JOB SEEKING APPLICATION

UCS2201 – Fundamentals and Practice of Software Development

A PROJECT REPORT

Submitted By

Abbhinav.E 3122225001001

Ananth Narayanan.P 3122225001010

Bhuvan.S 3122225001019

Department of Computer Science and Engineering

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Kalavakkam – 603110

July 2023

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this project report titled "Job Seeking Application" is the bonafide work of Abbhinav.E (3122225001001), Ananth Narayanan.P (3122225001010), and Bhuvan.S (3122225001019) who carried out the project work in the UCS2201 – Fundamentals and Practice of Software Development during the academic year 2022-23.

Internal Examiner External Examiner

Date:

TABLE OF CONTENTS

PROBLEM STATEMENT

JOB SEEKING APPLICATION

To create a mediating platform through which,

i. Job seekers can explore open positions which match their profiles.

ii. Talent acquisition managers can search for deserving applicants who could qualify for their job requirements.

EXTENDED EXPLORATION OF PROBLEM STATEMENT

ATS (APPLICANT TRACKING SYSTEM)

ATS (Applicant Tracking System) is a software application used by companies and recruiters to manage and streamline the hiring process. It is designed to automate, simplify, and centralize the various stages of recruitment, from receiving and reviewing resumes to scheduling interviews and making hiring decisions. ATSs serve as a repository for job openings and candidate profiles, allowing recruiters to track and manage applicants efficiently. Key features of ATSs typically include:

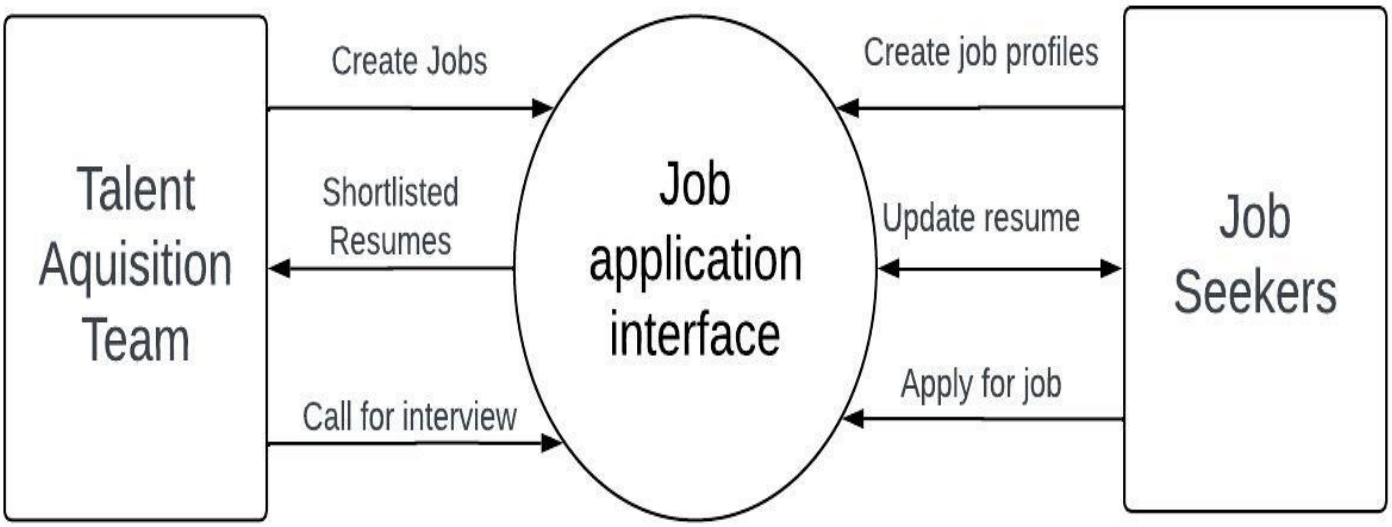
- **Resume Parsing:** The system can extract relevant information from resumes, such as contact details, work experience, education, and skills, and store it in a structured format for easy searching and filtering.
- **Application Screening and Ranking:** The system can automatically screen and rank incoming applications based on predefined criteria, such as qualifications, experience, and keywords.

The above application, however, uses Data Science and modern calculations formulas for data screening. Thus, the presented project uses the major takeaways of this system namely resume parsing via string comparisons and ranking using concrete, fixed percentages for various fields. The major reference taken is from the popular ATS software developed by Zoho Inc.

Reference: https://www.zoho.com/recruit/applicant-tracking-system.html

DATA FLOW DIAGRAM

LEVEL-0



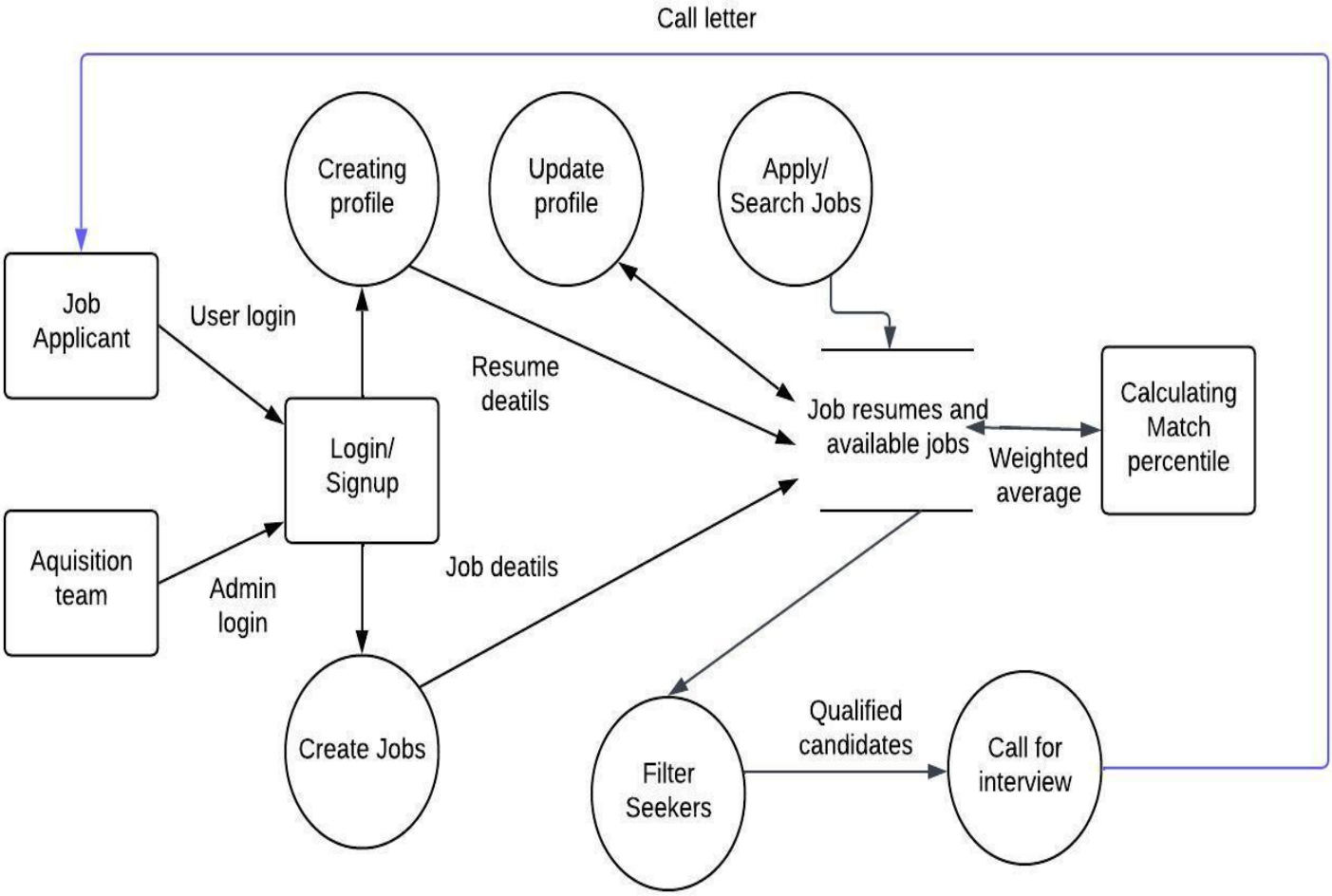
PART OF JOB SEEKER:

- To create job profile, update his/her resume and apply for the desired job after the computation.

PART OF TALENT ACQUISITION TEAM:

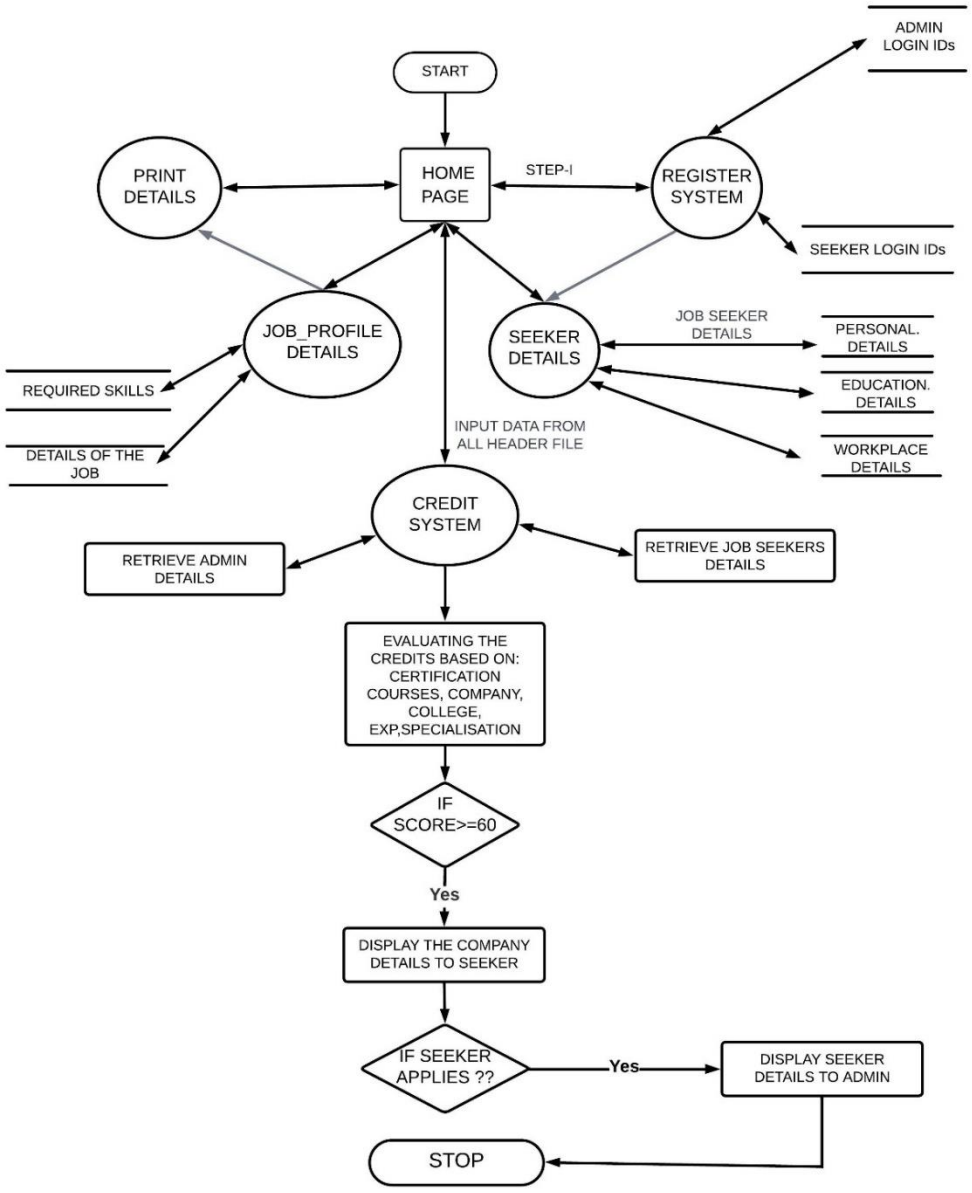
- To create the job profiles, and after the computation are displayed with the job seeker's details from where they can choose the desired ones to call for an interview.

LEVEL-1



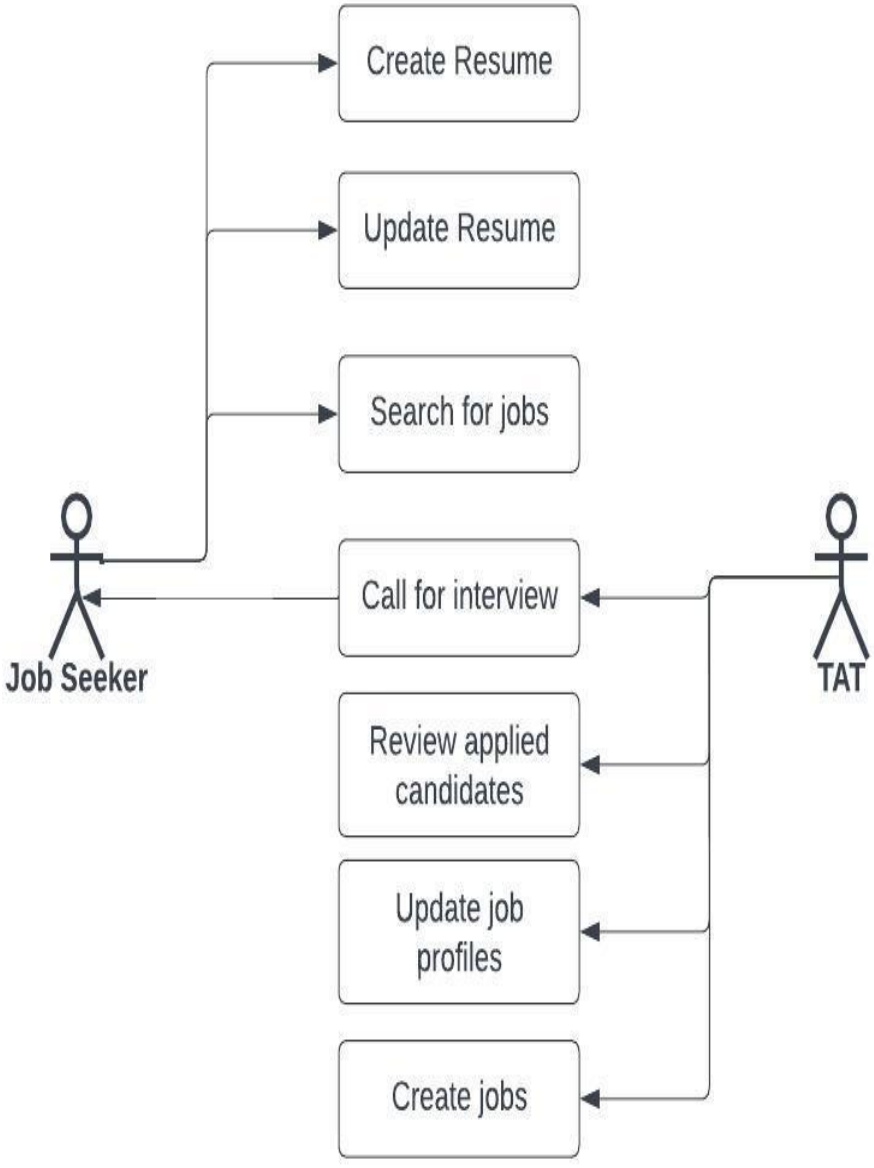
The job seeker and TAT team are required to sign up or login in the first step after which the job seeker will give in the details and TAT team will push in the details from their side about the job if they are signing up for the first time. After this when job seeker logs into his details and compared with the existing TAT details. The computation of match percentage is carried out and if it exceeds 60 % then he is displayed with the job details from where he can apply for the desired jobs.

LEVEL-2

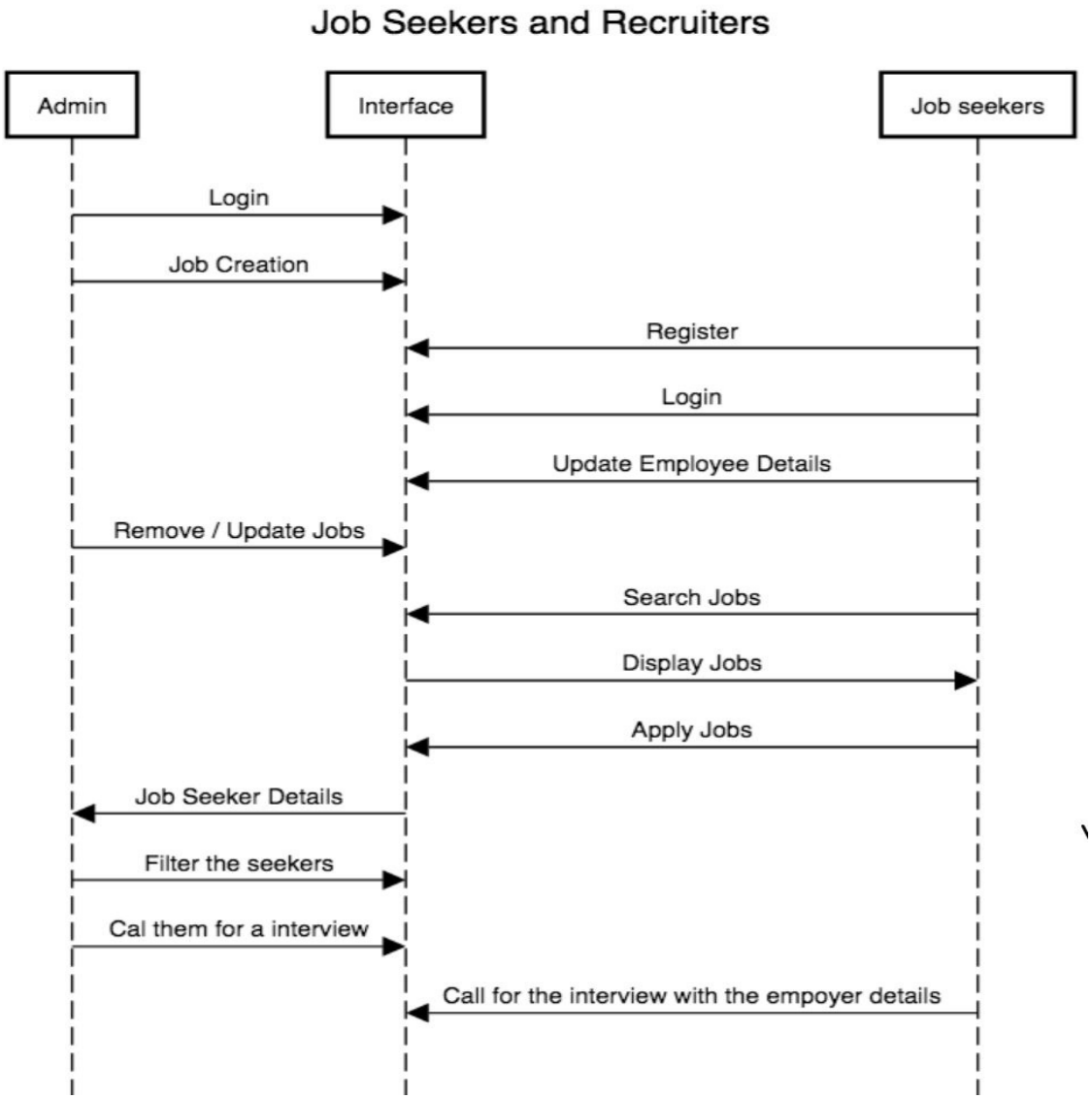


The personal, education and past experience details are stored in separate file. Similarly, Job Profile details (Company Details) and requirements are stored in respective file to maintain efficient data storage. In Credit calculation, these files are retrieved again, and details are stored in a separate file in the format <Job_Seeker_username>, <Job_Profile_username>, and <credit_score>. The job seeker is given the choice of selecting the companies for applying interview and these selected companies are stores up in another file. The selected candidate's details are shown to TAT, who can select the candidates for interview.

USE CASE DIAGRAM



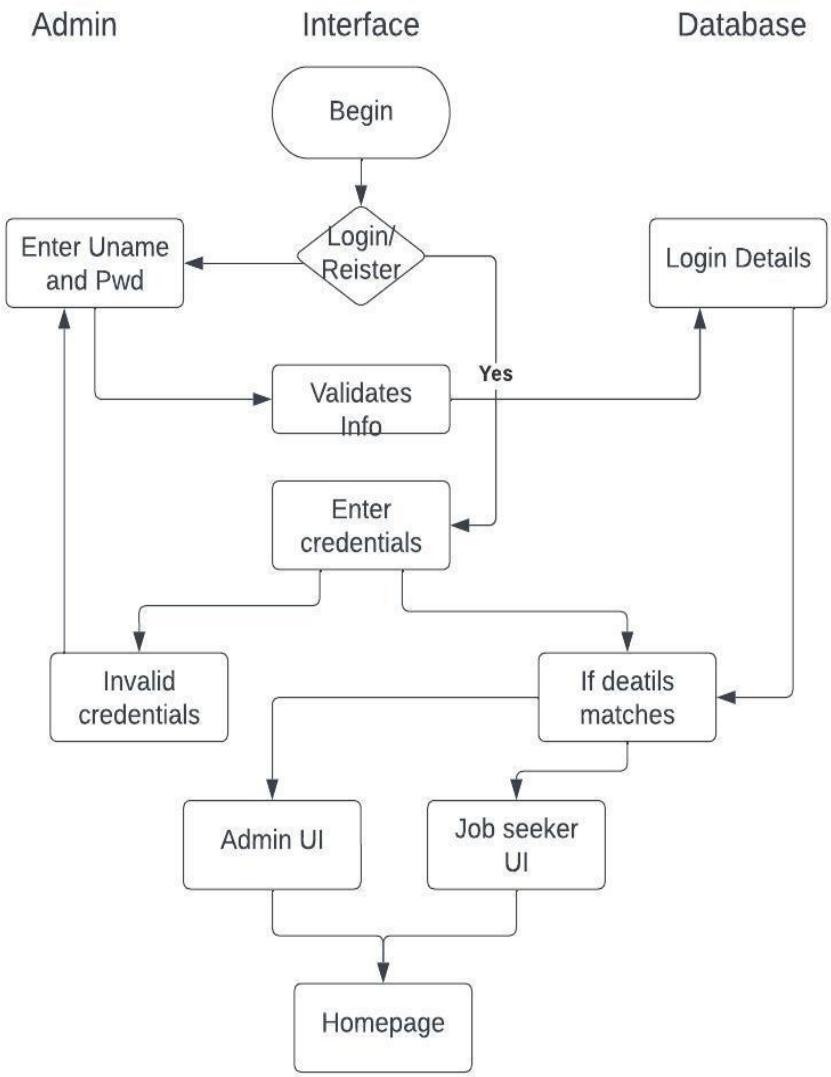
SEQUENCE DIAGRAM



The sequence diagram consists of mainly two components, (i.e.), Admin and Job Seekers. This sequence starts with the job seeker creating an account and providing his data and then admin logging in and pushing in the details of the job position like the company name, vacant position, expected salary and other details. This is followed by the comparison of job seeker's details with each of the admin's data and computing a percentage match between them.

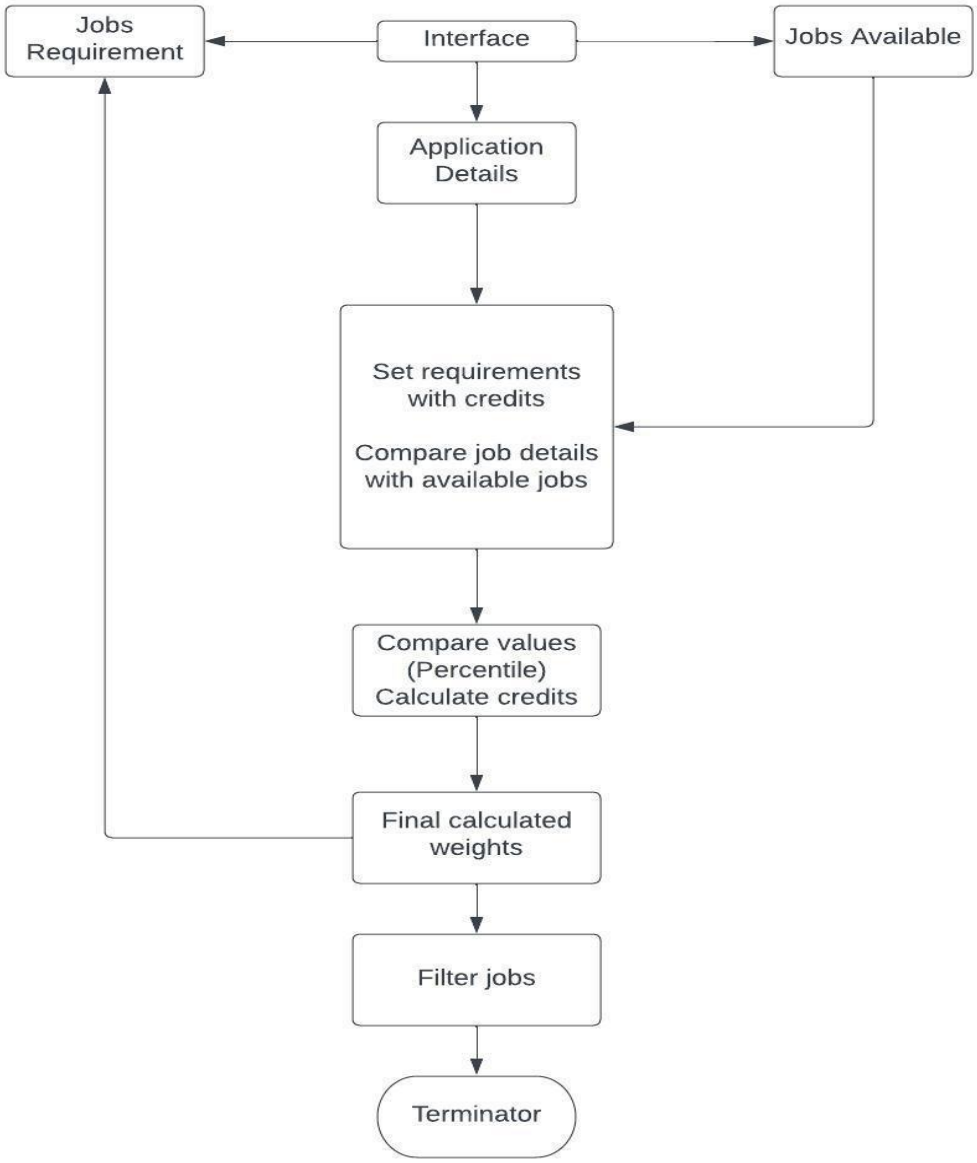
ACTIVITY DIAGRAM

LOGIN



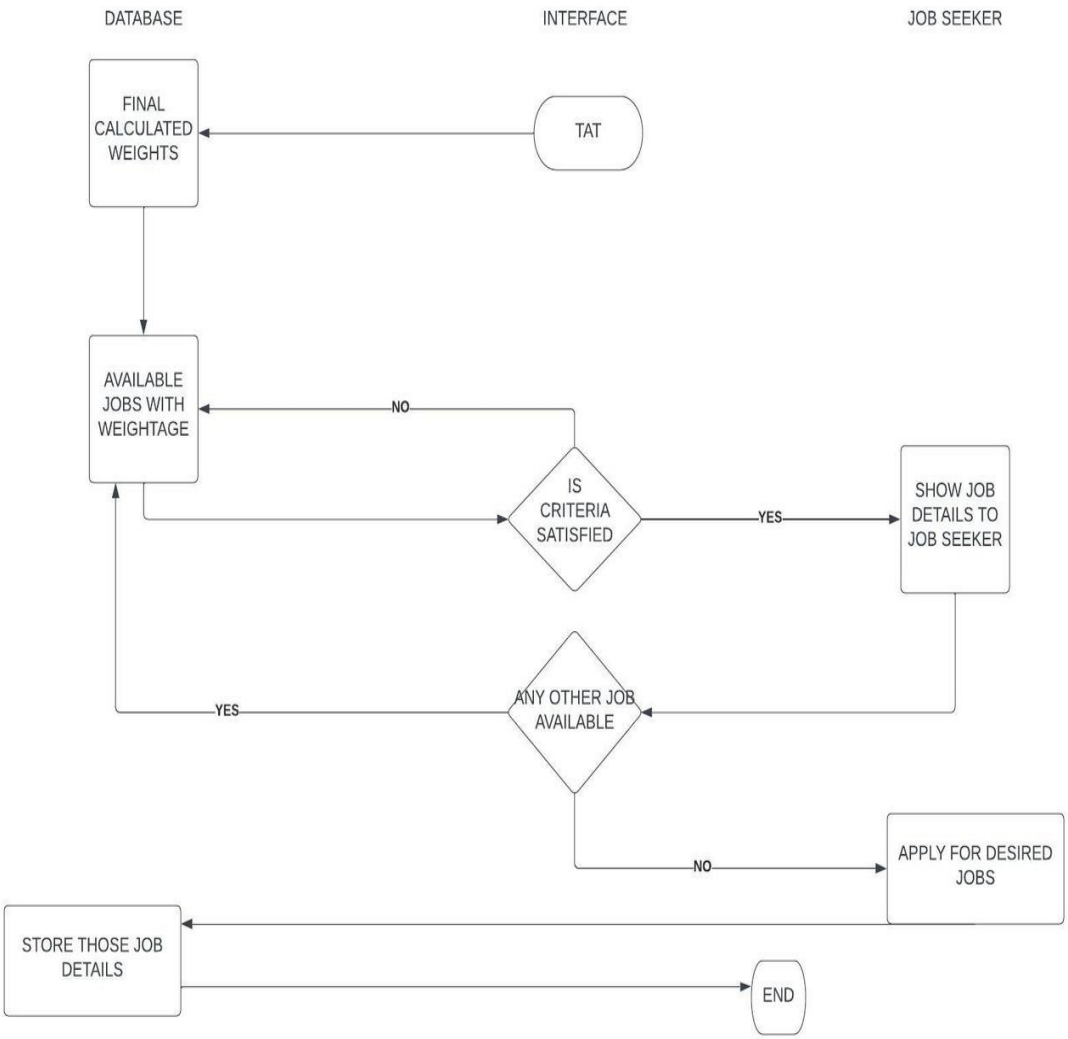
This is the activity diagram for the login module. This takes into consideration the main factor if the job seeker or admin already has an existing account or else they are expected to create/sign up a new account. After this when the job seeker or the admin is trying to login using wrong credentials he/ she is asked to re-enter the credentials.

CREDIT SYSTEM



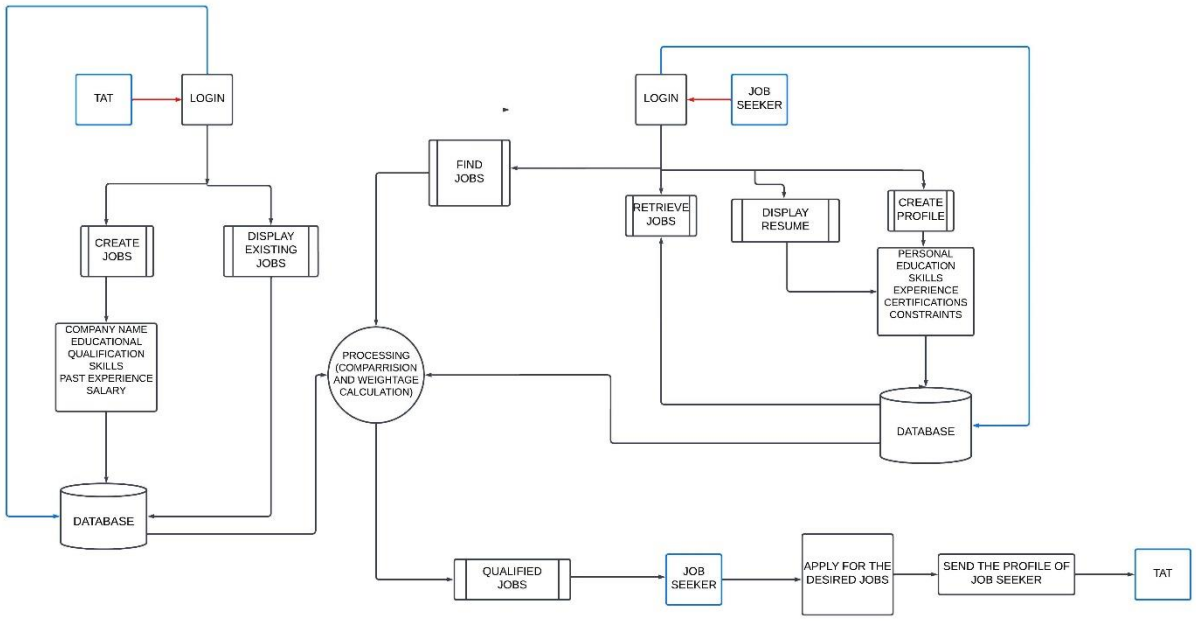
This activity diagram for credit system explains about the computation of weights and calculation of final match percentage between the user(job seeker) and corresponding talent acquisition teams.

JOB FILTERATION



In this section, the final calculated weights are used, and if the value is greater than 60 percentage between the job seeker and an admin team then those job details are displayed to the job seeker. After the job seeker applies for the desired jobs, then the admin is shown with the details of that job seeker when the admin is logged in.

STRUCTURE CHART



First ,Job seeker need to login or signup (for the first time). If the user is new ,the user needs to fill all the details i.e, personal, education, workplace details. If the user can able to see their resume by giving the choice to display the job seeker details . If the user wish to search jobs , the job seeker's detail is compared with the job applications and display the best suitable jobs as recommendations through the credit system.

In the admin side, the basic steps are same, but after signup, job necessary details are taken in as input. After saving it into a file, the admin also have the choice to print the created profiles.

In the final process, the qualified candidate’s details are shown to the TAT and the admin has the choice to select the candidates based on their printed profile, to call for interview.

DESCRIPTION OF EACH MODULE

We have used several modules in our code which we have integrated in main code of our program. The following are the modules that we have used in our program:

1. REGISTER MODULE:

In this module, we have allowed both the job seeker and Talent Acquisition Team to create new account using sign up or login to their respective accounts created already.

2. JOB SEEKER'S MODULE:

In this module, job seeker can create a new account where he is asked to provide his personal, educational and previous workplace details as input .

3. TALENT ACQUISITION TEAM'S MODULE:

In this module, Talent Acquisition Team is asked to enter the details about their company and about the job vacancy position and the minimum required skills for a candidate.

4. PRINT DETAILS MODULE:

In this module, the job seeker's module and the Talent Acquisition Team's details are displayed if they are asked to.

5. CREDIT CALCULATION MODULE:

In this module, the credit score for the job seeker compared with each job is calculated, and is displayed if the credit score is greater than or equal to 60 percent. Then he can apply for the desired jobs.

6. MAIN MODULE:

In this module, all the other modules are integrated into and the corresponding functions are called. Here the Talent Acquisition Team can select the desired candidates for interview.

IMPLEMENTATION

The data that is collected from the seeker and talent acquisition team is stored in separate files using structures to each operation and also the data from file is retrieved and stored in structure variables while displaying the profiles and also while calculating the credit score.

Here structures, with array variables and array of pointers are used because of their ease to collect and store the data in the files and also to retrieve them from the corresponding files, using file pointers and several read functions.

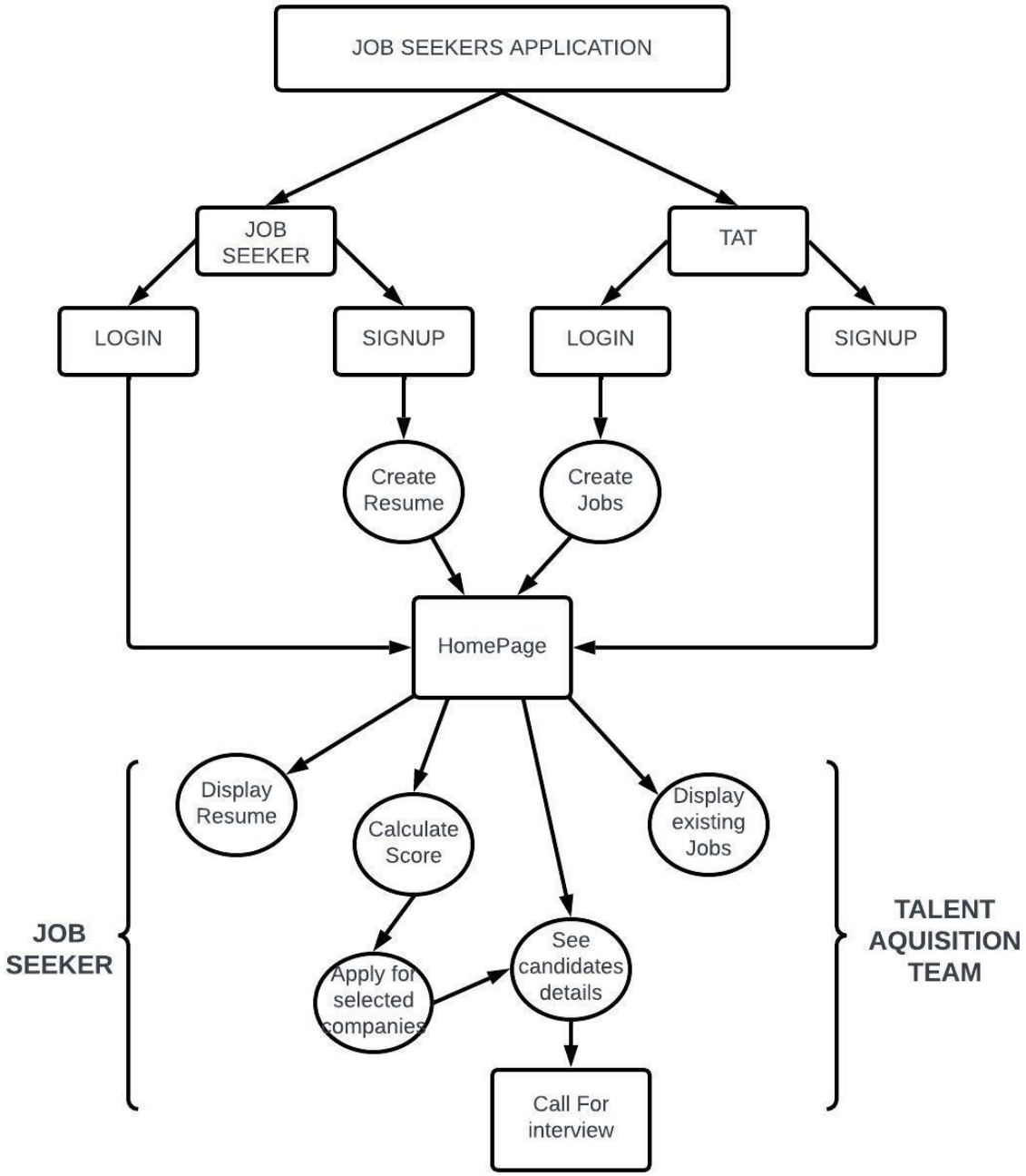
In our code, we have used 10 files, the following are the files and their functions:

1. **Admin.txt** - Stores the username, password and special password details of the talent acquisition team.
2. **Clg.txt -** Stores the top 30 colleges in Tamilnadu and also IITs, NITs and IIITs.
3. **Comp_compare.txt -**Stores the key details of job for Talent Acquisition Team.
4. **Comp_print.txt** -Contains the basic details about the job.
5. **Education.txt** -Stores the educational details os the job seeker.
6. **JobMatch.txt-** Stores the details for matching the seeker's username and the applied company's username.
7. **Match_Companies.txt** – Contains the corresponding credit scores of each seeker with respect to the company
8. **Personal.txt** -Stores the persoanl details of the job seeker.
9. **User.txt** -Stores the username and password- login details of job seeker.

10. **Workplace.txt -** Stores the previous work history details of job seeker.

Also in our code, we have used many string functions and standard functions like strcmp, used to compare two strings if they are equal, in many cases.

For our application, we have used both Repl.it and VS Code for our working and implementation of our code.



SAMPLE CODING

MAIN .C

```
 #include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "credit.h"
#include "job_profile.h"
#include "print_details.h"
#include "register.h"
#include "sekp1.h"

typedef struct { // for getting details
  char jobsekp[100];
  char jobprofile[100];
} job_seekers;

int main(void) {

  int cl[5];
```

```
  char uname[30];
  char cname[30];
  FILE *clg;
  int r1 = 0;
  int r2 = 0;
  int opt;
  printf("\n==================== WELCOME TO
====================\n");
  printf("\n===============JOB SEEKING
APPLICATION===============\n");
  Register(cl, uname);//login system

  if (cl[3] == 1) {    //if the login is valid
    if (cl[1] == 2) //&&(cl[2]==2)
    {
      while (1) { //menu driven for job seeker
        r1 = cl[1];
        r2 = cl[2];

printf("\n=============================================\
n");
        printf("\nWelcome %s", uname);
        printf("\n1.Create profile");
        printf("\n2. Display existing profile");
        printf("\n3. Search for jobs");
```

```
        printf("\n4. Quit");
        printf("\nEnter your choice : ");
        scanf("%d", &opt);

printf("\n==========================================\
n");

        if (opt == 1) {    //entering details seeker
          sekp(r1, r2, uname, cname);
        } else if (opt == 2) { //display the details
          printing(r1, uname);
        } else if (opt == 3) {
          credit_calc(uname);    //credit calculation to recommend job
        } else if (opt == 4) { // want to quit
          break;
        } else {
          break;
        }
      }
    }

    if (cl[1] == 1) {     //menu driven for Admin
      while (1) {
        r1 = cl[1];
```

```
        r2 = cl[2];
        printf("\n================== HOMEPAGE
==================\n");

printf("\n=========================================+=
==\n");
        printf("\nWelcome %s", uname);
        printf("\n1. Create profile");
        printf("\n2. Display existing jobs");
        printf("\n3. Search for candidates");
        printf("\n4. Quit");
        printf("\nEnter your choice : ");
        scanf("%d", &opt);

printf("\n=========================================\
n");

        /*if ){
        printf("Want to create a Job application (Y or N): ");
        char choice;
        scanf("\n%c",&choice);*/
        if (opt == 1) {    //posting jobs
          job_profile(uname);
        }
```

```
else if (opt == 2) {    //display the job details
  printing(r1, uname);

} else if (opt == 3) {
  FILE *candidates = fopen("JobMatch.txt", "r");
  FILE *seeker_personal_retrieval = fopen("personal.txt", "r");
  FILE *seeker_education_retrieval = fopen("education.txt", "r");
  FILE *seeker_work_retrieval = fopen("workplace.txt", "r");

  struct personal personal_retrieval;
  struct education education_retrieval;
  struct sign_up username_retrieval;

  job_seekers job_seek;
  while (!feof(candidates)) {
    fscanf(candidates, "%[^,],%[^,],\n", job_seek.jobsekp,
          job_seek.jobprofile);

    rewind(seeker_personal_retrieval);
    rewind(seeker_education_retrieval);

    while (!feof(seeker_personal_retrieval)) {
      fscanf(seeker_personal_retrieval,
            "%[^,],%[^,],%[^,],%d,%[^,],%[^,],%d,%[^,],%d,%[^,],\n",
```

```
            username_retrieval.user, personal_retrieval.name,
            personal_retrieval.gender, &personal_retrieval.age,
            personal_retrieval.father, personal_retrieval.mother,
            &personal_retrieval.DOB, personal_retrieval.address,
            &personal_retrieval.ph_no, personal_retrieval.email_id);

        fscanf(seeker_education_retrieval,
            "%[^,],%[^,],%d,%[^,],%[^,],%f,%[^,],%[^,],%[^,],\n",
            username_retrieval.user, education_retrieval.school_name,
            &education_retrieval.mark, education_retrieval.college,
            education_retrieval.degree, &education_retrieval.cgpa,
            education_retrieval.course_name,
            education_retrieval.specializtion,
            education_retrieval.certification);
        if (strcmp(uname, job_seek.jobprofile) == 0 &&
           (strcmp(job_seek.jobsekp, username_retrieval.user) == 0)) {
          printf("\n--------------CANDIDATE DETAILS-----------------
\n");
          printf("\nCandidate Name: %s", personal_retrieval.name);
          printf("\nCandidate College: %s", education_retrieval.college);
          printf("\nCandidate CGPA: %f ", education_retrieval.cgpa);
          printf("\nCandidate Specialization: %s",
               education_retrieval.specializtion);
          printf("\n-----------------------------------------------\n");
```

```
            printf("\n%s wishes to apply for interview: ",
                  personal_retrieval.name);
            printf("\nAccept Candidate (YES/NO): ");
            printf("\n-----------------------------------------\n");
            char select[100];
            scanf("\n%[^\n]", select);
            if (strcmp(select, "YES") == 0) {
              printf("\nSucessfull....Called candidate for interview");
              printf("\n-----------------------------------------\n");
            }
          }
        }
      }
      // credit_calc(uname);
    } else if (opt == 4) {    // want to quit
      break;
    } else {
      break;
    }
  }
}
```

```
  // clg=fopen("clg.txt","r");
}
```

CREDIT SYSTEM(CREDIT.H)

```
#include "job_profile.h"
#include "register.h"
#include "sekp1.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void credit_calc(char uname[]) {
  FILE *fp_work, *edu1, *work1, *jobs1, *cllg, *fp_print, *matches,
*job_match;
  cllg = fopen("clg.txt", "r");
  edu1 = fopen("education.txt", "r");
  work1 = fopen("workplace.txt", "r");
  fp_work = fopen("comp_compare.txt", "r");
  fp_print = fopen("comp_print.txt", "r");
  matches = fopen("Matched_Companies.txt", "a+");
  job_match = fopen("JobMatch.txt", "a+");

  struct personal record_1;
  struct education record_2;
```

```
  struct workplace record_3;
  struct sign_up username_company, username;
  Company comp_intro_print;
  // int choice = r1;
  // int ls = r2;
  char deg1[MAX_ARRAY_LENGTH];
  Skills key_skills1;
  Requirement other1;
  char *certi_arr[100]; // Assuming a maximum of 100 words
  char *skill_arr[100];
  char college_check[200];

  int credit_score = 0;

  while (!feof(edu1)) {
// To retrieve the details of job seeker from the necessary //files using fcanf
    fscanf(edu1, "%[^,],%[^,],%d,%[^,],%[^,],%f,%[^,],%[^,],%[^,],\n",
          username.user, record_2.school_name, &record_2.mark,
          record_2.college, record_2.degree, &record_2.cgpa,
          record_2.course_name, record_2.specializtion,
          record_2.certification);
//Storing workplace details
    fscanf(work1, "%[^,],%[^,],%[^,],%[^,],%d,\n", username.user,
record_3.work,
```

```
        record_3.pastwork, record_3.wdesig, &record_3.wexp);

    char *token = strtok(record_2.certification, "-"); // Split by spaces
    int i = 0;
    while (token != NULL) {
      certi_arr[i] = token;
      i++;
      token = strtok(NULL, "-");
    }

    if (strcmp(username.user, uname) == 0)

    {

      while (!feof(fp_work)) {
        //Retrieving the required details of the company from the file
        credit_score = 0;
        fscanf(fp_print, "%[^,],%[^,],%[^,],%[^,],\n",
username_company.user,
            comp_intro_print.company_name, comp_intro_print.designation,
            comp_intro_print.details);
        fscanf(fp_work, "%[^,],%[^,],%d,%[^,],%d,%[^,],%d,%[^,],\n", deg1,
            other1.field_of_expertise, &key_skills1.num_skills,
            key_skills1.skills, &other1.experience, other1.shift,
```

```
        &other1.salaryRange, other1.employmentType);

    // Assuming a maximum of 100 words

    char *token1 = strtok(key_skills1.skills, "-"); // Split by spaces

    int j = 0;
    while (token1 != NULL) {
      skill_arr[j] = token1;
      j++;
      token1 = strtok(NULL, "-");
    }
    // Credit score calculation based on certification skills.
    for (int x = 0; x < key_skills1.num_skills; x++) {
      if (strcmp(skill_arr[x], certi_arr[1]) == 0) {
        credit_score += 10;
      }
    }

    int check_variable = 0;
    // Credit score calculation based on certification company
    char companies_courses[100][100] = {"Google", "IBM",
"Microsoft",
                                        "NPTEL",  "MIT", "UCA"};
```

```
for (int v = 0; v < 6; v++) {
  if (strcmp(companies_courses[v], certi_arr[0]) == 0) {
    credit_score += 10;
    check_variable = 1;
    break;
  }
}
if ((check_variable == 0) &&
    (strcmp(record_2.certification, "NIL") != 0)) {
  credit_score += 5;
}
//Declaring top companies for calculationg part of credit system for certification company
char top_companies[100][100] = {"Meta",    "Apple",       "Amazon",
                                "Netflix", "Google",      "HCL",
                                "IBM",     "GoldmanSachs"};

if (strcmp(record_3.work, "YES") == 0) {

  credit_score += 7;

  for (int v = 0; v < 8; v++) {
    if (strcmp(top_companies[v], record_3.pastwork) == 0) {
      credit_score += 5;
```

```
    }
  }
 // Credit score calculation based on work experience
  if (record_3.wexp >= other1.experience) {
    credit_score += 8;
  } else {
    credit_score += 4;
  }
 }

 int check = 1;
//credsit system calculation based on college where job seeker had studied.
 while (!feof(cllg)) {
  fscanf(cllg, "%199[^\n]\n", college_check);
  if (strcmp(record_2.college, college_check) == 0) {
    break;
  }
  check++;
 }

 if (check <= 5) {
  credit_score += 20;
 } else if (check <= 10) {
```

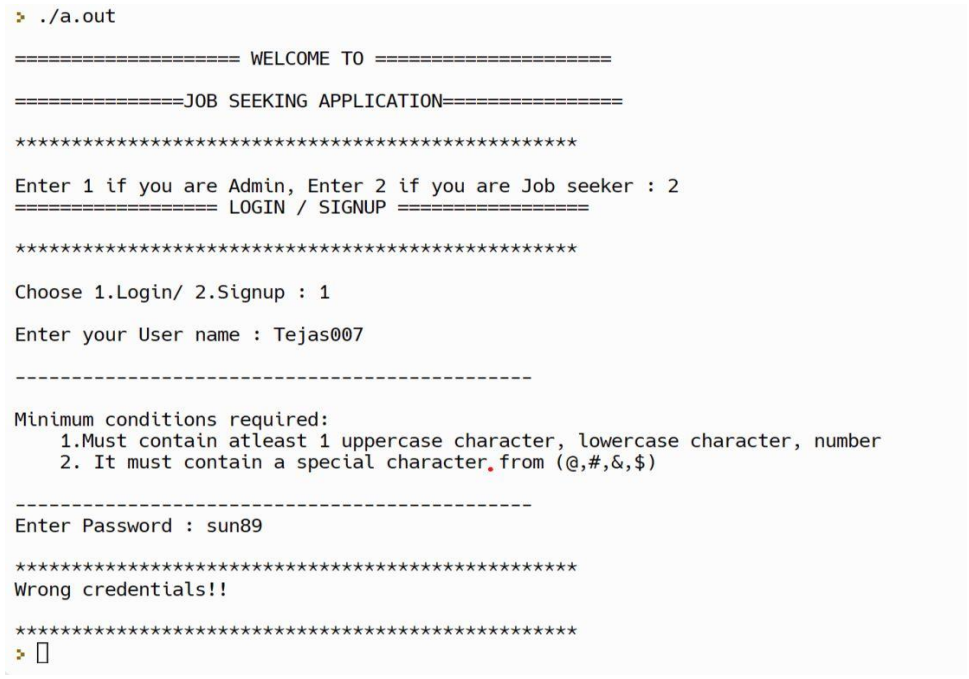
```
  credit_score += 16;
} else if (check <= 15) {
  credit_score += 12;
} else if (check <= 20) {
  credit_score += 8;
} else if (check <= 25) {
  credit_score += 4;
}
rewind(cllg);


if (strcmp(record_2.specializtion, other1.field_of_expertise) == 0) {
  credit_score += 15;
}
//Credit score based on cgpa of the job seeker

if (record_2.cgpa > 8.0) {
  credit_score += 10;
}

if (credit_score > 50) {
  printf("\n-----------------------------------------\n");
  printf("\nCompany Name: %s", comp_intro_print.company_name);
  printf("\nJob Available: %s", comp_intro_print.designation);
```

```
        printf("\nDetails: %s", comp_intro_print.details);
        printf("\nJob Match Percentage: %d", credit_score);
        fprintf(matches, "%s,%s,%d,%d,\n", username.user,
                username_company.user, credit_score, 0);
       }
      }
      printf("\n-----------------------------------------\n");
      break;
    }
  }
  char job[100];
  char *job1[100];
  printf(
      "\nEnter the company names that you wish to apply for separated by - :
");
  printf("\n-----------------------------------------\n");
  scanf("\n%[^\n]", job);
  char *token1 = strtok(job, "-"); // Split by spaces
  int apply = 0;
  while (token1 != NULL) {
    job1[apply] = token1;
    apply++;
    token1 = strtok(NULL, "-");
  }
```

```
  // Storing the job seeker username and the company's username in a file in each line .
  printf("%s %s", job1[0], job1[1]);
  for (int e = 0; e < apply; e++) {
    rewind(fp_print);
    while (!feof(fp_print)) {
      fscanf(fp_print, "%[^,],%[^,],%[^,],%[^,],\n", username_company.user,
          comp_intro_print.company_name, comp_intro_print.designation,
          comp_intro_print.details);
      if (strcmp(job1[e], comp_intro_print.company_name) == 0) {
        fprintf(job_match, "%s,%s,\n", username.user, username_company.user);
        printf("\nSent Profile to Company %s !", comp_intro_print.company_name);
        printf("\n-----------------------------------------\n");
        break;
      }
    }
  }
  //Closing all the opened files using fclose
  fclose(cllg);
  fclose(edu1);
  fclose(work1);
```

```
  fclose(fp_work);
  fclose(fp_print);
  fclose(matches);
  fclose(job_match);
}
```

SAMPLE OUTPUT



Here, in the first photo, correct login credentials are given and the seeker is successfully logged in. Whereas in the second photo, log in is not allowed when wrong credentials is given.

In these two photos, the Talent Acquisition Team is logging in the application and the details of the company is displayed as requested by the Talent Acquisition User.

```
===============JOB SEEKING APPLICATION================

************************************************

Enter 1 if you are Admin, Enter 2 if you are Job seeker : 2
================== LOGIN / SIGNUP ==================

************************************************

Choose 1.Login/ 2.Signup : 1

Enter your User name : Tejas007

------------------------------------------------

Minimum conditions required:
    1.Must contain atleast 1 uppercase character, lowercase character, number
    2. It must contain a special character from (@,#,&,$)

------------------------------------------------
Enter Password : Tj#72

************************************************
Logged in

************************************************
```

```
================================================

Welcome Tejas007
1.Create profile
2. Display existing profile
3. Search for jobs
4. Quit
Enter your choice : 3

================================================

------------------------------------------------

Company Name: Apple Inc
Job Available: Software intern
Details: Diligant hardworking candidate
Job Match Percentage: 90
------------------------------------------------

Company Name: Macrohard Ltd
Job Available: Debugging Analyst
Details: Adept coding knowlegde agil candidate required
Job Match Percentage: 65
------------------------------------------------

Enter the company names that you wish to apply for separated by - :
------------------------------------------------
Macrohard Ltd-Apple Inc
Macrohard Ltd Apple Inc
Sent Profile to Company Macrohard Ltd !
------------------------------------------------

Sent Profile to Company Apple Inc !
------------------------------------------------
```

```
================================================

Welcome Tejas007
1.Create profile
2. Display existing profile
3. Search for jobs
4. Quit
Enter your choice : 4

================================================
>
```

Here, the job seeker is logging in and is searching for jobs, where the job seeker is matched with two jobs under the minimum match percentage of 60 percent, and he is allowed to apply for the desired jobs and hence his profile is sent to the company he has applied for.

```
==========JOB SEEKING APPLICATION==========

**********************************************

Enter 1 if you are Admin, Enter 2 if you are Job seeker : 1
================= LOGIN / SIGNUP =================

**********************************************

Choose 1.Login/ 2.Signup : 1

Enter your User name : Apl15

----------------------------------------------

Minimum conditions required:
    1.Must contain atleast 1 uppercase character, lowercase character, number
    2. It must contain a special character from (@,#,&,$)

----------------------------------------------
Enter Password : macOS#10

**********************************************
Enter special pwd:777

**********************************************
Logged in

**********************************************
```

```
================== HOMEPAGE ==================

==============================================+===

Welcome Apl15
1. Create profile
2. Display existing jobs
3. Search for candidates
4. Quit
Enter your choice : 3

==============================================

--------------CANDIDATE DETAILS-----------------

Candidate Name: Tejas Srikanth
Candidate College: Birla Institute of Technology and Science
Candidate CGPA: 8.532000
Candidate Specialization: Augumented Reality
----------------------------------------------

Tejas Srikanth wishes to apply for interview:
Accept Candidate (YES/NO):
----------------------------------------------
YES

Sucessfull....Called candidate for interview
----------------------------------------------

================== HOMEPAGE ==================

==============================================+===

Welcome Apl15
1. Create profile
2. Display existing jobs
3. Search for candidates
4. Quit
Enter your choice : 4

==============================================
>
```

In these photos, the Talent Acquisition Team has logged in and when searching for candidates, he is displayed with the job seeker's details who have applied for that particular job, and he can call the desired candidates for interview.

LIMITATIONS

Non - Flexible data: Since major fields such as degree, specialization is entered via menu list, the inputs aren't flexible for the user, and must choose from given list. This also occurs in college list.

Rigid Credit System: Credits for each field fixed by the acquisition team affects the final score linearly and isn't intuitive. Grading is not relative.

Inefficient algorithm: The final weighted average score is based on simple formulas, which aren't accurate like the present systems such as Linear Regression which involves Data science.

Improper Data Storage: Our system involves storing details as a file and is based on indexing rather than databases, which are highly organized and user-friendly.

REAL – LIFE OBSERVATIONS

Societal Perspective: The system, even though the process isn't transparent, it ensures a fair and unbiased result, based purely on merit and skills. A direct connection ensures fair job matching.

Legal Perspective: All job profiles are secured and not available for public. Only administrators can view the profiles (Only the qualified ones). This ensures privacy and security of details. 2-way authentication system strengthens this idea.

Environmental Perspective: Getting only required details ensures a streamlined data which is efficiently stored in a file. This ensures their is no garbage data.

A user-friendly system makes the application process and talent aquisition process hassle free and intuitive.

LEARNING OUTCOMES

This project has helped in understanding the job applying environment and how the software works. It gives an idea about the work environment and the important fields for each job.

It has helped in understanding the key concepts of file handling, structure usage, pointers and how arrays works. It also helped us to grasp the intricacies of C programming and various caveats.

REFERENCES

- https://www.zoho.com/recruit/applicant-tracking-system.html
- https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=1398&context=jitim
- https://www.geeksforgeeks.org/strtok-strtok_r-functions-c-examples/
- https://www.naukri.com