|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A mediating Platform for Job seekers and Talent acquisition Managers** | | | | | | | |
| UCS2201 – Fundamentals and Practice of Software Development  A PROJECT REPORT  Submitted By  V.HARISH KUMAR\_3122 22 5001 037  R. HEMALATHA\_3122 22 5001 039  G.A. INIYADHARSHINI\_3122 22 5001 041  A blue and white logo  Description automatically generated  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 “A mediating platform for jobseekers and Talent acquisitions Managers” is the bonafide work of “V.HARISH KUMAR (3122 22 5001 037), R.HEMALATHA (3122 22 5001 039) and G.A.INIYADHARSHINI(3122 22 5001 041)” 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**   |  |  | | --- | --- | | **S.NO** | **Content** | | 1. | Problem Statement and Objective | | 2. | Diagrams (use case, activity, sequence) | | 3. | Flow chart for Each module | | 4. | Description for each module | | 5. | Source Code | | 6. | Test Cases | | 7. | Limitations |   **Problem Statement/Objectives:**  **To create a mediating platform through which,**  i. Job seekers can explore the open positions which match their profiles.  ii. Talent acquisition managers can search for deserving applicants who could qualify for their job requirements.    **Abstract**  1. Retrieve Job Seeker Profile:  - Retrieve the profile of a job seeker from the database, including their skills,  experience, education, and other relevant information.  2. Retrieve Open Job Positions:  - Retrieve the details of open job positions from the database, including job  requirements, skills needed, and other relevant criteria.  3. Calculate Job Seeker Score:  - Compare the job seeker profile with the requirements of each open position.  - Assign a matching score or rank to the job seeker based on their compatibility with the job requirements.  - Consider factors such as skills, experience, education, and any specific preferences or  constraints defined by the job seeker.  4. Sort and Filter Job Seekers:  - Sort the job seekers based on their matching scores, from highest to lowest.  - Apply additional filters, if provided, such as location, job type, or salary range, to narrow down the list of potential candidates.  5. Display Job Seeker Results:  - Present the sorted and filtered list of job seekers to the talent acquisition manager.  - Display relevant details from each job seeker&profile, such as skills, experience, education, and any additional information provided.  6. Interaction and Selection:  - Enable the talent acquisition manager to interact with the platform and review the job seekers profiles.  - Provide options for the manager to communicate with selected candidates, schedule interviews, or take any further action.  7. Repeat for Additional Job Positions:  - Repeat the process for each open job position, retrieving the relevant details and matching job seekers accordingly.  - Update the displayed results for each position to reflect the most suitable candidates. | |
| Use Case Diagram | | | | | |
| Job Seekers Activity Diagram | | |
| HR Manager Activity Diagram    Sequence Diagram | | | | |
| Module1: Login | | | | | |
| Module2: Search jobs  A diagram of a work flow  Description automatically generated | | | | | | |
| Module3: Search jobseekers  A diagram of a job application  Description automatically generated  **Description for each module:**    **1. Signup as HR Manager:**  This module allows HR Managers to sign up by providing their username and password. The entered credentials are stored in the "Admins.csv" file, which maintains the list of registered HR Managers.  **2. Log in as HR Manager:**  HR Managers can log in using their previously registered credentials. The system checks the input username and password against the records in the "Admins.csv" file. If the login is successful, the HR Manager gains access to various administrative functionalities.  **3. Add Job:**  HR Managers can use this module to add new job listings to the system. They are prompted to input the necessary details, such as the company name, job title, required qualification, experience, salary, etc. The new job is added to the "jobs.csv" file, which maintains a list of available job openings.  **4. Edit the Job Details:**  HR Managers can use this module to edit the details of an existing job listing. They are asked to enter the company name they want to edit. If the company name exists in the system, the HR Manager can modify the job's title, qualification, experience, and salary. The changes are updated in the "jobs.csv" file.  **5. Search Job Seeker:**  This module enables HR Managers to search for job seekers based on specific criteria. The HR Manager can input the job name and the desired recommendation percentage. The system then searches the "Users.csv" file, which contains information about registered job seekers, to find matches based on the given criteria.  **6. Signup for Job Seekers:**  Job seekers can sign up for the platform using this module. They are required to provide their full name, email, contact number, password, educational qualification, the job they are looking for, and their desired recommendation percentage. The job seeker's details are added to the "Users.csv" file.  **7. Login for Job Seekers:**  Registered job seekers can log in using their username and password. The system checks the provided credentials against the records in the "Users.csv" file. Upon successful login, the job seeker gains access to further functionalities.  **8. Edit Details:**  This module allows job seekers to edit their personal details, such as full name, email, and contact number. After entering their username and password, they can modify the corresponding fields, and the changes are updated in the "Users.csv" file.  **9. Search Jobs:**  Job seekers can use this module to search for jobs based on specific criteria. They enter the desired job title, years of experience, expected salary, and whether they are available for night shifts. The system then searches the "jobs.csv" file for job openings that match the given criteria.  **10. Match the Profile:**  This module provides a recommendation system for job seekers. It calculates a recommendation percentage based on the matching criteria, such as job title, salary, experience, etc. The system then displays job openings with recommendation percentages greater than or equal to 80%.  **11. Apply for Jobs:**  After finding a suitable job through the search process, job seekers can use this module to apply for the job. They provide their 10th and 12th marks, information about any arrears, languages known, and additional skills. The application is then submitted for further process.  **Source code:**    **Header files:**    #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <windows.h>  #define MAX\_USERS 100  #define MAX\_FIELD\_SIZE 100  #define MAX\_JOBS 100  #define CSV\_FILENAME "jobs.csv"  **Files:**  **Users.csv** (To store the details of the users)  **Jobs.csv** (To store the details of the jobs)  **Admins.csv** (To store the details of the HR managers as admin)    **Structures:**  **1. Struct user**  **2. Struct job**  **3. Struct admin**    struct user  {  char fullname[MAX\_FIELD\_SIZE];  char email[MAX\_FIELD\_SIZE];  char password[MAX\_FIELD\_SIZE];  char username[MAX\_FIELD\_SIZE];  char phone[MAX\_FIELD\_SIZE];  char job[MAX\_FIELD\_SIZE];  char qualification[MAX\_FIELD\_SIZE];  int recommendationPercentage;  };  struct Job  {  char name[MAX\_FIELD\_SIZE];  char qualification[MAX\_FIELD\_SIZE];  char experience[MAX\_FIELD\_SIZE];  double salary;  char companyName[MAX\_FIELD\_SIZE];  char ch[3];  int nightShift;  char link[20];  char location[20];  };  struct Admin  {  char name[MAX\_FIELD\_SIZE];  char pword[MAX\_FIELD\_SIZE];  } admin1;  **Functions:**  1**. void takeInput(char ch[MAX\_FIELD\_SIZE])**  **2. void takePassword(char pwd[MAX\_FIELD\_SIZE])**  **3. void editDetails()**  **5. int login()**  **6. int adlogin()**  7.      void takeInput(char ch[MAX\_FIELD\_SIZE])  {  fgets(ch, MAX\_FIELD\_SIZE, stdin);  ch[strcspn(ch, "\n")] = '\0'; // Remove newline character  }  void takePassword(char pwd[MAX\_FIELD\_SIZE])  {  int i = 0;  char ch;  while (1)  {  ch = getch();  if (ch == '\r' || ch == '\n')  {  pwd[i] = '\0';  break;  }  else if (ch == '\b')  {  if (i > 0)  {  i--;  printf("\b \b");  }  }  else  {  pwd[i++] = ch;  printf("\*");  }  }  }  void generateUsername(char email[MAX\_FIELD\_SIZE], char username[MAX\_FIELD\_SIZE])  {  int i;  for (i = 0; i < strlen(email); i++)  {  if (email[i] == '@')  {  break;  }  else  {  username[i] = email[i];  }  }  username[i] = '\0';  }  int login()  {  char username[MAX\_FIELD\_SIZE], pword[MAX\_FIELD\_SIZE];  struct user person;  FILE \*fp;  printf("\nEnter the username: ");  takeInput(username);  printf("\nEnter your password: ");  takePassword(pword);  fp = fopen("Users.csv", "r");  if (fp == NULL)  {  printf("Error opening file!\n");  return 0;  }  while (fscanf(fp, "%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%d,%[^\n]\n",  person.fullname, person.email, person.password, person.username,  person.phone, person.job, &person.recommendationPercentage, person.qualification) == 8)  {  if (strcmp(person.username, username) == 0)  {  printf("\n\t\t\t\t\tWelcome %s", person.fullname);  printf("\n\n|Full Name :\t%s", person.fullname);  printf("\n|Email :\t%s", person.email);  printf("\n|Username :\t%s", person.username);  printf("\n|Contact number:\t%s\n", person.phone);  fclose(fp);  return 1;  }  }  fclose(fp);  return 0;  }  void editDetails()  {  char username[MAX\_FIELD\_SIZE], pword[MAX\_FIELD\_SIZE];  struct user person;  int found = 0;  FILE \*fp, \*temp;  fp = fopen("Users.csv", "r");  temp = fopen("temp.csv", "w");  if (fp == NULL || temp == NULL)  {  printf("Error opening file!\n");  return;  }  printf("\nEnter the username to edit details: ");  takeInput(username);  printf("\nEnter your password: ");  takePassword(pword);  while (fscanf(fp, "%[^,],%[^,],%[^,],%[^,],%[^\n]\n",  person.fullname, person.email, person.password, person.username, person.phone) == 5)  {  if ((strcmp(person.username, username) == 0) && (strcmp(person.password,pword)==0))  {  printf("\n\t\t\t\t\tWelcome %s", person.fullname);  printf("\n\n|Full Name :\t%s", person.fullname);  printf("\n|Email :\t%s", person.email);  printf("\n|Username :\t%s", person.username);  printf("\n|Contact number:\t%s\n", person.phone);  printf("\nEnter the new Full Name: ");  takeInput(person.fullname);  printf("\nEnter the new Email: ");  takeInput(person.email);  printf("\nEnter the new Contact number: ");  takeInput(person.phone);  fprintf(temp, "%s,%s,%s,%s,%s\n", person.fullname, person.email,  person.password, person.username, person.phone);  printf("\nDetails updated successfully!\n");  found = 1;  }  else  {  fprintf(temp, "%s,%s,%s,%s,%s\n", person.fullname, person.email,  person.password, person.username, person.phone);  }  }  fclose(fp);  fclose(temp);  if (found)  {  remove("Users.csv");  rename("temp.csv", "Users.csv");  }  else  {  remove("temp.csv");  printf("\nUser not found.\n");  }  }  void loadJobs(struct Job jobs[], int \*numJobs)  {  FILE \*file = fopen(CSV\_FILENAME, "r");  if (file == NULL)  {  printf("Error opening file!\n");  return;  }  char line[256];  while (fgets(line, sizeof(line), file))  {  struct Job job;  sscanf(line, "%[^,],%[^,],%[^,],%lf,%[^,],%[^,],%[^\n]", job.name, job.qualification,  job.experience, &job.salary, job.companyName,job.link,job.location);  jobs[\*numJobs] = job;  (\*numJobs)++;  }  fclose(file);  }  int adlogin()  {  char username[MAX\_FIELD\_SIZE], pword[MAX\_FIELD\_SIZE];  FILE \*fp;  printf("\nEnter the username: ");  takeInput(username);  printf("\nEnter your password: ");  takePassword(pword);  fp = fopen("Admins.csv", "r");  if (fp == NULL)  {  printf("Error opening file!\n");  return 0;  }  while (fscanf(fp, "%[^,],%[^\n]\n", admin1.name, admin1.pword) == 2)  {  if ((strcasecmp(admin1.name, username) == 0) && (strcmp(admin1.pword, pword) == 0))  {  fclose(fp);  return 1;  }  }  fclose(fp);  return 0;  }  void editJobDetails(struct Job jobs[], int numJobs)  {  int found = 0;  char searchCompany[MAX\_FIELD\_SIZE];  printf("\nEnter the company name to edit job details: ");  takeInput(searchCompany);  for (int i = 0; i < numJobs; i++)  {  if (strcmp(jobs[i].companyName, searchCompany) == 0)  {  printf("\nCurrent job details:");  printf("\nJob Name: %s", jobs[i].name);  printf("\nQualification Required: %s", jobs[i].qualification);  printf("\nExperience Required: %s", jobs[i].experience);  printf("\nSalary: %.2f", jobs[i].salary);  printf("\nCompany Name: %s\n", jobs[i].companyName);  printf("\nEnter the new Job Name: ");  takeInput(jobs[i].name);  printf("\nEnter the new Qualification Required: ");  takeInput(jobs[i].qualification);  printf("\nEnter the new Experience Required: ");  takeInput(jobs[i].experience);  printf("\nEnter the new Salary: ");  scanf("%lf", &jobs[i].salary);  getchar(); // Clear the newline character from the input buffer  printf("\nJob details updated successfully!\n");  found = 1;  break;  }  }  if (!found)  {  printf("\nNo jobs found with the given company name.\n");  }  }  void takeinput(char field[])  {  fgets(field, MAX\_FIELD\_SIZE, stdin);  field[strcspn(field, "\n")] = '\0'; // Remove newline character from input  }  void addJob(struct Job jobs[], int \*numJobs)  {  if (\*numJobs >= MAX\_JOBS)  {  printf("\nMaximum job limit reached!\n");  return;  }  struct Job newJob;  char ch;  printf("\nEnter your company name: ");  takeinput(newJob.companyName);  printf("\nEnter job name: ");  takeinput(newJob.name);  printf("\nEnter qualification required: ");  takeinput(newJob.qualification);  printf("\nEnter experience required: ");  takeinput(newJob.experience);  printf("\nEnter salary per annum: ");  scanf("%lf", &newJob.salary);  printf("\nCisco certification is required? : ");  scanf("%s", &newJob.ch);  printf("\nIs night shift required? (1 for Yes, 0 for No): ");  scanf("%d", &newJob.nightShift);  printf("\nEnter the link: ");  scanf("%s",&newJob.link);  printf("\nEnter the location: ");  scanf("%s",&newJob.location);  getchar(); // Clr the newline character from the input buffer  jobs[\*numJobs] = newJob;  (\*numJobs)++;  // Append the new job to the CSV file  FILE \*file = fopen(CSV\_FILENAME, "a");  if (file == NULL)  {  printf("Error opening file!");  return;  }  fprintf(file, "%s,%s,%s,%.2f,%s,%s,%s\n", newJob.name, newJob.qualification, newJob.experience, newJob.salary, newJob.companyName,newJob.link,newJob.location);  fclose(file);  printf("\nJob added successfully!\n");  }  void searchJobsByTitle(struct Job jobs[], int numJobs, char title[], char ciscoCert[], double seekerSalary, int seekerNightShift, int seekerExperience)  {  int found = 0;  struct user person1;  printf("\nMatching Jobs:\n");  FILE\* file = fopen(CSV\_FILENAME, "r");  if (file == NULL)  {  printf("Error opening file!\n");  return;  }  char line[256];  while (fgets(line, sizeof(line), file))  {  int recommendationPercentage = 0;  struct Job job;  sscanf(line, "%[^,],%[^,],%[^,],%lf,%[^,],%[^,],%[^\n]", job.name, job.qualification,  job.experience, &job.salary, job.companyName,job.link,job.location);  if (strcmp(job.name, title) == 0)  {  recommendationPercentage += 30; // Title match contributes 30% to recommendation  }  if (job.salary > seekerSalary)  {  recommendationPercentage += 30; // Salary match contributes 30% to recommendation  }  if (strcmp(ciscoCert, "yes") == 0)  {  recommendationPercentage += 20; // Cisco certificate contributes 20% to recommendation  }  if (job.nightShift == seekerNightShift)  {  recommendationPercentage += 10; // Night shift match contributes 10% to recommendation  }  if (atoi(job.experience) < seekerExperience)  {  recommendationPercentage += 10; // Experience match contributes 10% to recommendation  }  if (recommendationPercentage >= 80)  {  printf("\nJob Name: %s", job.name);  printf("\nQualification Required: %s", job.qualification);  printf("\nExperience Required: %s", job.experience);  printf("\nSalary: %.2f", job.salary);  printf("\nCompany Name: %s", job.companyName);  printf("\nLink for verify the reliability of the company: %s",job.link);  printf("\nLocation: %s",job.location);  printf("\nRecommendation Percentage: %d%%\n", recommendationPercentage);  found = 1;  }  }  fclose(file);  if (!found)  {  printf("\nNo jobs found with the given title or matching constraints.\n");  }  }  void searchJobSeekersByRecommendation()  {  char searchjob[MAX\_FIELD\_SIZE];  int found = 0;  int recommendationPercentage;  struct user person;  FILE \*fp;  printf("\nEnter the jobname: ");  takeInput(searchjob);  printf("\nHow much recommendation do you require: ");  scanf("%d", &recommendationPercentage);  fp = fopen("Users.csv", "r");  if (fp == NULL)  {  printf("Error opening file!\n");  return;  }  while (fscanf(fp, "%[^,],%[^,],%[^,],%[^,],%[^,],%[^,],%d,%[^\n]\n",  person.fullname, person.email, person.password, person.username,  person.phone, person.job, &person.recommendationPercentage, person.qualification) == 8)  {  if ((strcmp(person.job, searchjob) == 0) && (person.recommendationPercentage >= recommendationPercentage))  {  printf("\n\t\tMatching job seekers:");  printf("\n\n|Full Name :\t%s", person.fullname);  printf("\njob name :\t%s", person.job);  printf("\n|Email :\t%s", person.email);  printf("\n|Username :\t%s", person.qualification);  printf("\n|Contact number:\t%s", person.phone);  printf("\n|recommendation percentage:\t%d\n", person.recommendationPercentage);  found++;  }  }  fclose(fp);  if (found == 0)  {  printf("\nNo job seekers found with the given criteria.\n");  }  }  **Main function:**    int main()  {  struct Job jobs[MAX\_JOBS];  int applyjob;  int mark1;  int mark2;  char dob[10];  char arrear[10];  char lan[10];  char skills[10];  int numJobs = 0; // Number of jobs in the array  char searchCompany[100];  int choice;  char userType;  printf("Are you a job seeker (S), or an admin (A)? ");  scanf("%c", &userType);  getchar(); // Clear the newline character from the input buffer  if (userType == 'A' || userType == 'a')  {  int aopt;  FILE \*fp;  loadJobs(jobs, &numJobs);  while (aopt != 3)  {  printf("\n1. Sign up as admin");  printf("\n2. Log in as admin");  printf("\n3. Exit");  printf("\nEnter your choice: ");  scanf("%d", &aopt);  getchar();  switch (aopt)  {  case 1:  printf("Enter the admin name: ");  takeinput(admin1.name);  printf("Enter your password: ");  takePassword(admin1.pword);  fp = fopen("Admins.csv", "a");  if (fp == NULL)  {  printf("Error opening file!\n");  return 1;  }  fprintf(fp, "%s,%s\n", admin1.name, admin1.pword);  fclose(fp);  printf("\n\nAdmin registration is successful!\nThank you!\nYour username is %s\n",admin1.name);  break;  case 2:  if (adlogin())  {  while(choice!=3)  {  printf("\n\t\t\t Welcome %s", admin1.name);  printf("\nAdmin Options:");  printf("\n1. Add Job");  printf("\n2. Edit the job details");  printf("\n3.search jobseeker");  printf("\n4.Exit");  printf("\nEnter your choice: ");  scanf("%d", &choice);  getchar(); // Clear the newline character from the input buffer  switch (choice)  {  case 1:  addJob(jobs, &numJobs);  break;  case 2:  editJobDetails(jobs,numJobs);  break;  case 3:  searchJobSeekersByRecommendation();  break;  case 4:  printf("\nThank You for your time!!!\n Bye.. Bye....\n");  exit(0);  return 0;  }  }  }  }  }  }  else if (userType == 'S' || userType == 's')  {  FILE \*fp;  int opt;  struct user person1;  char password2[MAX\_FIELD\_SIZE];  printf("\n\t\t\tWelcome to the job seeker platform!");  printf("\nPlease choose your operation");  printf("\n1. Signup");  printf("\n2. Login");  printf("\n3.Edit deatails");  printf("\n4. Exit");  printf("\nEnter your choice: ");  scanf("%d", &opt);  fgetc(stdin);  switch (opt)  {  case 1:  printf("\nEnter your Full name: ");  takeInput(person1.fullname);  printf("\nEnter your email: ");  takeInput(person1.email);  printf("\nEnter your contact number: ");  takeInput(person1.phone);  printf("\nEnter your password: ");  takePassword(person1.password);  printf("\nConfirm your password: ");  takePassword(password2);  printf("Enter your education Qualification: ");  takeInput(person1.qualification);  printf("\nEnter the job you want to search:");  takeInput(person1.job);  printf("Enter the recommendation percentage: ");  scanf("%d",&person1.recommendationPercentage);  if (strcmp(person1.password, password2) == 0)  {  generateUsername(person1.email, person1.username);  fp = fopen("Users.csv", "a");  if (fp == NULL)  {  printf("Error opening file!\n");  return 1;  }  fprintf(fp, "%s,%s,%s,%s,%s,%s,%d,%s\n", person1.fullname, person1.email,  person1.password, person1.username, person1.phone,person1.job,person1.recommendationPercentage,person1.qualification);  fclose(fp);  printf("\n\nUser registration is successful!\nThank you!\nYour username is %s\n",  person1.username);  }  else  {  printf("\n\nPasswords do not match.\n");  break;  }  break;  case 2:  if (login())  {  printf("\nLogin successful.\n");  char ciscoCert[MAX\_FIELD\_SIZE];  printf("\nDo you have a Cisco certificate for internetworking? (yes/no): ");  takeInput(ciscoCert);  if (strcmp(ciscoCert, "yes") == 0)  {  // Job Seeker - Search for jobs  printf("\n\t\t\tWell done! \n");  char searchTitle[MAX\_FIELD\_SIZE];  FILE \*fp;  fp = fopen("Users.csv", "a");  if (fp == NULL)  {  printf("Error opening file!\n");  return 1;  }  fprintf(fp, "%s", searchTitle);  fclose(fp);  double seekerSalary;  int seekerNightShift;  int seekerExperience;  printf("\nEnter job title to search: ");  takeInput(searchTitle);  printf("\nEnter your years of experience: ");  scanf("%d", &seekerExperience);  printf("\nEnter your expected salary per annum: ");  scanf("%lf", &seekerSalary);  printf("\nAre you available for night shifts? (1 for Yes, 0 for No): ");  scanf("%d", &seekerNightShift);  getchar(); // Clear the newline character from the input buffer  searchJobsByTitle(jobs, numJobs, searchTitle, ciscoCert, seekerSalary, seekerNightShift, seekerExperience);  printf("\nDid you accept the job offer (1 for yes/0 for no): ");  scanf("%d",&applyjob);  if (applyjob==1){  printf("\nyou are accept the job offer!!");  printf("\nplease provide the following asked details....");  printf("\nEnter your 10th mark: ");  scanf("%d",&mark1);  printf("\nEnter your 12th mark: ");  scanf("%d",&mark2);  printf("\nDid you have any arrear (yes/no): ");  scanf("%s",&arrear);  printf("\nMention what are the languages you know: ");  scanf("%s",&lan);  printf("\nIf you have any other skills please mention it: ");  scanf("%s",&skills);  printf("\n\n\t\t Thank you for providing above details....!!");  printf("\nYour application is in process....");  printf("\nPlease wait few weeks until you get information from HR Manager");  }else{  printf("\nSorry!You have not applied for the job.... ");  printf("\n\t\tThank you for your time.....!");  }  }  else  {  printf("\nSorry, you need to have a Cisco certificate for internetworking to proceed.\n");  }  }  else  {  printf("\nInvalid username or password.\n");  Beep(300, 700);  }  break;  case 3:  editDetails();  break;  case 4:  printf("\nThank You for your time!!!\n Bye.. Bye....\n");  return 0;  }  }  return 0;  }  **Testcases**:  Match the jobs with jobseeker’s profile:  A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated  Match the jobseekers with HR Manager search:      Job added by HR Manager:  A screenshot of a computer  Description automatically generated  Special feature:  Nowadays credibility is questionable in Jobseeker platforms because the jobs posted there are questionable whether the jobs are real or the companies are real so we provide a link to the job seekers through that link they can visit the company's website and confirm the credibility of the company.  A screen shot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated  **Limitations**   1. **Security Vulnerabilities**: The project might be susceptible to security vulnerabilities, such as SQL injection, cross-site scripting (XSS), or insufficient authentication and authorization mechanisms. These vulnerabilities could expose user data to potential attackers and compromise the overall system's integrity. Addressing these security concerns should be a top priority to ensure the platform's safety and protect user information.In future we definitely develop our platform including with advanced security systems. 2. **Scalability Challenges**: As the number of users and job listings grows, the current implementation might face scalability challenges. Handling a significant increase in traffic and data volume could lead to performance issues or system overload. Ensuring that the platform is designed to scale efficiently will be essential to support a large user base and accommodate a growing number of job listings. 3. **Limited Recommendation Algorithm**: The current recommendation system may be rudimentary, relying on a few criteria for matching job seekers with job listings. This approach might not provide the most accurate and personalized job recommendations, potentially leading to mismatches between job seekers and job opportunities. Enhancing the recommendation algorithm to consider more relevant factors and employing machine learning techniques could significantly improve the quality of job suggestions. | | | | | | |
|  | | | | | | |