12012005)**

**Mukesh (12012087)**

**Vivek Kumar (12012031)**

**Under the Supervision of Dr. Niyati Baliyan**

**Asst. Prof.**



**Department of Computer Engineering National Institute of Technology, Kurukshetra Haryana-136119, India**

**Jan-May 2024**



***Certificate***

We, hereby certify that the work which is being presented in this B.Tech Project (CSPE40) report entitled “***Job Recommendation system based on user-skills***”, in partial fulfillment of the requirements for the **Bachelor of Technology in Computer Engineering** is an authentic record of our own work carried out during a period from January, 2024 to May, 2024 under the supervision of **Dr. Niyati Baliyan, Assistant Professor**, Computer Engineering Department.

The matter presented in this project report has not been submitted for the award of any other degree elsewhere.

*Signature of Candidate*

**Ajay Kumar (12012005)**

**Mukesh (12012087)**

**Vivek Kumar (12012031)**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

**Date**: 03.04.2024

*Signature of Supervisor Faculty Mentor*

## Dr. Niyati Baliyan

Asst. Prof.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **SNo** | **Title** | **Page No** |
|  | *Abstract* | 1 |
| 1 | Introduction | 2 |
| 2 | Motivation | 3 |
| 3 | Related Work/Background | 4 |
| 4 | Proposed Work | 4 |
|  | *4.1 Conceptual Design Diagram* | 6 |
|  | *4.2 Proposed Algorithm <if Any>* | 7 |
|  | *4.3 Working Example <if Any>* | 9 |
| 5 | Experimental Setup *<if any>* |  |
|  | *5.1 Environment Setting (HW/SW requirements)* | 10 |
|  | *5.2 Details of Data and Data Modeling <if any>* | 11 |
|  | *5.3 Experimental Observations and Analysis <if any>* | 13 |
| 6 | Current Status and Future Plans of Project Work | 16 |
| 7 | Conclusion | 16 |
|  | *References* <if any> | 16 |
| *Appendix:* <if any> | | |
|  | *A. Hypertension Code* | 17 |
|  | *B. Diabetes Code* | 23 |
|  | *C. Coronary Heart Disease (CHD) Code* | 28 |
|  | *D. Stroke Code* | 33 |
|  | *E. Integrated Mode Running Code* | 43 |

***Abstract***

In an era of rapid technological advancement and evolving job markets, the efficient matching of job seekers with suitable employment opportunities is of paramount importance. This project introduces a novel approach to job recommendation by leveraging user skills and qualifications. Through the integration of machine learning algorithms and data analysis techniques, a recommendation engine has been developed to provide personalized job suggestions to users. The system aggregates job listings from diverse sources and profiles users based on their skills and preferences. By analyzing user profiles and job requirements, the recommendation engine identifies relevant job opportunities, thereby streamlining the job search process for both job seekers and employers. Evaluation of the system's performance demonstrates its effectiveness in accurately matching users with suitable job openings. Additionally, considerations for privacy protection and ethical implications are addressed to ensure the fairness and transparency of job recommendations. Overall, this project contributes to the advancement of job matching technologies, offering a valuable tool for enhancing the efficiency and effectiveness of the job search process in today's competitive employment landscape.

# Introduction

In today's dynamic job market, the process of finding suitable employment opportunities can be daunting for job seekers and employers alike. Job seekers often struggle to identify positions that align with their skills and qualifications, while employers face challenges in efficiently filtering through numerous applications to find the most suitable candidates. To address these issues, this project focuses on developing a job recommendation system that utilizes user skills and qualifications as key criteria for matching job seekers with relevant job openings.

By harnessing the power of machine learning algorithms and data analysis techniques, the system aims to streamline the job search process by providing personalized job recommendations tailored to the individual profiles of users. Through the aggregation and analysis of job listings and user profiles, the recommendation engine identifies potential matches, thereby facilitating efficient job placements. This project not only aims to enhance the effectiveness of job searches but also contributes to the advancement of job matching technologies, ultimately fostering a more efficient and equitable job market ecosystem.

# Motivation

The motivation behind this project stems from the recognition of the profound challenges faced by both job seekers and employers in today's fast-paced and competitive job market. Job seekers often find themselves overwhelmed by the sheer volume of job listings available, leading to frustration and inefficiency in their search for suitable employment opportunities. Conversely, employers struggle to identify qualified candidates amidst a sea of resumes and applications, resulting in time-consuming and resource-intensive recruitment processes.

Moreover, traditional job search methods often fail to account for the nuanced skills and qualifications of individual job seekers, leading to mismatches between candidates and job openings. This not only hampers the job search process but also results in missed opportunities for both job seekers and employers.

By developing a job recommendation system that leverages user skills and qualifications, we aim to address these challenges and streamline the job search process for all stakeholders involved. Through the application of advanced machine learning algorithms and data analysis techniques, we seek to create a platform that provides personalized job recommendations tailored to the unique profiles of individual users.

# Related Work

**1. Collaborative Filtering:**

Collaborative filtering is a technique used in recommendation systems to generate predictions or recommendations by leveraging the preferences or behaviors of similar users. It operates under the assumption that users who have agreed in the past on certain items are likely to agree again in the future. Collaborative filtering does not require explicit information about the items or users; instead, it relies solely on the user-item interaction data.

There are two main types of collaborative filtering:

* **User-Based Collaborative Filtering:** This approach identifies users who have similar preferences to the target user and recommends items that they have liked or interacted with. It calculates the similarity between users based on their historical interactions with items and then suggests items that similar users have rated highly.
* **Item-Based Collaborative Filtering:** In this approach, the system first calculates the similarity between items based on how users have interacted with them. It then recommends items that are similar to those that the target user has already liked or interacted with.

Collaborative filtering is effective in capturing user preferences and providing personalized recommendations. However, it may suffer from the cold start problem when dealing with new users or items with limited interaction data. Additionally, scalability can be an issue with large datasets due to the computational overhead of calculating user or item similarities.

**2. Content-Based Filtering:**

Content-based filtering recommends items to users based on the characteristics or features of the items themselves, as well as the user's preferences or profile. It analyzes the content or attributes of items and compares them to the user's preferences to generate recommendations.

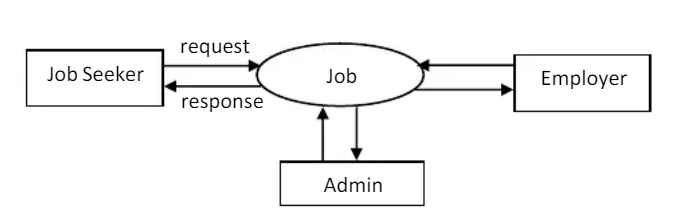
The process of content-based filtering typically involves the following steps:

* **Feature Extraction:** Extract relevant features or attributes from the items, such as keywords, genres, or metadata.
* **Profile Creation:** Create a user profile based on their preferences, past interactions, or explicit feedback. This profile is used to capture the user's preferences and interests.
* **Matching:** Compare the features of items to the user profile using similarity measures such as cosine similarity or Euclidean distance. Items that are most similar to the user profile are recommended to the user.

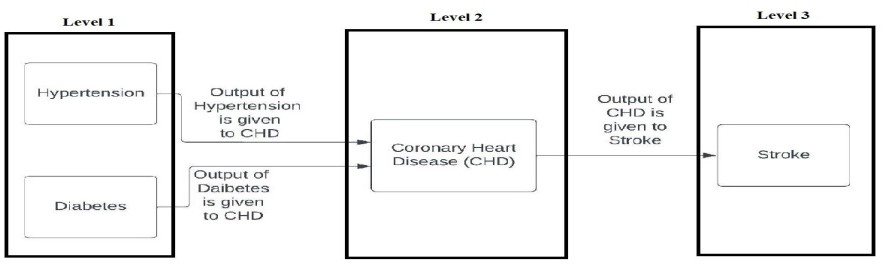
Content-based filtering is advantageous because it can provide recommendations for new users or items without relying on historical interaction data. It also offers transparency in the recommendation process since recommendations are based on explicit features of the items and user preferences. However, content-based filtering may suffer from the overspecialization problem, where recommendations are limited to items similar to those the user has already interacted with, leading to a lack of diversity in recommendations.

# Proposed Work

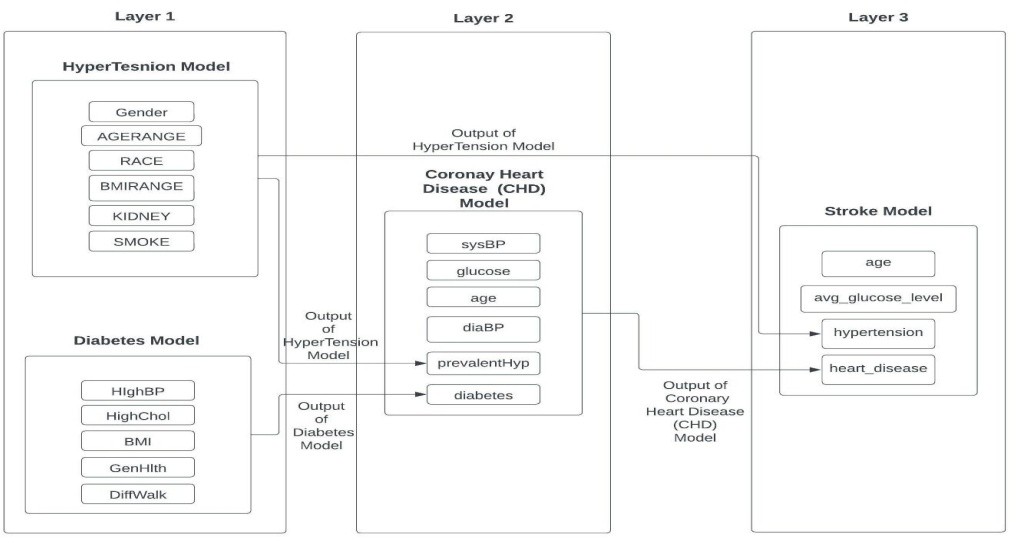
## Conceptual Design Diagram

**

**Figure 1**.*Level 0 DFD*



**Figure** 1.*Level 1 DFD*



# Proposed Algorithm

**Figure 3**.*Level 2 DFD*

**<***if any, add here a Pseudo-code, lemma, etc to explain the internal steps of the proposed solution>***>**

# Working Example

*<if Any, add here a self-explanatory example of the proposed Solution>*

# Experimental Setup

## Data Settings

In this project, we first run “scrapeJobs” models for creating a database of recent jobs. Then, database of is used in job recommdation

A brief detail about job dataset used:

* + - Job Dataset creation

|  |  |
| --- | --- |
| * Library used | Features in dataset |
| 1. bs4 2. time 3. pymongo 4. selenium | * TITLE |
|  | o EXPERIENCE |
|  |  |
|  | * SALARY |
|  |  |
|  |  |
|  | o LOCATION |
|  |  |
|  | * SKILLS |
|  |  |

Library used:

Flask

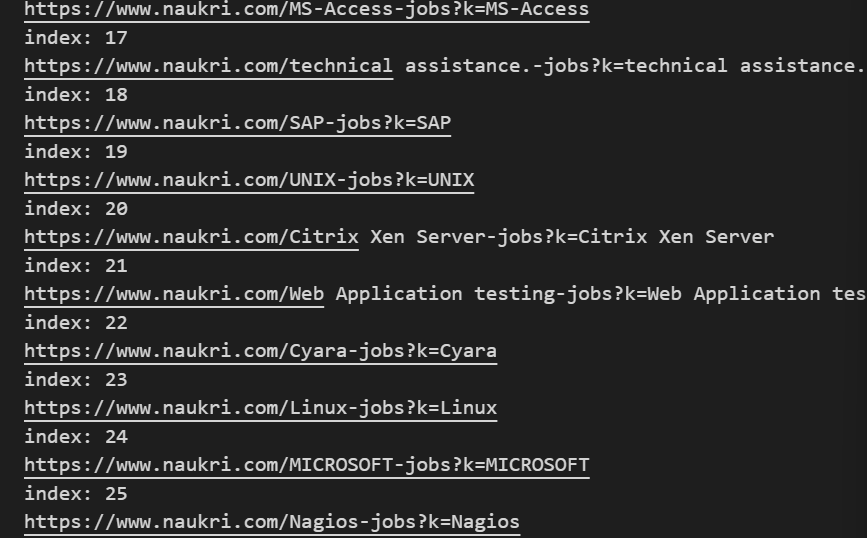
Pymongo

Operator

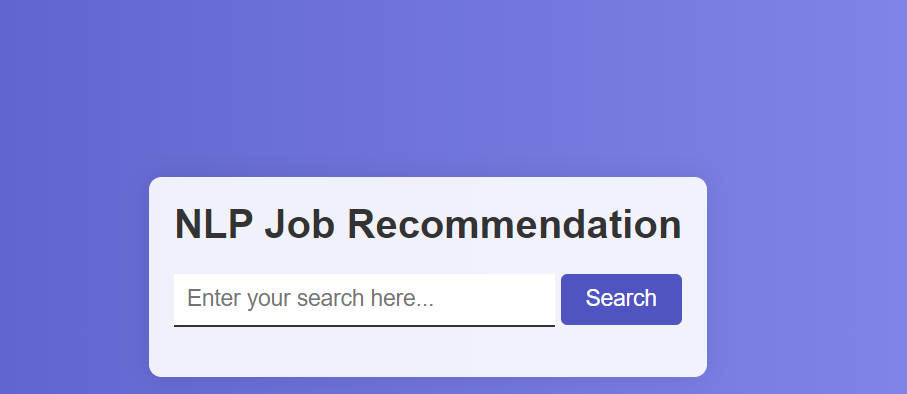
Spacy

Nltk

* 1. **Result**
* **Creating a database for new job**

****

* **UI for taking input**

****

## Recommending Jobs

## 

## Phase 1:

Creating database:

1. Scraping job data from www.naukri.com
2. Storing job data in mongodb

## Phase 2:

Creating Recommending job list and show it in UI:

1. Python libraries “nltk” and trie data structure is used for extracting skills.
2. On basis of skill , job is shown.

## Current Status and Future Plans of Project Work

* 1. **Current Status**

Following works are completed in the project work:

1. Job database creation is completed.
2. UI generated for taking text.
3. Recommendations based on skill is completed.

## Planned Future Work

Following work are planned for the upcoming duration of the semester:

1. Using resume uploading for recommending job
2. Using linked in profile for recommending job
3. Using better libraries than “nltk” for extracting skills

# Conclusion

Through this project, we were able to successfully develop a Job Recommendation system using a Machine Learning Model. We used the Content-Based Filtering algorithm for the recommendation. We explored different recommendation algorithms and chose Content based filtering for our implementation which uses similarity between jobs and user skills for recommendations. We used the selenium to interact with the database. We developed a easy to use, attractive UI using HTML and CSS. We were able to simplify the tedious process of job hunting by providing a centralized application which gives details of all the relevant jobs based on skills and interested domains of the user. It can also help people from the IT sector to evaluate their talents, skills and explore a particular job profile from numerous available options.

# References

[1] B. D. Puspasari, L. L. Damayant i, A. Pramono and A. K. Darmawan,

“Implement ation K-Means Clustering Method in Job Recommendat

ion Syst em,” in *2019 5th International Conference on*

*Big Data and Information Analytics (BigDIA), 2019*, pp. 1-6, doi:

10.1109/ICEEIE52663.2021.9616654.

[2] M. Jianjun, “ Research on collaborat ive filt ering recommendat ion algorithm

based on user behavior charact erist ics,” in *2020 International Conference*

*on Big Data & Artificial Intelligence & Software Engineering*

*(ICBASE), 2020*, pp. 425-428, doi: 10.1109/ICBASE51474.2020.00096.

[3] Appadoo, Kevin Soonnoo, Muhammad & Mungloo-Dilmohamud,

Zahra., “ JobFit : Job Recommendat ion using Machine Learning and Recommendation

Engine,” *2020*, 1-6.10.1109/CSDE50874.2020.9411584.

[4] A. Nigam, A. Roy, H. Singh and H. Waila, “ Job Recommendat ion

through P rogression of Job Select ion,” *2019 IEEE 6th International*

*Conference on Cloud Computing and Intelligence Systems (CCIS), 2019*,

pp. 212-216, doi:10.1109/CCIS48116.2019.9073723.

[5] S. Gadegaonkar, D. Lakhwani, S. Marwaha and P. A. Salunke, "Job Recommendation System using Machine Learning," 2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2023, pp. 596-603, doi: 10.1109/ICAIS56108.2023.10073757.

[6] L. Zhao and Z. Pan, “Research on Online Course Recommendat ion

Model Based on Improved Collaborat ive Filt ering Algorithm,” *P2021*

*IEEE 6th International Conference on Cloud Computing and Big*

*Data Analytics (ICCCBDA), 2021*, pp. 437-440, doi: 10.1109/ICCCBDA51879.2021.9442575.

**APPENDIX**

# Scraping job data

from bs4 import BeautifulSoup

from time import sleep

from skill\_list import skills

from pymongo import MongoClient

import pymongo

from selenium import webdriver

from selenium.webdriver.chrome.service import Service

options = webdriver.ChromeOptions()

web = webdriver.Chrome(options=options)

connection\_string = "mongodb://localhost:27017"

client = MongoClient(connection\_string)

def do():

ex = []

wrong = []

db = client['jobs']

print("hi")

print(db)

db.create\_collection("narkuri")

print("Database created........")

t = 0

for search in skills:

try:

print("index:" ,t)

t+=1

search\_title = search

url = "https://www.naukri.com/" + search\_title + "-jobs?k=" + search\_title

print(url)

web.get(url)

sleep(1)

html = web.page\_source

soup = BeautifulSoup(html)

data = []

TITLE = soup.find\_all("a", {"class": "title"})

EXPERIENCE = soup.find\_all("li", {"class": "experience"})

SALARY = soup.find\_all("li", {"class": "salary"})

LOCATION = soup.find\_all("li", {"class": "location"})

SKILLS = soup.find\_all("ul", {"class": "has-description"})

for i in range(20):

x = {

'title' : TITLE[i].text,

'url' : TITLE[i].get('href'),

'experience': EXPERIENCE[i].text,

'salary' : SALARY[i].text,

'location' : (LOCATION[i].text).split(', '),

'skills' : [j.text for j in SKILLS[i]],

'search\_type' : search\_title

}

data.append(x)

db = client['jobs']

mydb = db['narkuri']

x = mydb.insert\_many(data)

except Exception as e:

ex.append(e)

wrong.append((search , url))

do()

2.Running web app

from flask import Flask , render\_template , url\_for , redirect , abort , request

from pymongo import MongoClient

import pymongo

import operator

import spacy

import nltk

from trie import insertList,filtered\_skills

insertList()

app = Flask(\_\_name\_\_)

#EXTRACT SKILLS FROM THE GIVEN TEXT

def extract\_information\_from\_user(text):

key=[]

value=[]

tokens= nltk.word\_tokenize(text)

stopwords = nltk.corpus.stopwords.words('english')

SKILLS = [word for word in tokens if word not in stopwords]

SKILLS=filtered\_skills(SKILLS)

Dict = {key[i]: value[i] for i in range(len(key))}

#SKILLS= doc.split(",")

print(SKILLS)

Dict.update(SKILLS=SKILLS)

text = Dict["SKILLS"]

print(text)

return retirve\_info\_from\_db(text)

#RETRIVE RELATED JOBS BASED ON JACCARD COEFFICIENT

def retirve\_info\_from\_db(user\_list):

len\_user\_list = len(user\_list)

n = mydb.find( { 'skills': { '$in': user\_list}} ,{'\_id':0}) #COLLECTING JOBS BASED ON MATCHING SKILLS

jobs = []

for i in n:

job\_skills = i['skills']

match = len([k for k , val in enumerate(job\_skills) if val in user\_list])

total\_len = len(job\_skills) + len\_user\_list

i['rank'] = match/total\_len #RANKING COEFFICIENT

jobs.append(i)

return show\_info(jobs , user\_list , len(jobs))

#SORT THE JOBS RANK WISE AND DISPLAY

def show\_info(jobs , job\_skills , job\_len):

jobs.sort(key=operator.itemgetter('rank') , reverse=True) #SORTING JOBS BASED ON THE RANK SCORE

return render\_template('show\_job.html' , jobs=jobs , job\_skills=job\_skills , job\_len=job\_len)

@app.route('/')

def hello():

return render\_template('index.html')

@app.route('/', methods=['POST'])

def my\_form\_post():

text = request.form['text']

print("fahfakshfaskhfask ",text)

return extract\_information\_from\_user(text)

if \_\_name\_\_ == "\_\_main\_\_":

#CONNECTING WITH MONGO DB

connection\_string = "mongodb://localhost:27017"

client = MongoClient(connection\_string)

db = client['jobs']

mydb = db['narkuri\_tech\_jobs']

#STARTING THE APPLICATION

app.run(host="0.0.0.0" ,port=5000, debug = True)