JobConnect: Platform For Jobs

UCS2201 – Fundamentals and Practice of Software Development

A PROJECT REPORT

Submitted By

Saisandeep Sangeetham - 3122 22 5001 119

Shaun Allan H - 3122 22 5001 127

Shreyamanisha C Vinay - 3122 22 5001 130

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 "**JobConnect: Platform For Jobs**" is the bonafide work of "Saisandeep Sangeetham (3122 22 5001 119), Shaun Allan H (3122 22 5001 127) , Shreyamanisha C Vinay (3122 22 5001 130) " 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

1. INTRODUCTION

The purpose of this report is to present the development and implementation of a job seeking platform called Job Connect using the C programming language. JobConnect aims to revolutionize the traditional job search process by providing a comprehensive platform for job seekers to connect with potential employers and explore employment opportunities and vice versa. This report will discuss the key features, design principles, and implementation details of JobConnect, highlighting the utilization of C programming concepts and techniques.

In today's highly competitive job market, finding suitable employment opportunities has become a challenging task for job seekers and similarly finding candidates that satisfy the specifications of a company has become difficult. To address this issue, the C project "JobConnect" has been developed as a job seeking platform, who's primary objective is to create a user-friendly platform that simplifies the job search process and enhances the overall experience for both job seekers and employers. By leveraging the power of C programming, the project aims to implement essential functionalities such as job listing, candidate profiles, application management, and communication channel that offers a streamlined and efficient approach to form a connection between the candidates and companies.

By leveraging C programming, the project demonstrates the versatility and power of the language in creating efficient and robust systems. Furthermore, this project serves as a learning opportunity for students to enhance their programming skills, particularly in C, and gain practical experience in developing real-world applications.

In the subsequent sections of this report, we will delve into the design and implementation aspects of JobConnect, showcasing the key functionalities, data structures, algorithms, and best practices employed in its development.

2. PROBLEM STATEMENT

Develop a software system to create a platform for Job Seekers and Talent Acquisition Managers where it acts as an intermediating device between the Job Seekers and Talent Acquisition Team.

Here, the Job Seekers can explore the available job positions which match their profile. And the Talent Acquisition Team can search for deserving applicants who will qualify their job requirements.

This can be done by calculating the recommendation percentage after inputting the constraints from both sides using the formula recommendation percentage is the ratio of accumulation of weighted scores of all the satisfying constraints to the weighted scores of all the available constraints multiplied by 100.

3. EXPLORATION BEYOND PROBLEM STATEMENT

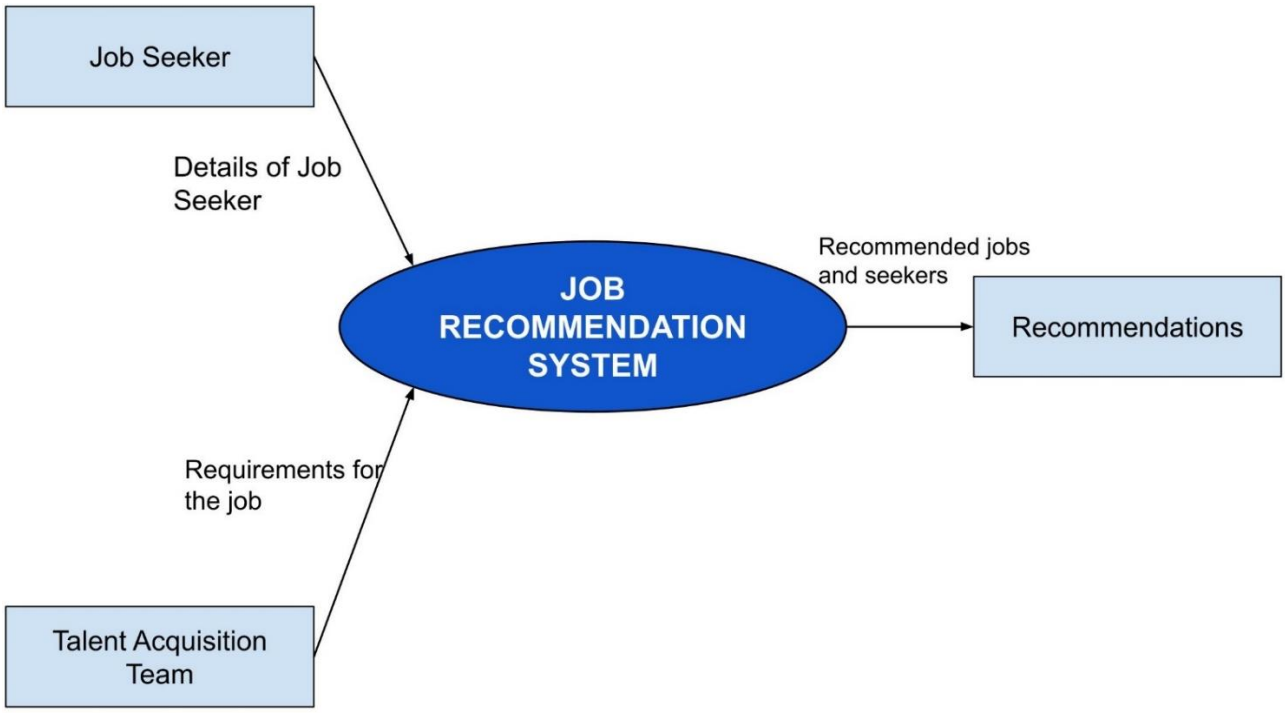
We have explored a lot beyond the problem statement. Here are the following:

- Encrypted the password before storing in database
- Added more constraints (Job Seeker side - Job Type, Company Type, Location, Day/Night Shift, Work from Home, Salary)

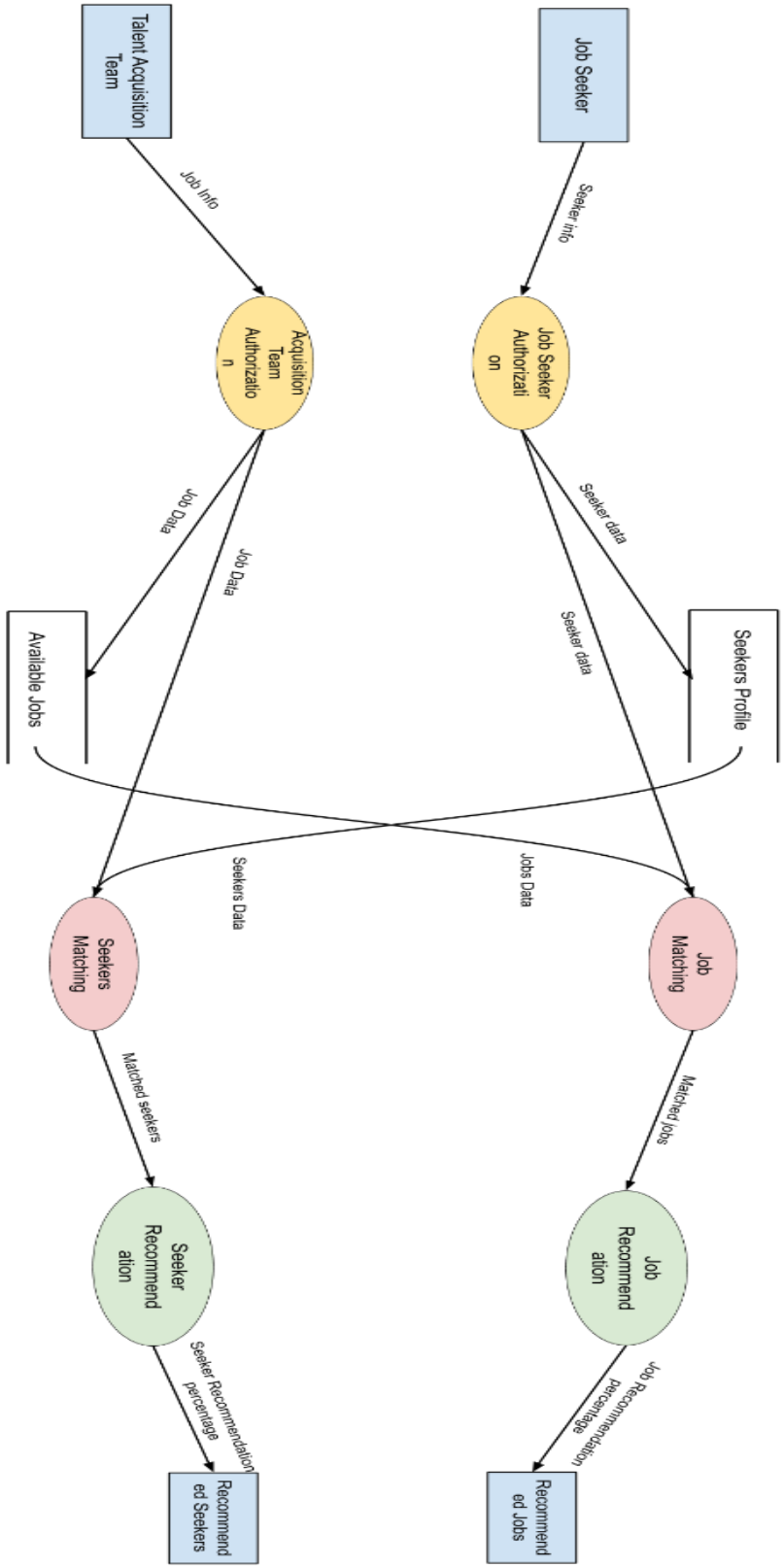
  (Talent Acq side - Qualification, Age, Experience, Shift, CGPA, 10^{th} & 12^{th} Marks)
- Get the 10^{th}, 12^{th} and degree certificate from seeker to check manually
- Integrate Machine learning algorithms to calculate score
- Application status (Seeker applies a job to a company and the talent acquisition Side either rejects or accepts the application and this is notified to the seeker)

4. DATA FLOW DIAGRAMS

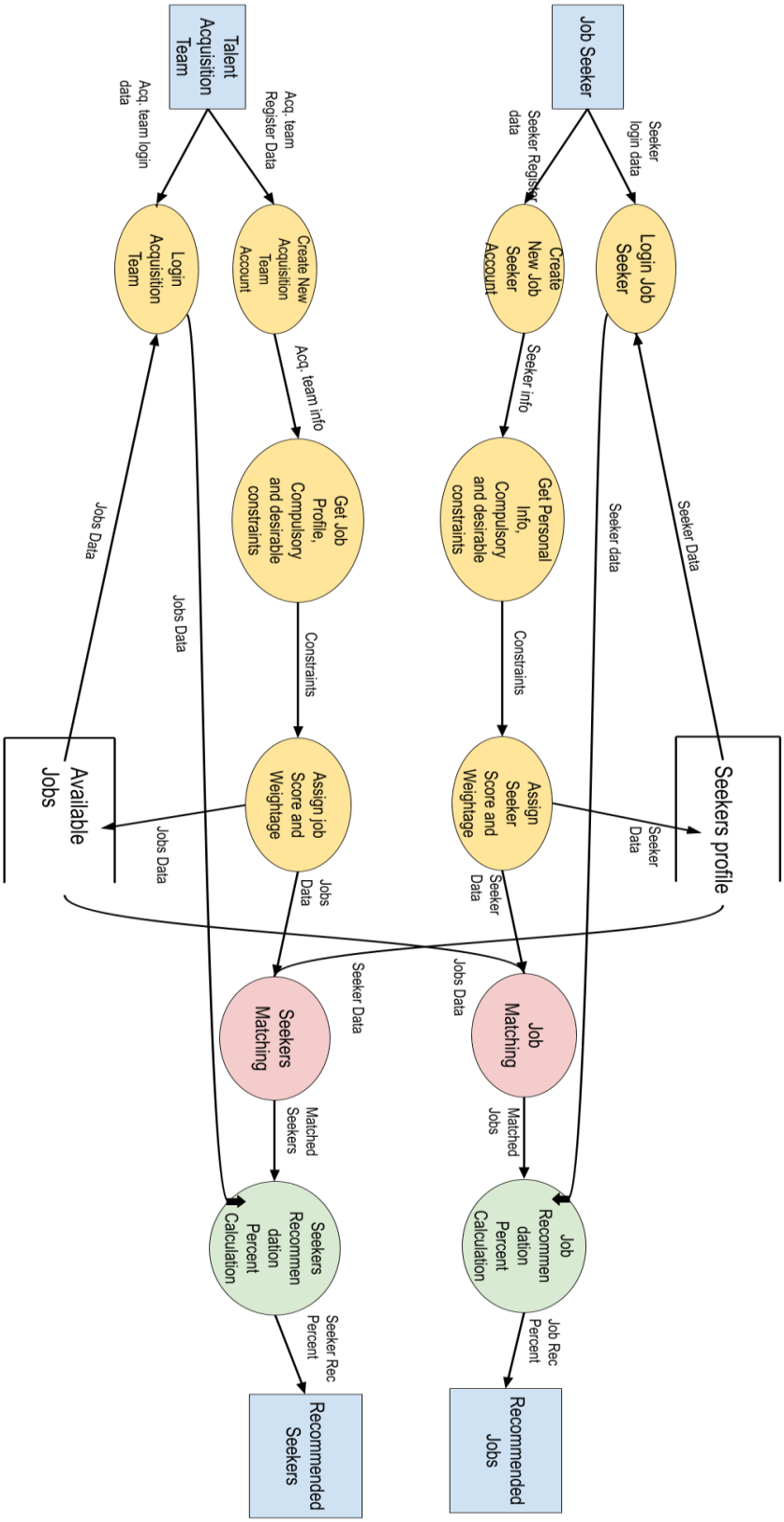
4.1. LEVEL 0



4.2. LEVEL 1


Talent Acquisition Team
Job Seeker
Job Info
Seeker info
Acquisition Team Authorizatio n
Job Seeker Authorizati on
Job Data
Job Data
Seeker data
Seeker data
Available Jobs
Seekers Profile
Seekers Data
Jobs Data
Seekers Matching
Job Matching
Matched seekers
Matched Jobs
Seeker Recommend ation
Job Recommend ation
Seeker Recommendation percentage
Job Recommendation percentage
Recommend ed Seekers
Recommend ed Jobs

4.3. LEVEL 2

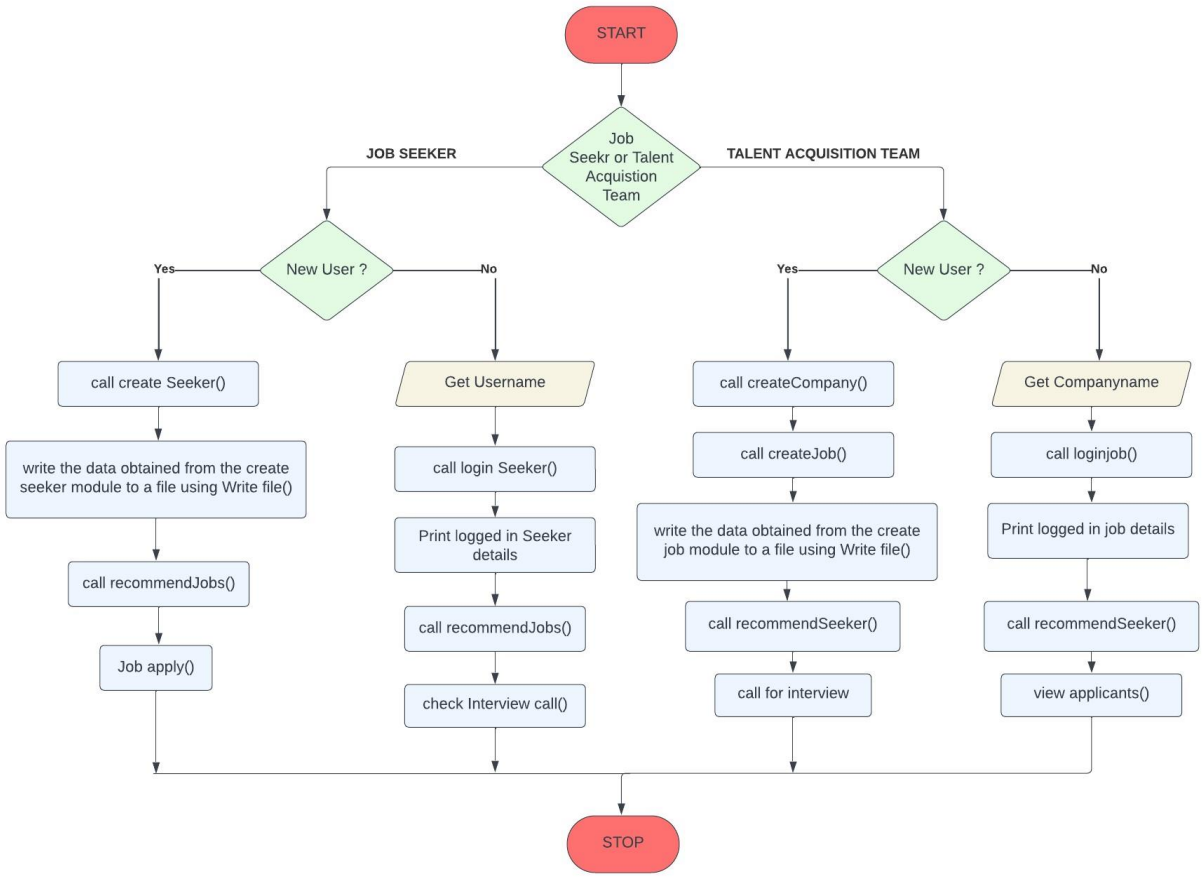

Job Seeker
Talent Acquisition Team
Seeker login data
Seeker Register data
Acq. team login data
Acq. team Register Data
Login Job Seeker
Create New Job Seeker Account
Login Acquisition Team
Create New Acquisition Team Account
Seeker info
Acq. team info
Get Personal Info, Compulsory and desirable constraints
Get Job Profile, Compulsory and desirable constraints
Seeker data
Seeker Data
Jobs Data
Constraints
Assign Seeker Score and Weightage
Assign job Score and Weightage
Seekers profile
Available Jobs
Seeker Data
Jobs Data
Job Matching
Seekers Matching
Matched Jobs
Matched Seekers
Job Recommendation Percent Calculation
Seekers Recommendation Percent Calculation
Job Rec Percent
Seeker Rec Percent
Recommended Jobs
Recommended Seekers

5. DETAILED DESIGN

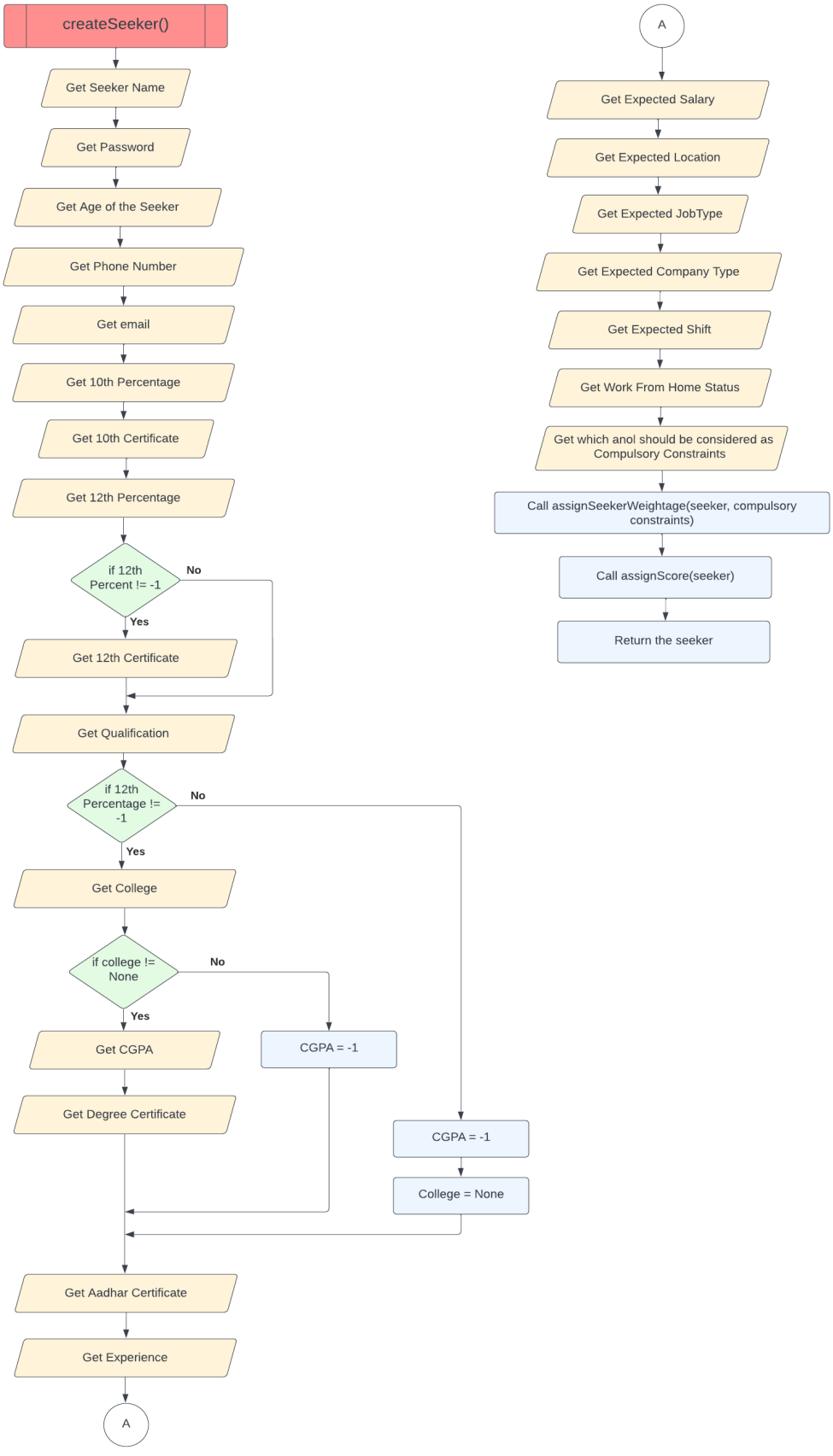
5.1. ARCHITECTURAL DESIGN

5.2. FLOWCHART FOR EACH MODULE

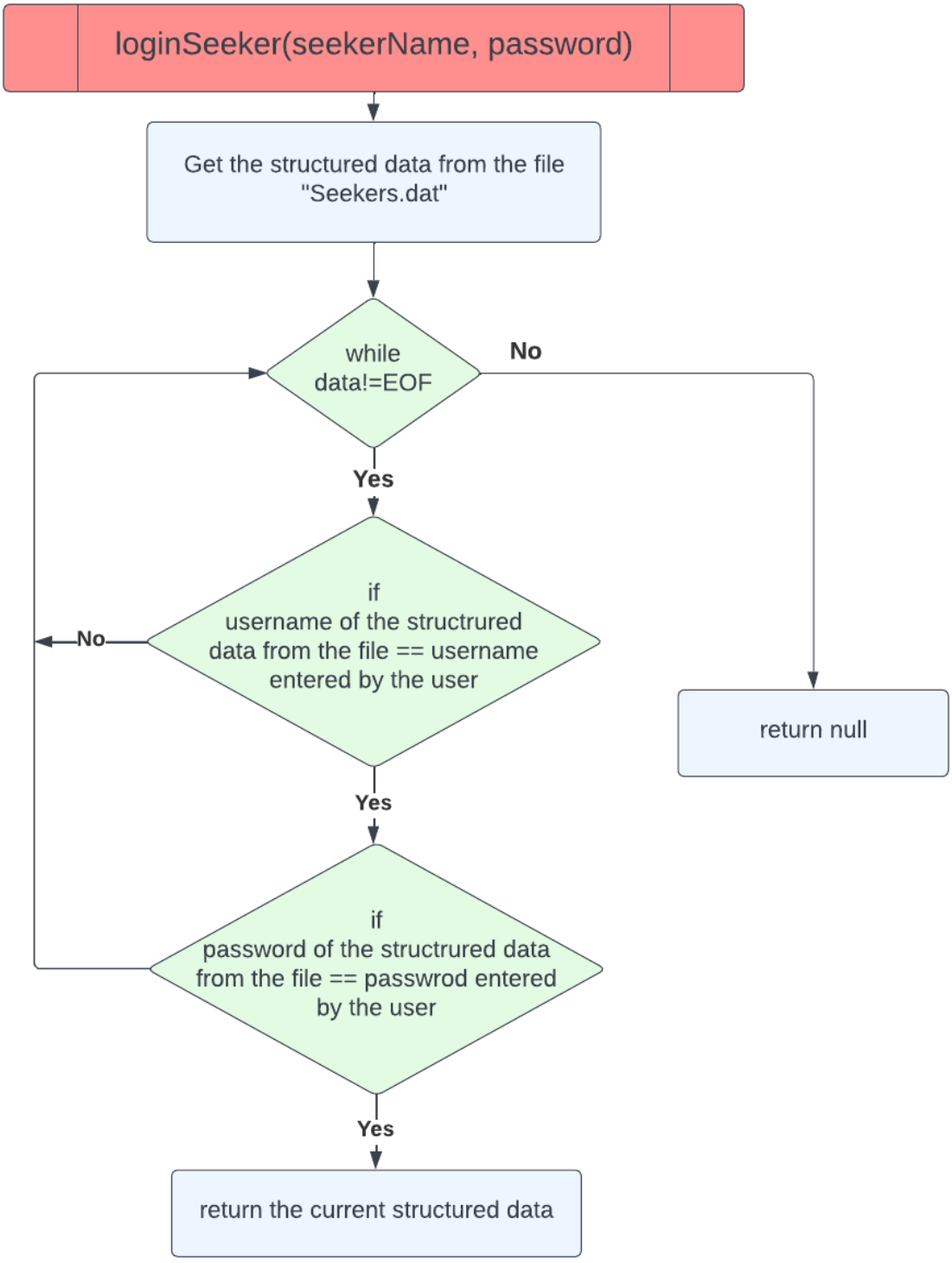
5.2.1. Main Module:



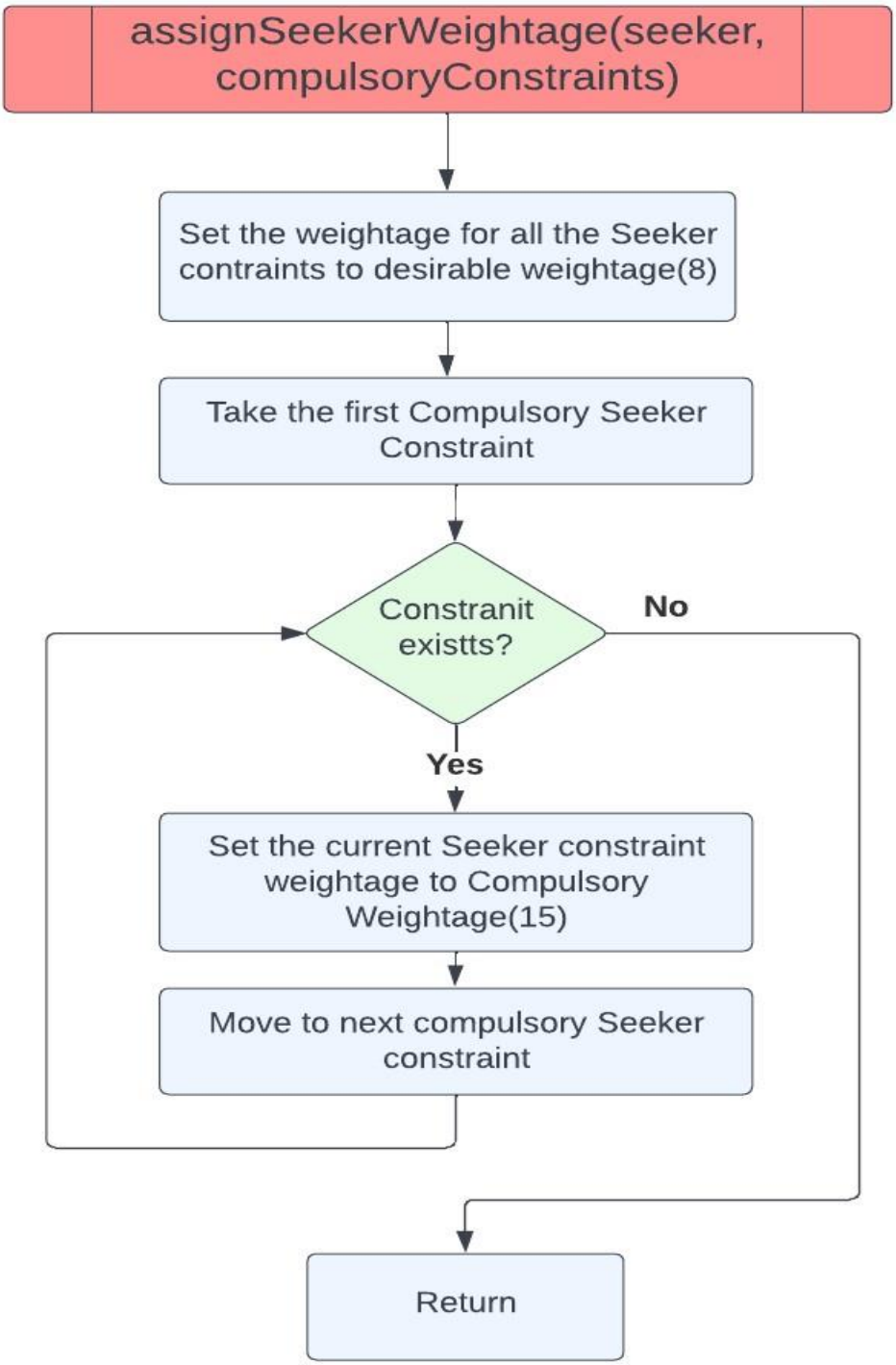
5.2.2. Create Seeker Module:



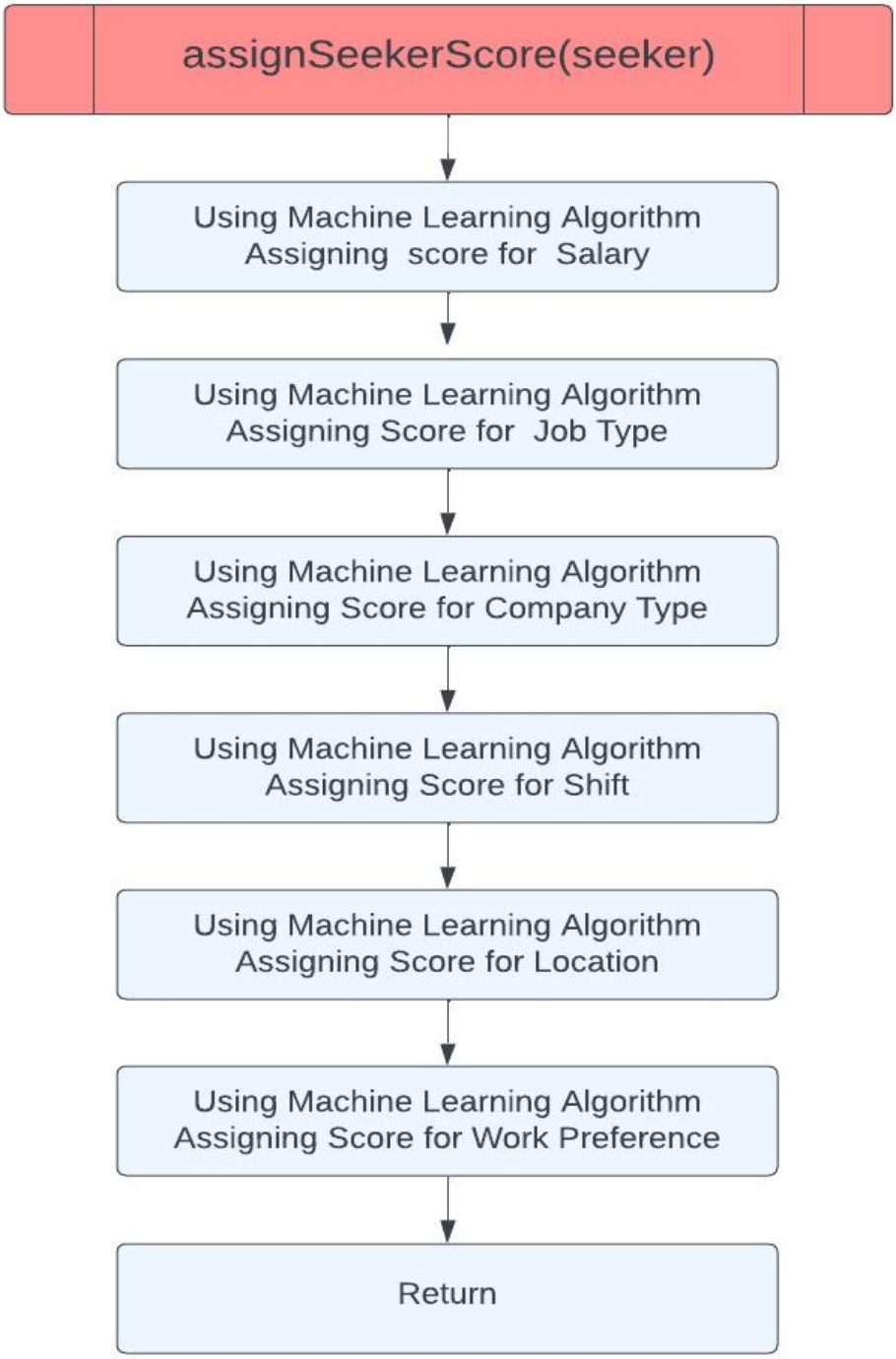
5.2.3. Seeker Login Module:



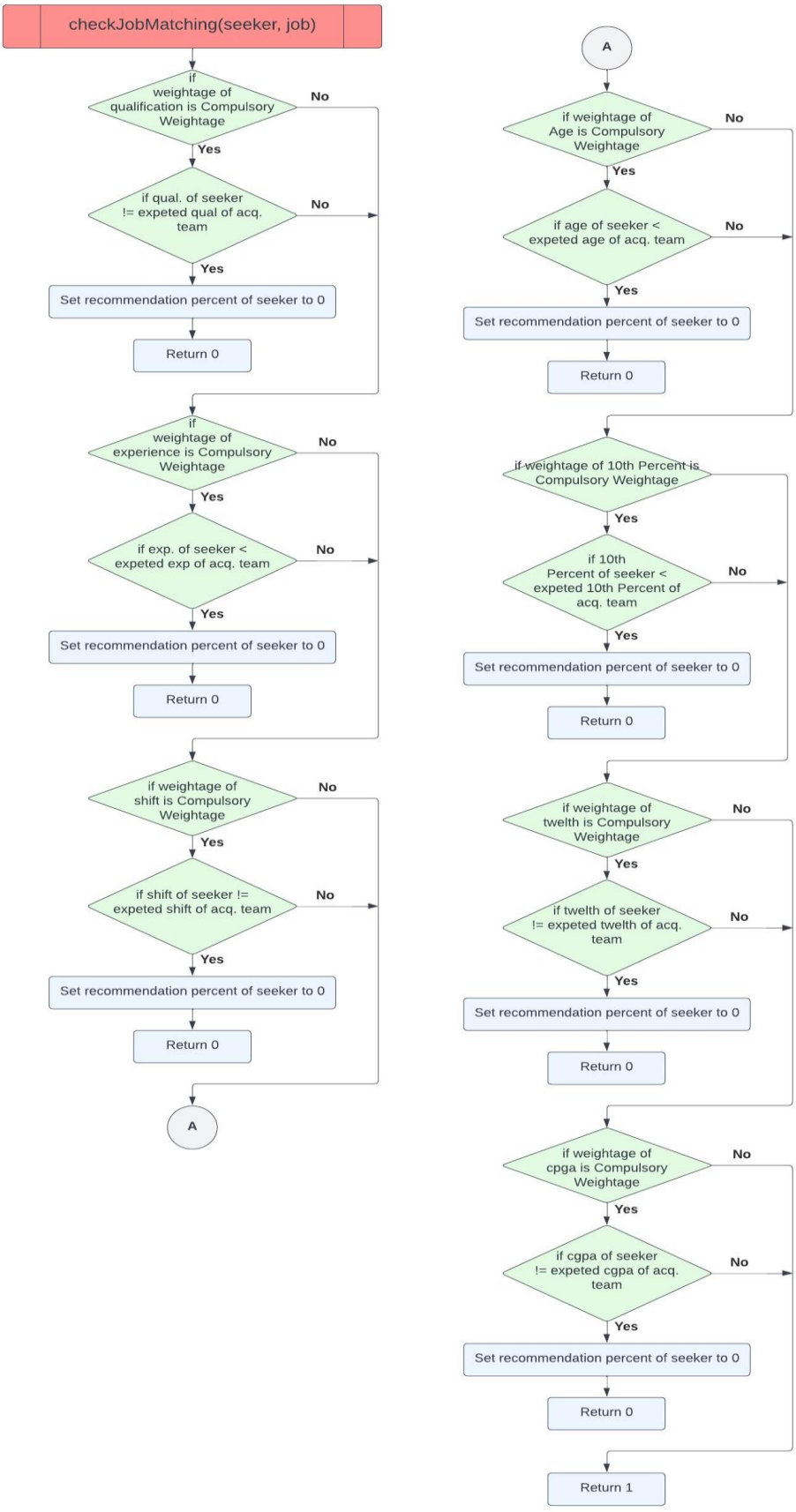
5.2.4. Assign Seeker Weightage:



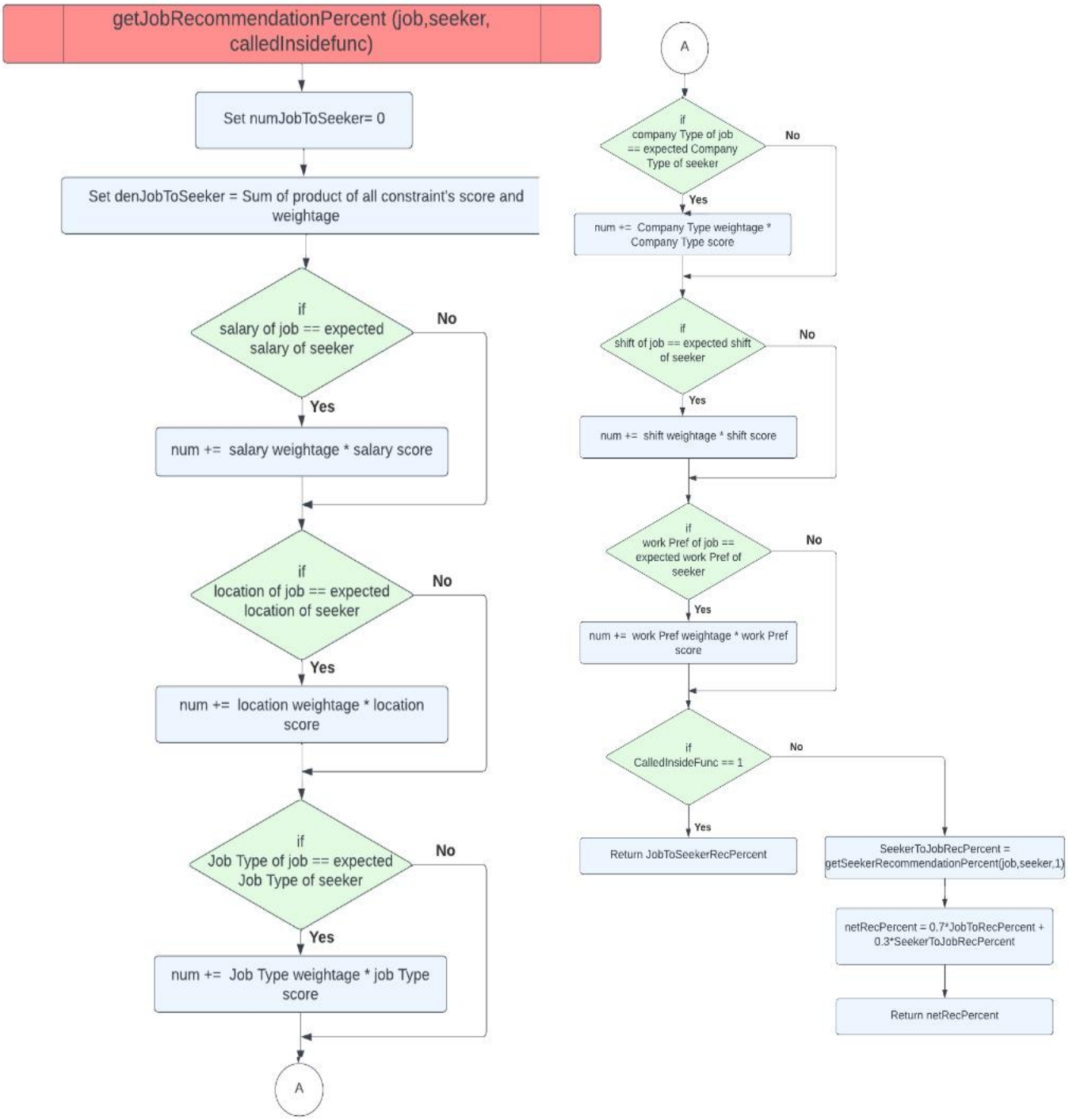
5.2.5. Assign Seeker Score:



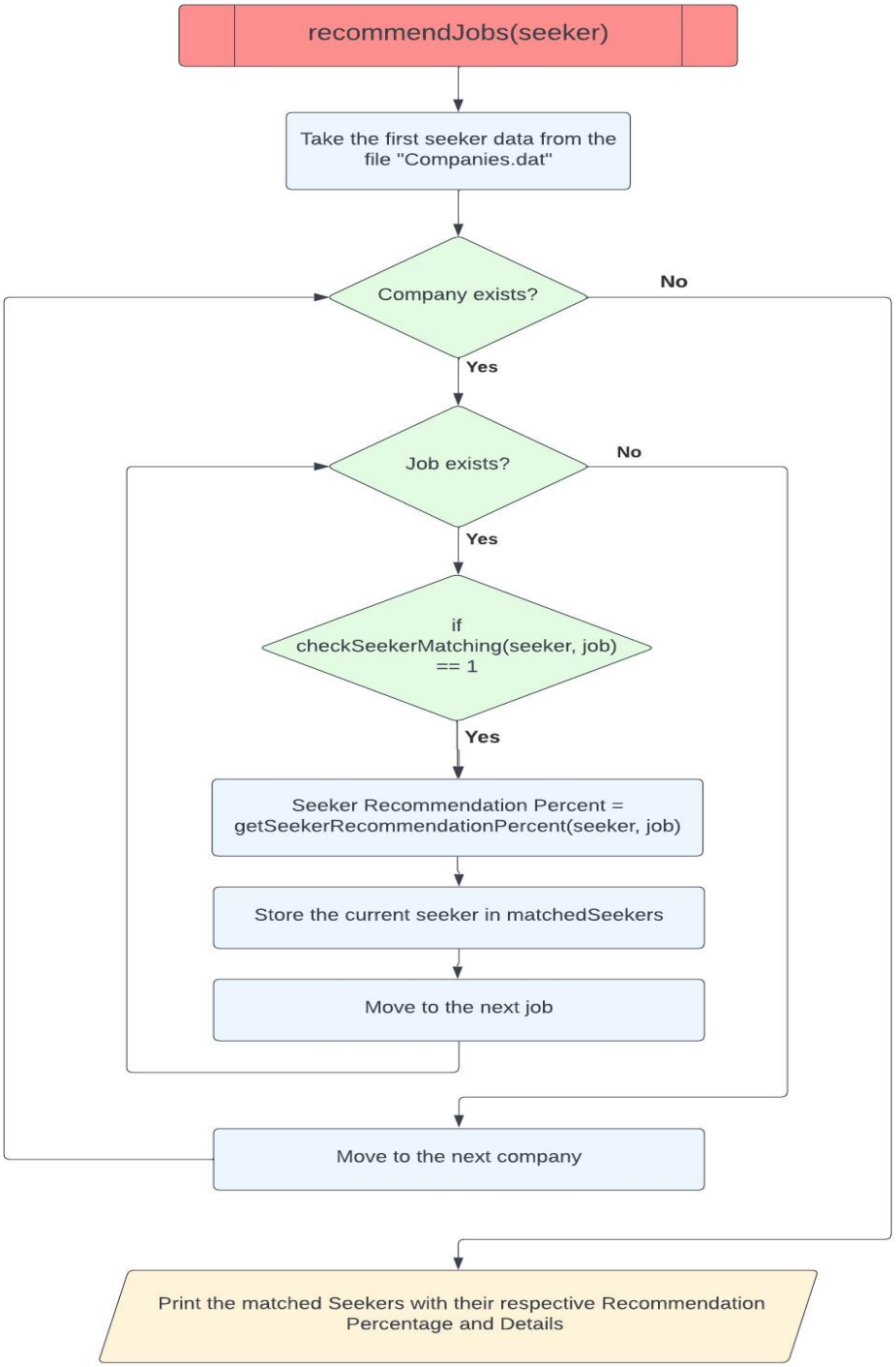
5.2.6. Check Job Matching:



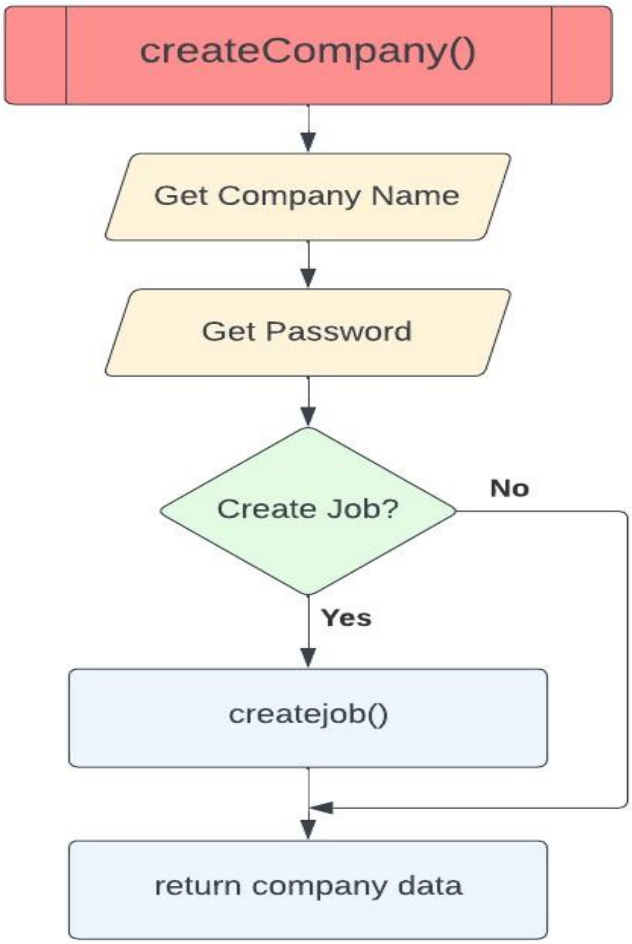
5.2.7. Get Job Recommendation Percent:



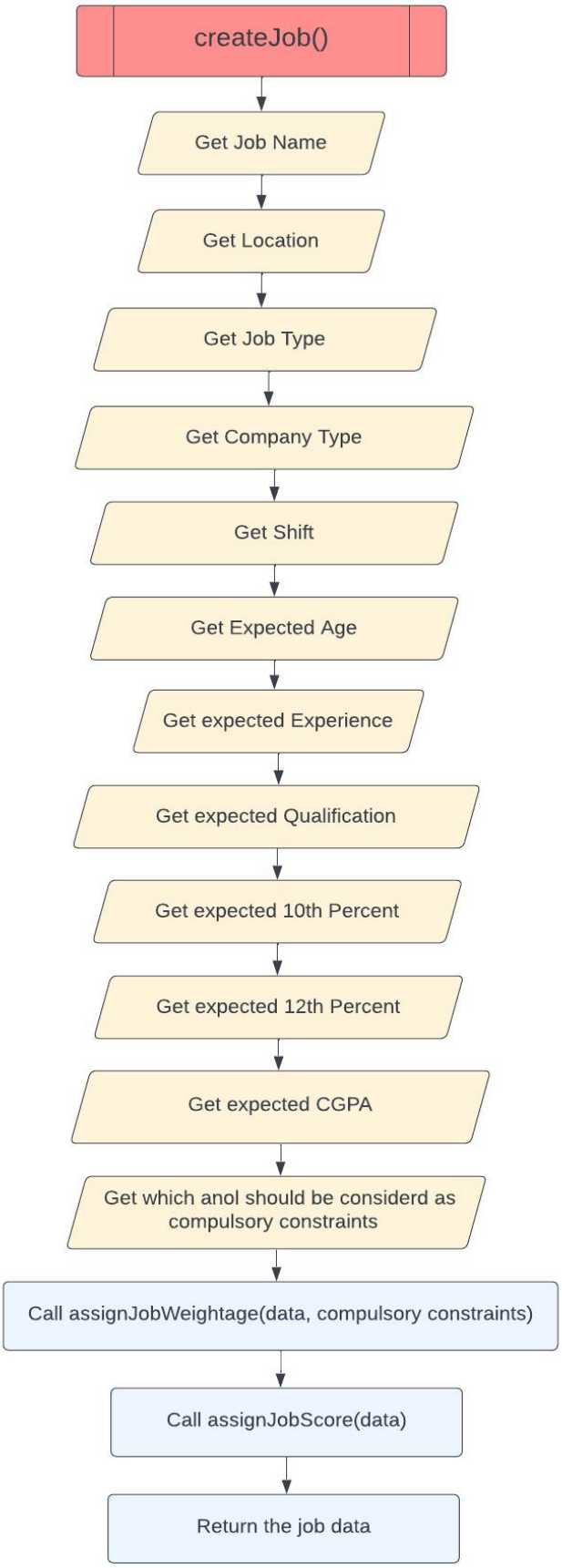
5.2.8. Recommend Jobs:



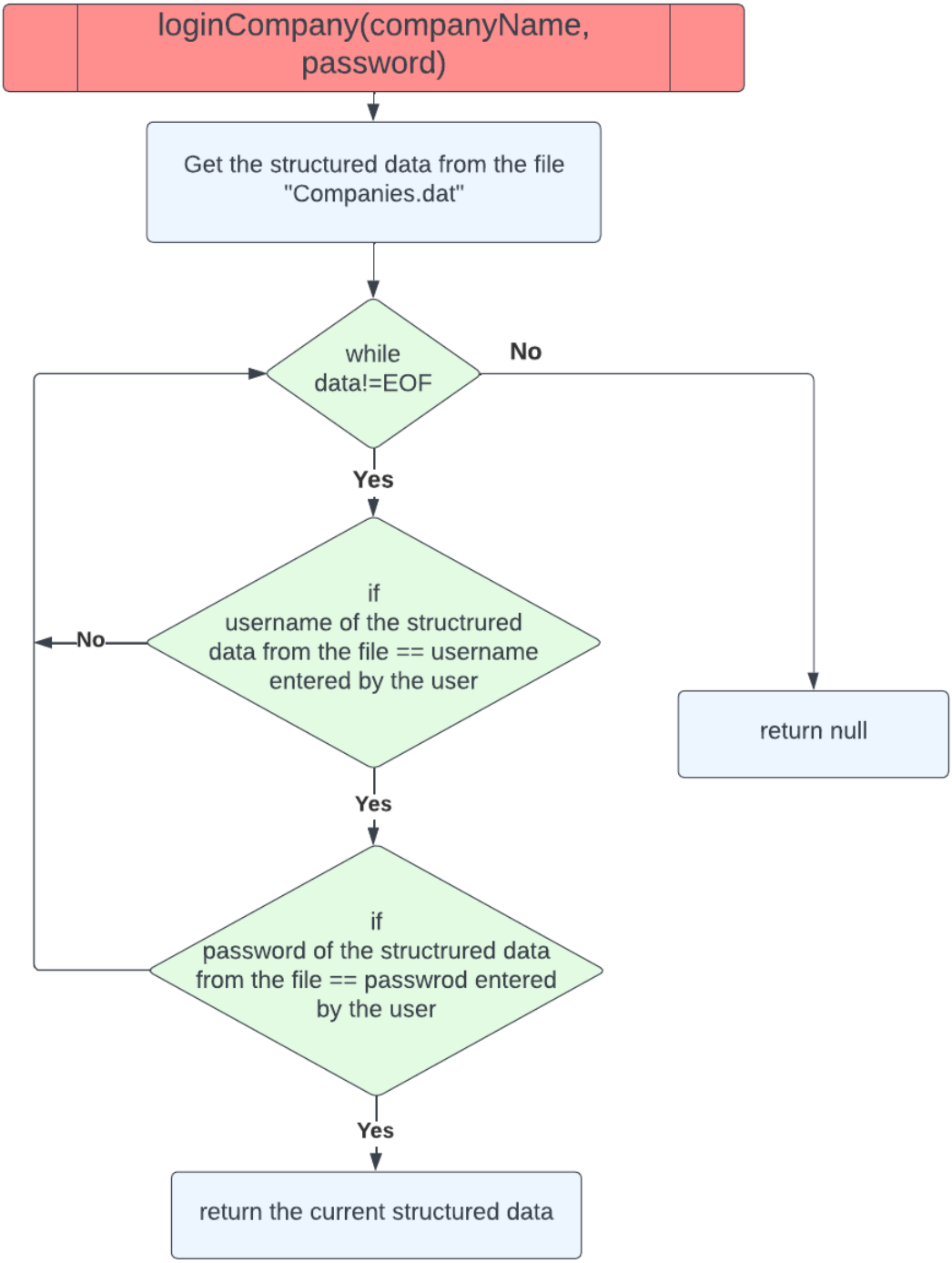
5.2.9. Create Company:



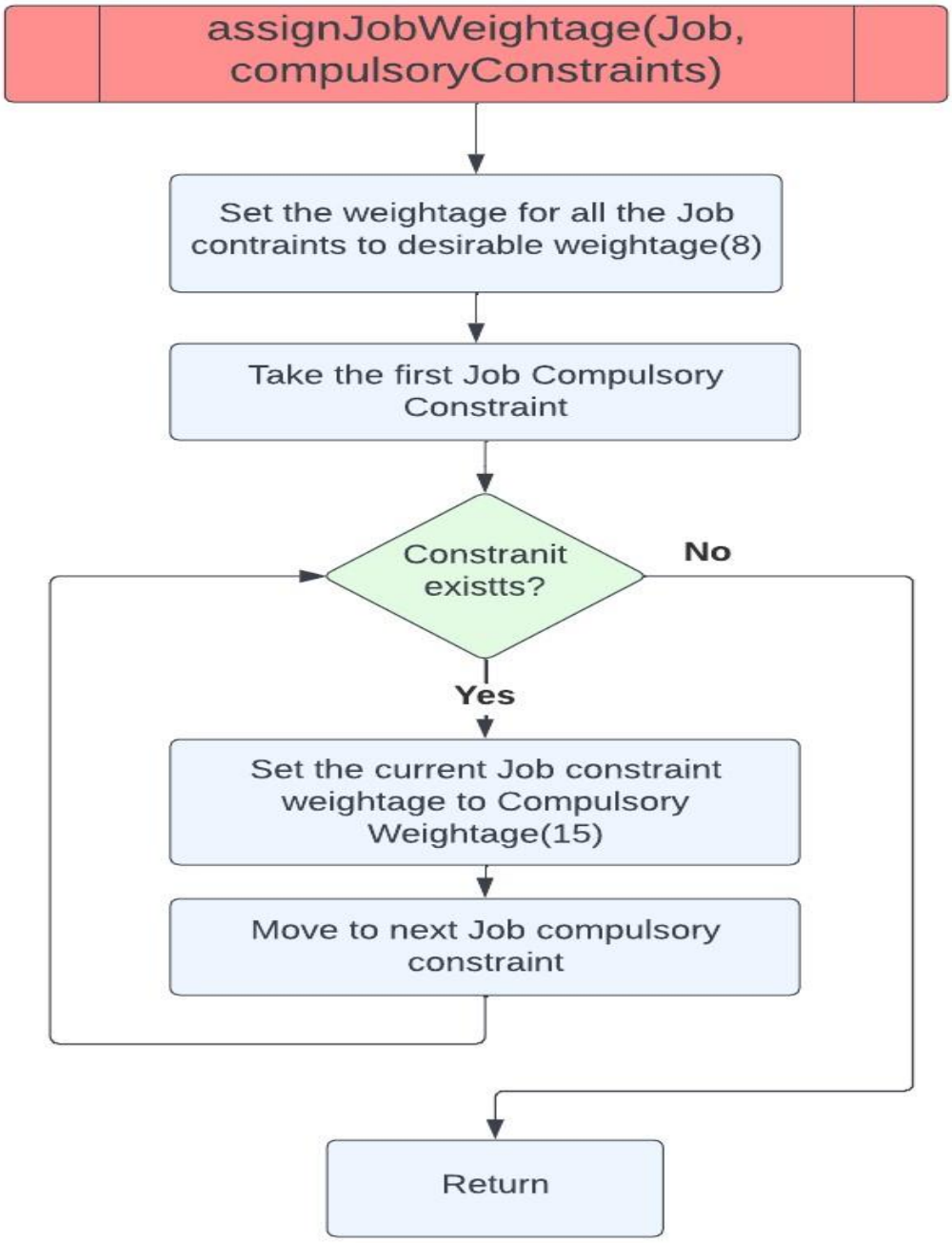
5.2.10. Create Job Module:



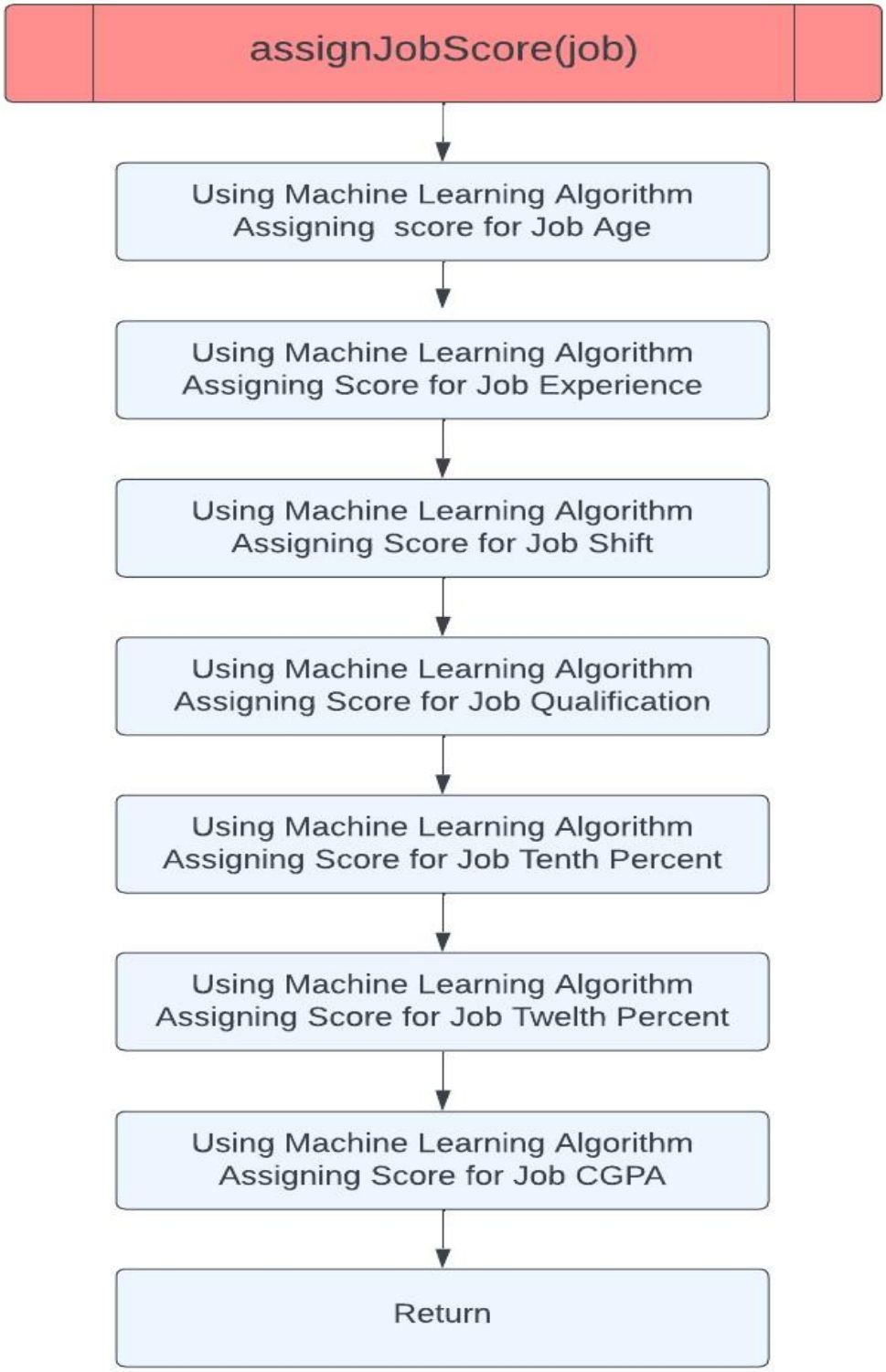
5.2.11. Job Login Module:



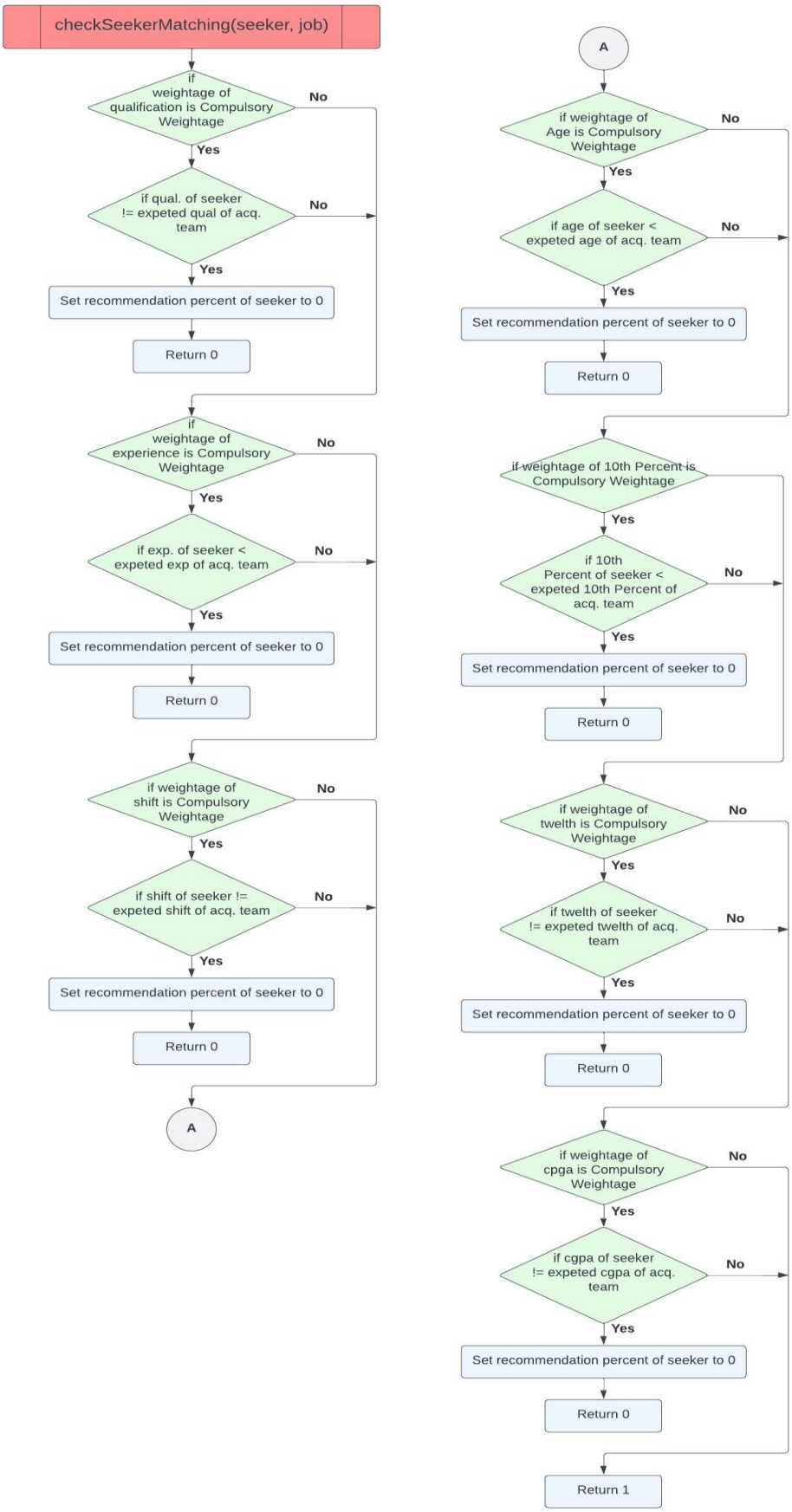
5.2.12. Assign Job Weightage:



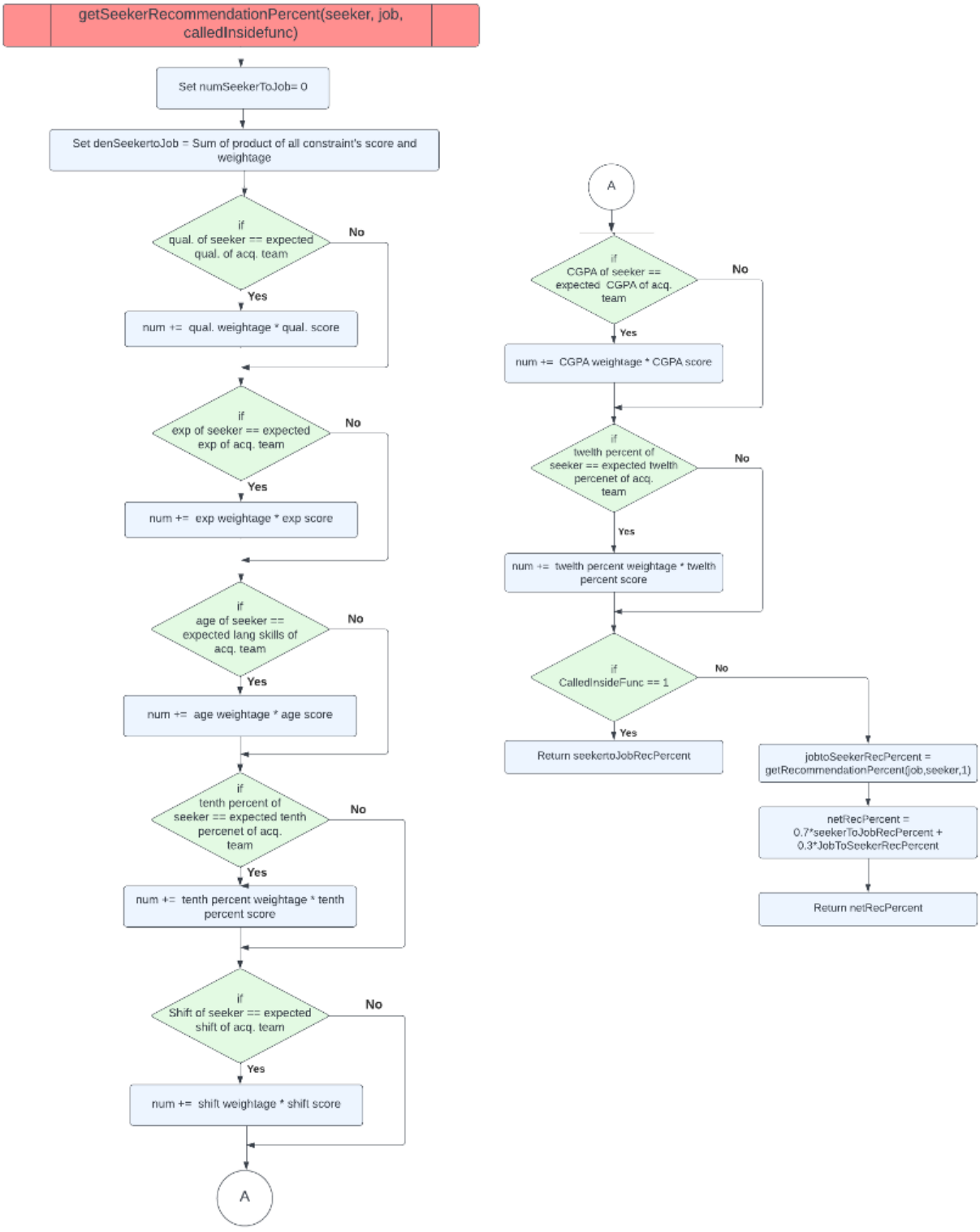
5.2.13. Assign Job Score:



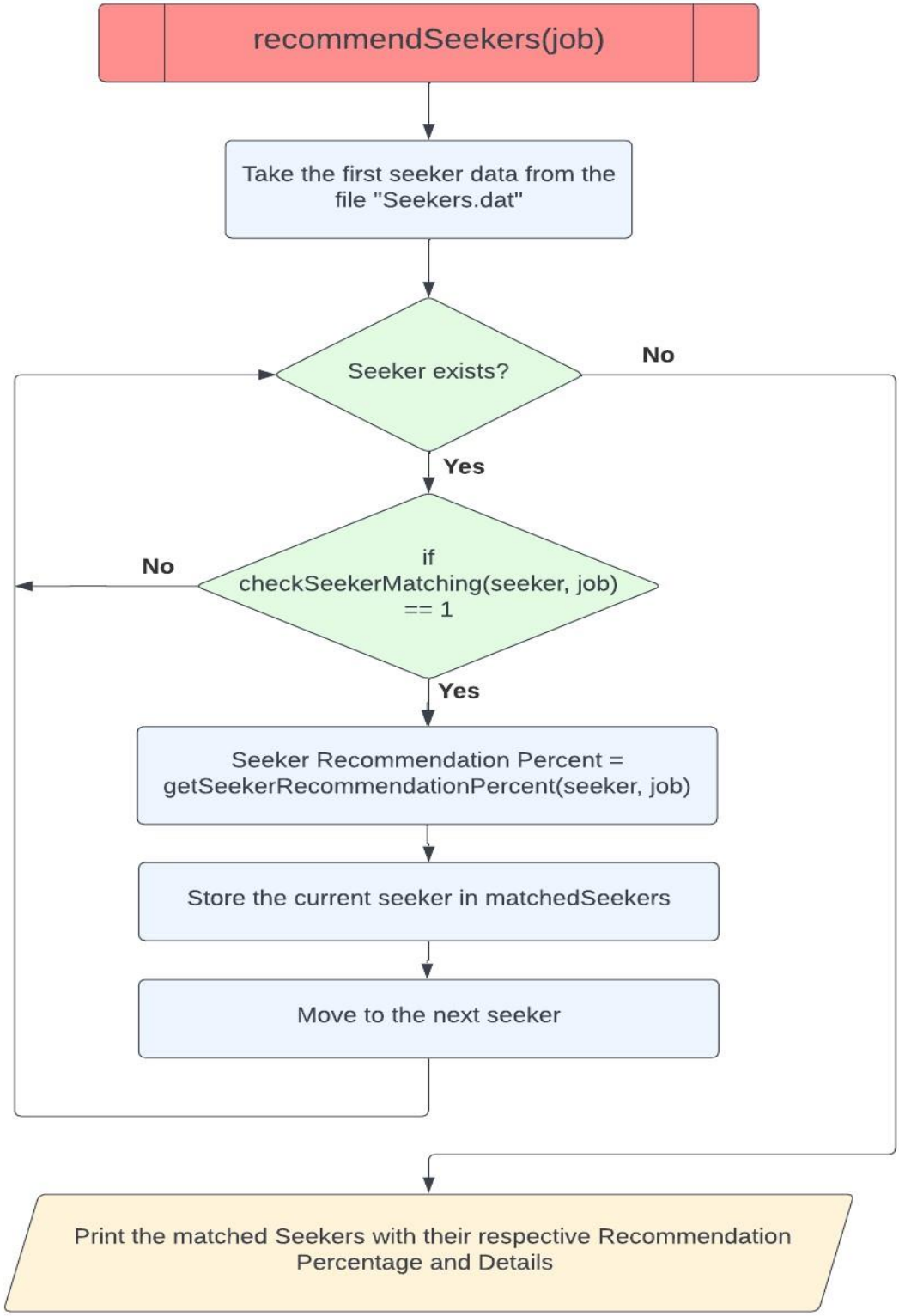
5.2.14. Check Seeker Matching:



5.2.15. Get Recommendation Percentage Module:



5.2.16. Recommend Seekers:



6. DESCRIPTION

6.1. MAIN MODULE

- This main module serves as the central component that brings together all the functionalities and modules required for the system. It begins by prompting the user to choose between the job seeker or talent acquisition team options.
- For job seekers, it provides functionalities such as creating an account or logging in, with the data being stored in files. Additionally, seekers can access features like viewing recommended jobs and applying for them. An extra module is included specifically for the login side to handle checking for interview calls.
- On the other hand, for the talent acquisition team, the module supports functionalities such as creating a company profile, posting job openings, viewing recommended seekers, and initiating interview calls. Similarly, an extra module is available for the login side to facilitate viewing applicant information.
- By encompassing all these functionalities, the main module acts as the central hub for coordinating and integrating the different features required for both job seekers and the talent acquisition team

6.2. CREATE SEEKER MODULE

- we simply ask the user for input regarding the constraints, such as seeker name, password, age, phone number, email, documents (10th, 12th), CGPA, qualification, etc. Next, we get information about the number of compulsory constraints and which constraints are required to be provided in SL.NO order. Finally, we call the assign weightage and assign score modules and return the seeker structure.

6.3. SEEKER LOGIN MODULE

- The seeker name and password are passed through this module, and after getting the structured data from the file "Seekers.dat" and reading it from beginning to end, we check to see if the seeker name is available. If it is, we then check to see if the password is correct. If both constraints are satisfied, the module returns the seeker's current structured data, if neither of them fails, it returns null.

6.4. ASSIGN SEEKER WEIGHTAGE MODULE

- In this module that receives an array of Compulsory Constraints as well as structured seeker data. Then, after first defining each constraint's weightage to a desirable value, we iterate over the compulsory constraint array, which contains the SL.NO, and in accordance with its contents, the weightage is changed to the value that represents the compulsory weightage value, returning null.

6.5. ASSIGN SEEKER SCORE MODULE

- In this module's assignment, we used the pre-defined machine learning approach (KNN- algorithm) from the software that contains the training data. With the help of this, we automatically determine a score based on the preferences of the seeker and, in the end, determine the score for those constraints.

6.6. CHECK JOB MATCHING MODULE

- This module passes structured data that involves the job and seeker, and then checks, if a constraint is given as a compulsory, then it is checked to see if the expected seeker constraint is different from the expected constraint of the talent acquisition team. If both conditions are satisfied, then the recommendation percent will be set to 0, but if any of them fails, the module will move on to the next constraint.

6.7.GET JOB RECOMMENDATION PERCENT MODULE

- In this module, the recommendation percentage is calculated based on a set of constraints between job requirements and seeker preferences. The calculation begins by initializing the numerator to 0 and the denominator to the sum of the products of all constraint scores and their corresponding weightages.
- The module then iterates through each constraint, comparing the expected constraint of the job to the expected constraint of the seeker. If the constraint is satisfied, the numerator is incremented by the product of the constraint weightage and its score. This process is repeated for all the constraints until they have all been evaluated.
- After evaluating the constraints, the module checks if the value returned by the function “call within fun” is equal to 1. If so, the job is returned to the seeker with the recommended percentage. Alternatively, if the function call is not equal to 1, the seeker to job recommendation percentage is calculated using the "get seeker recommendation percent" module.
- Finally, the net recommendation percentage is computed as 0.7 times the job to seeker recommendation percentage plus 0.3 times the seeker to job recommendation percentage. This net recommendation percentage is then returned as the result of the module.

6.8. RECOMMEND JOBS MODULE

- The structured data of the seeker is passed as an input to this module's recommend jobs; this module now also accepts data from the “companies.dat”. Once the dat file has been extracted, each company is iterated through, first checking to see if they exist. If they do, then the internally matched jobs are incremented the job matching module is then called, and if the returned value is 1, the job recommendation percent is calculated and the current job's data is stored in the matched jobs array. examining the array of matched jobs, print the recommended percentage for that job, and then go on to the next job. At last, then go on to the next company.

6.9. CREATE COMPANY MODULE

- In this module, the inner structure that holds the job structure has a maximum of 15 jobs and is called Create Company. where we provide the company's name, password, and if they want to generate jobs before calling the create jobs module and, finally, return the company data.

6.10. CREATE JOB MODULE

- We simply require data from the user regarding the constraints, such as the job title, location, job kind, business type, anticipated age, qualification, experience, Exp CGPA, etc. Next, we get information about the number of compulsory constraints and which constraints are required to be provided in SL.NO order. Finally, we call the assign weightage and assign score modules and return the Job structure.

6.11.JOB LOGIN MODULE

- The Job name and password are passed through this module, and after getting the structured data from the file “Companies.dat” and reading it from beginning to end, we check to see if the Job name is available. If it is, we then check to if the password is correct. If both constraints are satisfied, the module returns the job’s current structured data, if neither of them fails, it returns null.

6.12. ASSIGN JOB WEIGHTAGE MODULE

- In this module that receives an array of Compulsory Constraints as well as structured job data. Then, after first defining each constraint's weightage to a desirable value, we iterate over the compulsory constraint array, which contains the SL.NO, and in accordance with its contents, the weightage is changed to the value that represents the compulsory weightage value, returning null.

6.13. ASSIGN JOB SCORE MODULE

- In this module's assignment, we used the pre-defined machine learning approach (KNN- algorithm) from the software that contains the training data. With the help of this, we automatically determine a score based on the preferences of the talent acquisition team and, in the end, determine the score for those constraints.

6.14. CHECK SEEKER MATCHING MODULE

- This module passes structured data that involves the job and seeker, and then checks, if a constraint is given as a compulsory, then it is checked to see if the expected job constraint is different from the expected constraint of the seeker. If both conditions are satisfied, then the recommendation percent will be set to 0, but if any of them fails, the module will move on to the next constraint.

6.15.GET SEEKER RECOMMENDATION PERCENT MODULE

- In this module, the recommendation percentage is calculated based on a set of constraints between seeker requirements and job preferences. The calculation begins by initializing the numerator to 0 and the denominator to the sum of the products of all constraint scores and their corresponding weightages.
- The module then iterates through each constraint, comparing the expected constraint of the seeker to the expected constraint of the job. If the constraint is satisfied, the numerator is incremented by the product of the constraint weightage and its score. This process is repeated for all the constraints until they have all been evaluated.
- After evaluating the constraints, the module checks if the value returned by the function “call within fun” is equal to 1. If so, the seeker is returned to the job with the recommended percentage. Alternatively, if the function call is not equal to 1, the job to seeker recommendation percentage is calculated using the "get job recommendation percent" module.
- Finally, the net recommendation percentage is computed as 0.7 times the seeker to job recommendation percentage plus 0.3 times the job to seeker

recommendation percentage. This net recommendation percentage is then returned as the result of the module.

6.16. RECOMMEND SEEKERS MODULE

- The structured data of the job is passed as an input to this module's recommend seekers; this module now also accepts data from the “Seekers.dat”. Once the dat file has been extracted, each seeker is iterated through, first checking to see if they exist. If they do, the seeker matching module is then called, and if the returned value is 1, the seeker recommendation percent is calculated and the current seeker's data is stored in the matched seekers array. examining the array of matched seekers, print the recommended percentage for that job, and then go on to the next seeker.

7. IMPLEMENTATION

7.1. DATA ORGANIZATION

7.1.1. SEEKER

Construct Used: Structures stored in Files

Contents of the Structure - Seeker:

Data	**Data Type**
Seeker Name	String
Password	String
Age	Integer
Phone Number	Long Long Integer
Email	String
Qualification	String
Experience	Integer
College	String
Tenth Percentage	Float
Twelfth Percentage	Float
CGPA	Float
Expected Salary	Integer
Salary Score and Weightage	Integer
Expected Location	String
Location Score and Weightage	Integer
Expected Job Type	String
Job Type Score and Weightage	Integer
Expected Company Type	String
Company Type Score and Weightage	Integer
Expected Shift	Character
Shift Score and Weightage	Integer
Work From Home Status	Character
Work From Home Score and Weightage	Integer
No of Applied Jobs	Integer
No of Newly Appeared Jobs	Integer
Applied Jobs	Array of AppliedJobs – User Defined Structure

Reason for Choosing this type of Construct:

Seeker is basically a collection of heterogenous data (dissimilar data types). So, we use a structure to store a seeker. After getting the details of the seeker from the user, we store them in Seekers.dat file to store the data permanently.

7.1.2. APPLIED JOBS

Construct Used: Structures

Contents of the Structure – Applied jobs:

Data	Data Type
Company Name	String
Job	String
Status	Integer

Reason for choosing this type of Construct:

Applied jobs is used to store the information of the jobs that the seeker has applied for. So, we see, it stores heterogenous data. Therefore, we use structures.

7.1.3. COMPANY

Construct Used: Structures stored in Files

Contents of the Structure – Company:

Data	Data Type
Company Name	String
Password	String
No of Jobs	Integer
Jobs	Array of Job – User defined Structure

Reason for Choosing this type of Construct:

Company will contain a collection of data related to the company. A Company will contain jobs inside it which is another user defined structure. We see the data for a Company is heterogenous. So, structures are the best suited to store this type of data. After getting the data for structures, we store the structure in Companies.dat file to store them permanently.

7.1.4. JOB

Construct Used: Structure

Contents of the Structure – Job:

Data	**Data Type**
Job	String
Salary	Integer
Location	String
Job Type	String
Company Type	String
Work From Home Status	Character
Shift	Character
Shift Score and Weightage	Integer
Expected Age	Integer
Age Score and Weightage	Integer
Expected Experience	Integer
Experience Score and Weightage	Integer
Expected Qualification	String
Qualification Score and Weightage	Integer
Expected Tenth Percentage	Float
Tenth Percent Score and Weightage	Integer
Expected Twelfth Percentage	Float
Twelfth Percent Score and Weightage	Integer
Expected CGPA	Float
CGPA Score and Weightage	Integer

Reason for choosing this type of Construct:

Jobs is used to store the information of jobs under a company. We have so many data regarding jobs that are of dissimilar data type. Therefore, it is best suited to use structures to store the information of the job.

7.1.5. COMPANY APPLICATION

Construct Used: Structures stores in Files

Contents of the Structure – Company Application:

Data	**Data Type**
Company	String
No of Applicants	Integer
Newly Appeared Applicants	Integer
Applications	Array of Applications – User Defined Structure

Reason for choosing this type of Construct:

A Company Application is a collection of data which stores the data of applications for a particular company. These data are heterogenous. Therefore, structures can be used to store the data. After storing the information in the structure, we store the structure in Applications.dat to store the data permanently.

7.1.6. APPLICATION

Construct Used: Structures

Contents of the structure – Application:

Data	Data Type
Applicant	String
Job	String

Reason for choosing this type of Construct:

Applications stores the information of the Applicant that is the applicant Name and the Job he has applied for. Though the data are homogenous, it is convenient to store and access data if we store them in a structure.

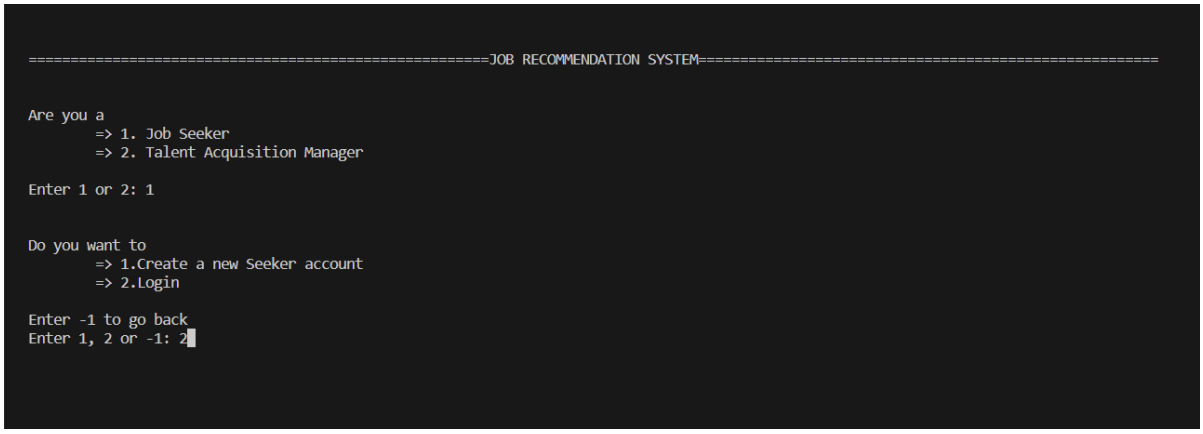
7.2. EXTERNAL LIBRARIES AND APIs

sha256.h	--	used to hash a string in SHA256 hashing algorithm using the SHA256() function of the library in which the string to be hashed is passed as a parameter
score.h	--	contains user modified internet-based machine learning functions for KNN Algorithm

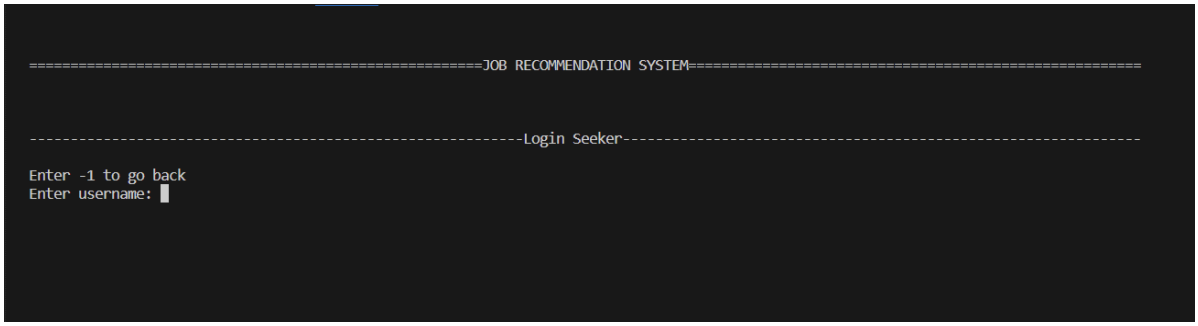
7.3. USER INTERFACE DESIGN

The project is an application that runs on the terminal. To provide a good look and feel to the users, the outputs of the program is well formatted. Also, to provide a good flow or to avoid confusion to the users, we refresh the terminal, that is we remove the previous module's contents from the terminal and display only the current module's contents.

Home Page:



Login Page:



The contents of the home page are removed and only the contents of login Seeker module are displayed

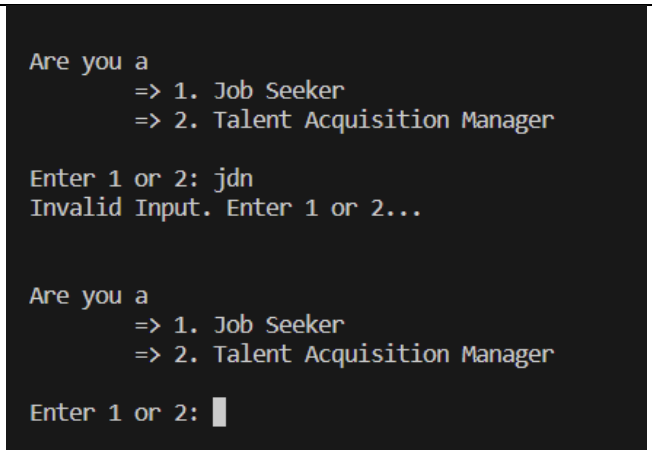
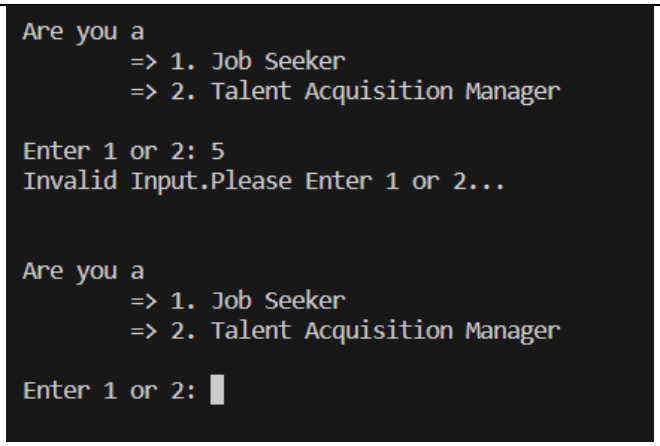
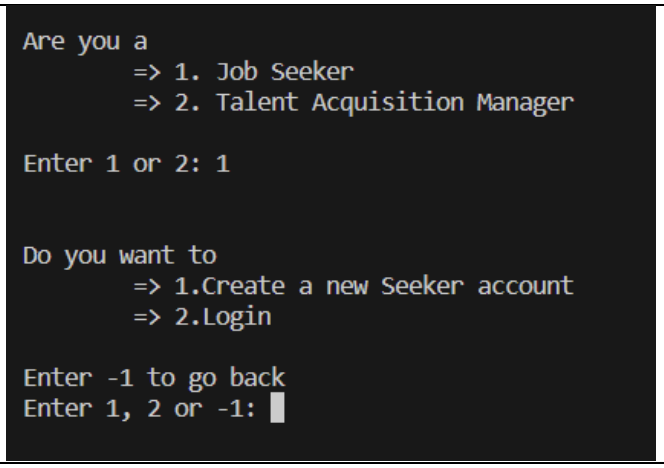
7.4. PLATFORMS USED FOR CODE DEVELOPMENT

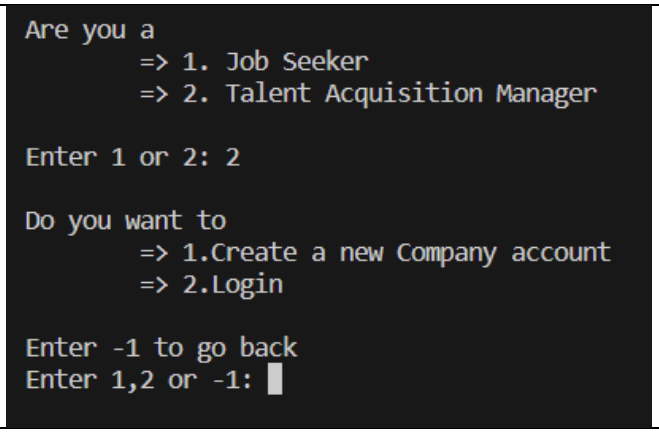
VS Code was used to develop and implement the Project using C language. Outputs were also recorded in the terminal of VS Code.

8. TEST CASES

8.1. MAIN MENU

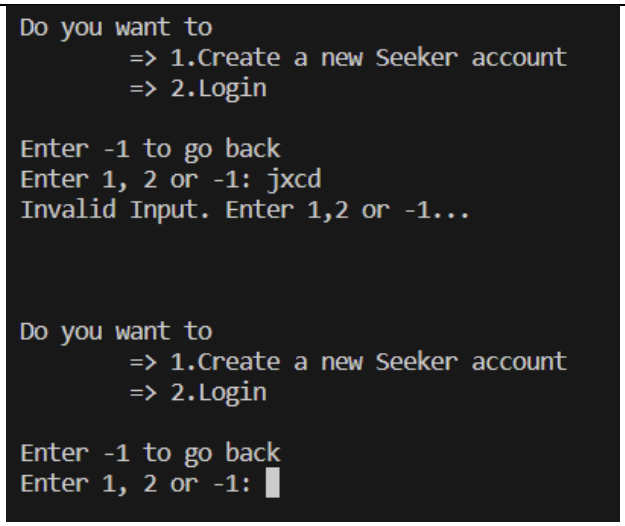
8.1.1. Are you a Job Seeker or Talent Acq. Manager

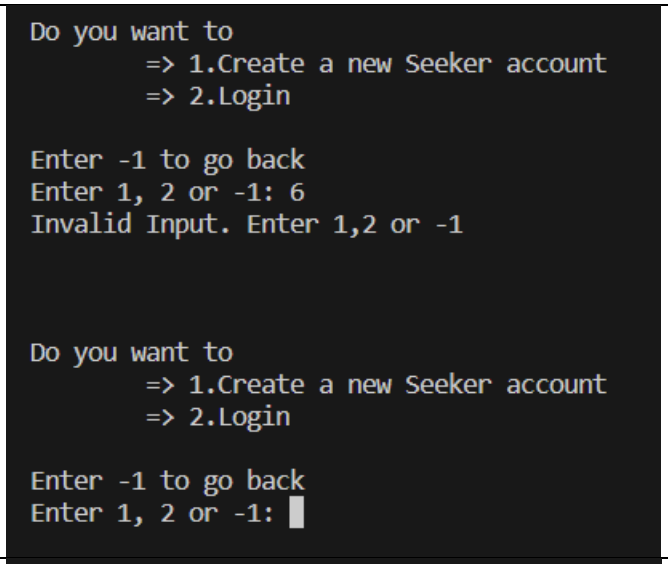
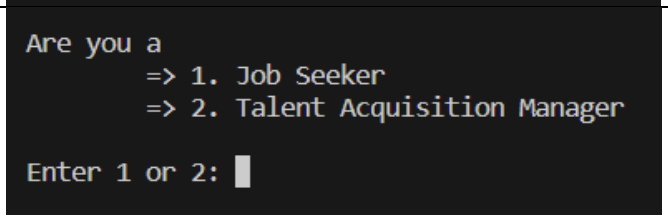
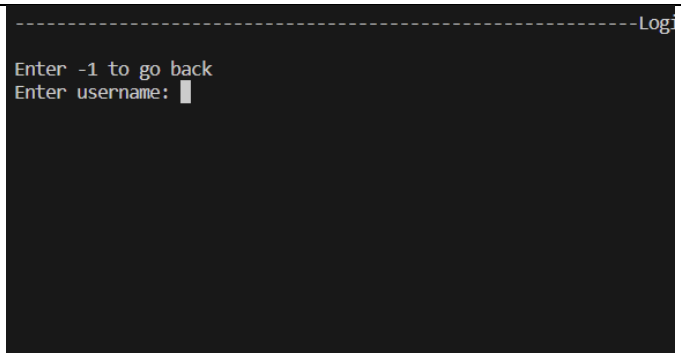
Input	Expected Output	Actual Output	Validation
Any sort of string input	Warning Prompt: Invalid Input		Successful
Integer other than 1 or 2	Warning Prompt: Invalid Input		Successful
1	Move to Job Seeker page		Successful

2	Move to Company Page		Successful

8.2. SEEKER MENU

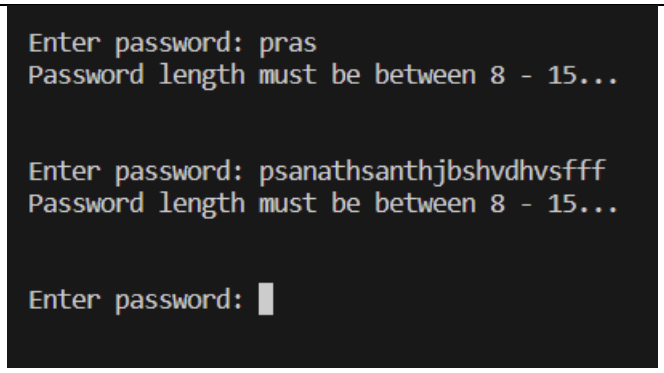
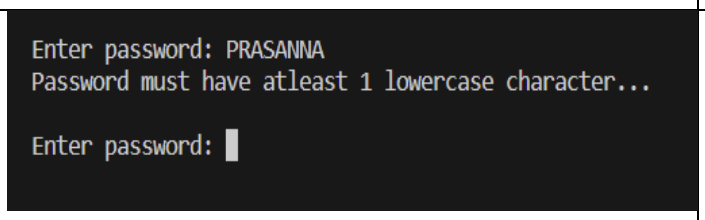
8.2.1. Create Account or Login

Input	**Expected Output**	**Actual Output**	**Validation**
Any sort of string input	Warning Prompt: Invalid Input		Successful

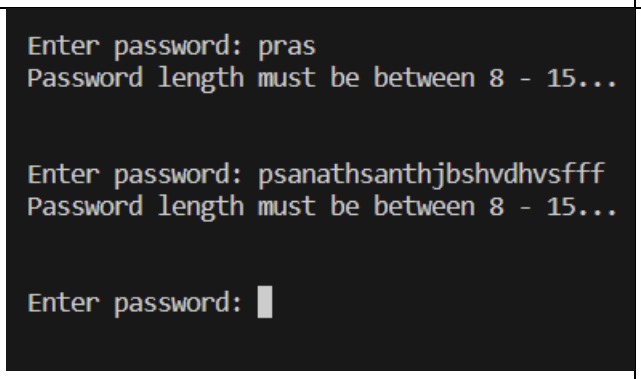
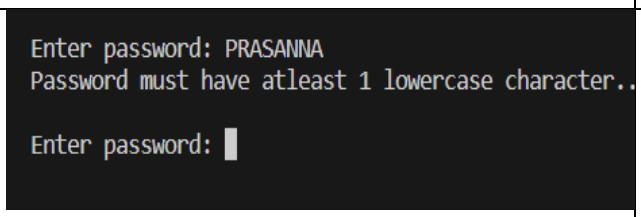
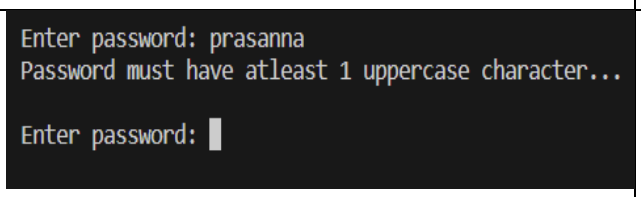
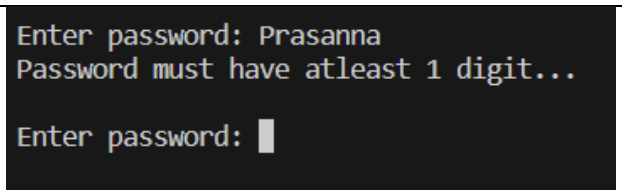
Integer other than 1, 2, or -1	Warning Prompt: Invalid Input		Successful
-1	Move to main menu		Successful
1	Move to create seeker module		Successful
2	Move to login Seeker Module		Successful

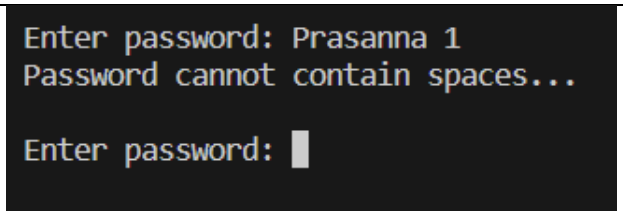
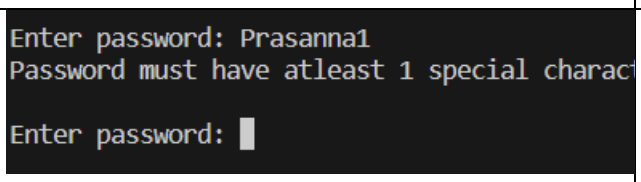
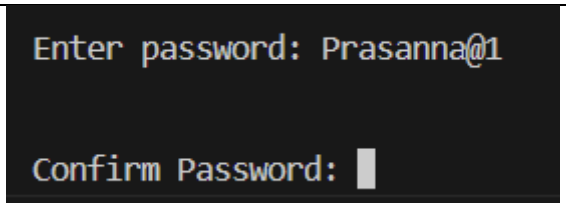
8.3. CREATE SEEKER

8.3.1. Seeker Name

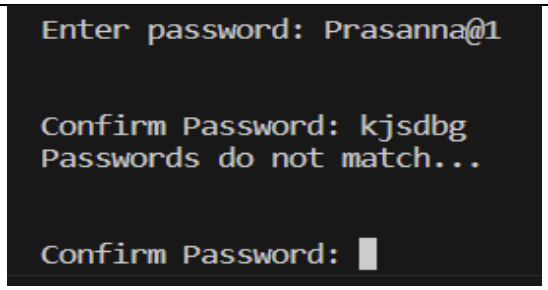
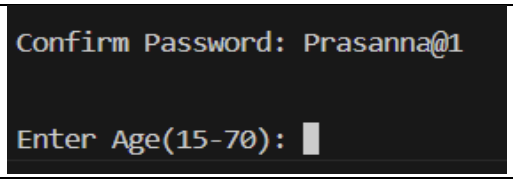
Input	Expected Output	Actual Output	Validation
Input Name already exists	Seeker Name Already Exists. Do you want to login Instead?		Successful
Input Name doesn't exist	Seeker Name is stored		Successful

8.3.2. Password

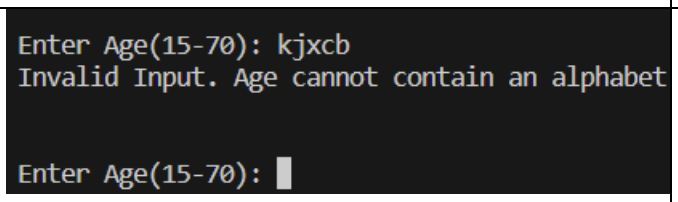
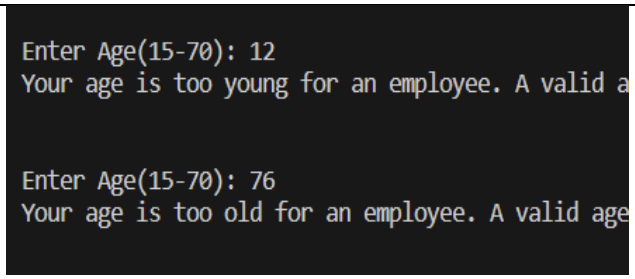
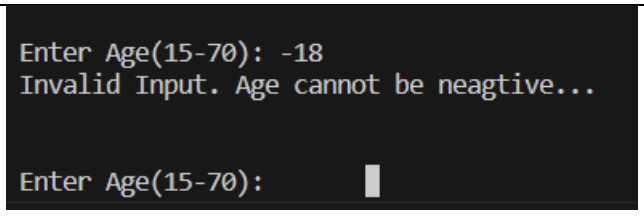
Input	Expected Output	Actual Output	Validation
Less than 8 characters or more than 15 characters	Warning Prompt: Password length must be between 8 - 15		Successful
No Lowercase characters	Warning Prompt: Password must have at least 1 lowercase character.		Successful
No Uppercase character	Warning Prompt: Password must have at least 1 uppercase character		Successful
No Digits	Warning Prompt: Password must have at least 1 digit		Successful

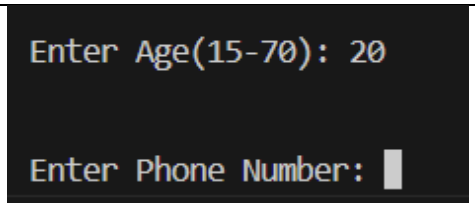
Spaces	Warning Prompt: Password cannot contain spaces		Successful
No Special Characters	Warning Prompt: Password must have at least 1 special character		Successful
All the constraints satisfied	Password accepted		Successful

8.3.3. Confirm Password

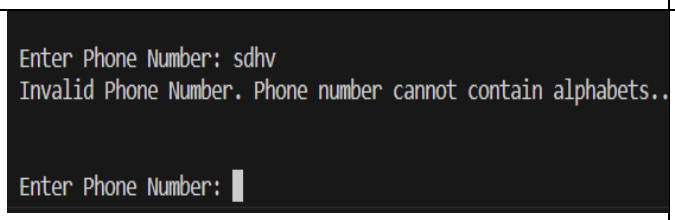
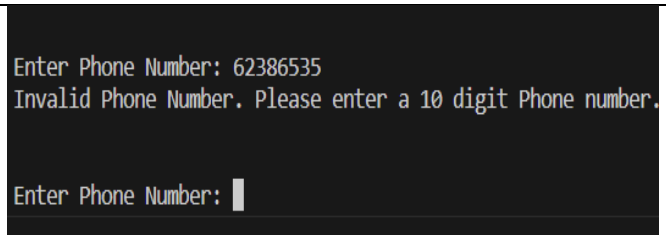
Input	**Expected Output**	**Actual Output**	**Validation**
Confirm Password! = Password	Warning Prompt: Passwords do not match		Successful
Confirm Password == Password	Password gets stored		Successful

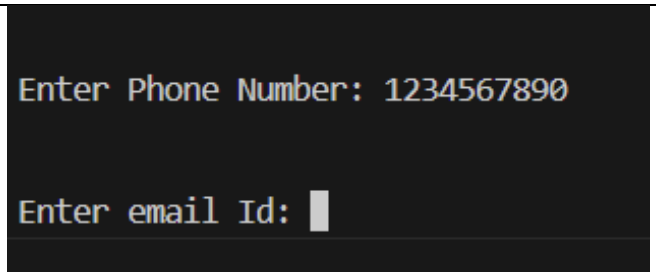
8.3.4. Age

Input	Expected Output	Actual Output	Validation
Any sort of string input	Warning Prompt: Invalid Input. Age cannot contain an alphabet or special character.		Successful
Integer Less than 15 or greater than 70	Warning Prompt: Your age is too young or old for an employee . A valid age range between 15 - 70...		Successful
Negative Integer	Warning Prompt: Invalid Input. Age cannot be negative.		Successful

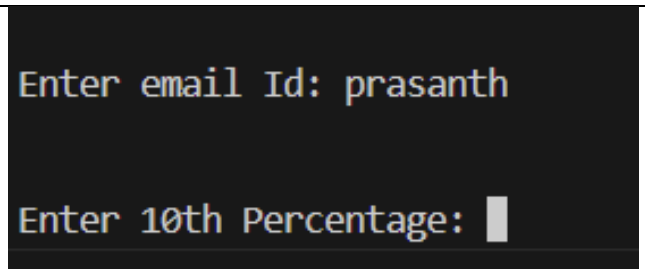
Integer between 15 - 70	Age is stored		Successful

8.3.5. Phone Number

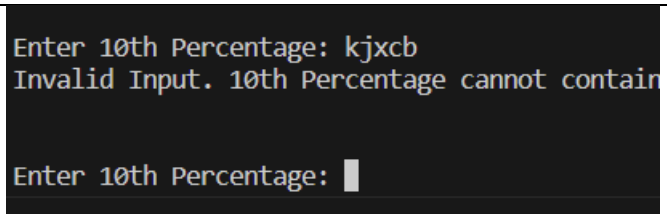
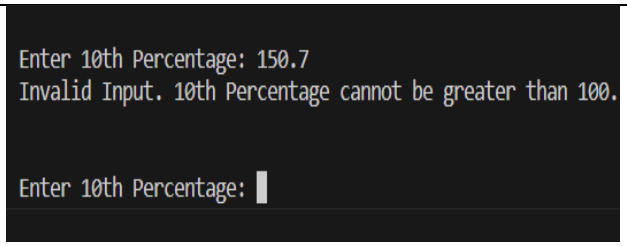
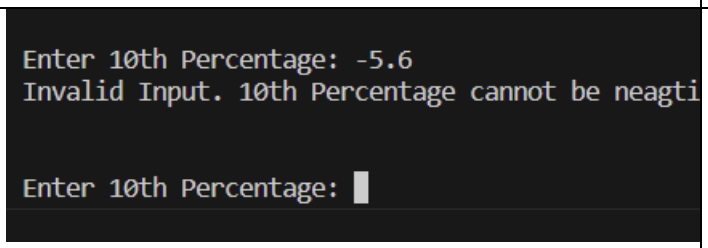
Input	**Expected Output**	**Actual Output**	**Validation**
Any sort of string input	Warning Prompt: Invalid Input. Age cannot contain an alphabet or special character.		Successful
Not 10 digits	Warning Prompt: Your age is too young or old for an employee . A valid age range between 15 - 70...		Successful

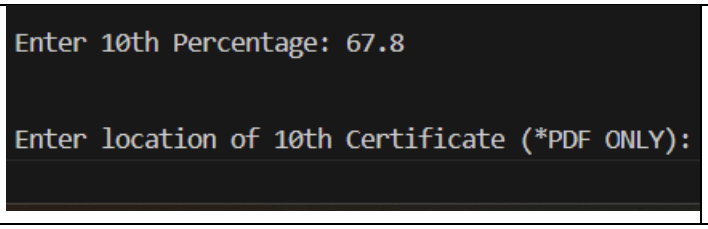
Any 10-digit Number . Even if its a non-existent phone number	Phone Number is stored.		Not Successful

8.3.6. Email Id

Input	**Expected Output**	**Actual Output**	**Validation**
Any form of string even if it's not a valid mail id	Email gets stored		Not Successful

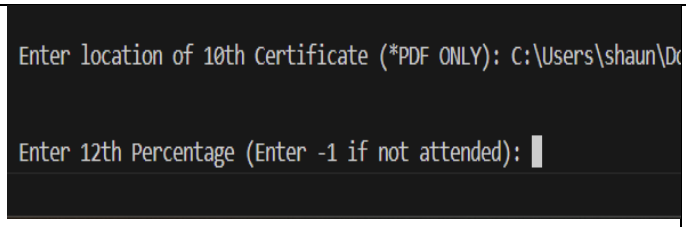
8.3.7. 10th Percentage

Input	Expected Output	Actual Output	Validation
Any sort of string input	Warning Prompt: Invalid Input. 10th Percentage cannot contain alphabets or special characters		Successful
Float value greater than 100	Warning Prompt: Invalid Input. 10th Percentage cannot be greater than 100		Successful
Negative Float value	Warning Prompt: Invalid Input. 10th Percent cannot be negative.		Successful

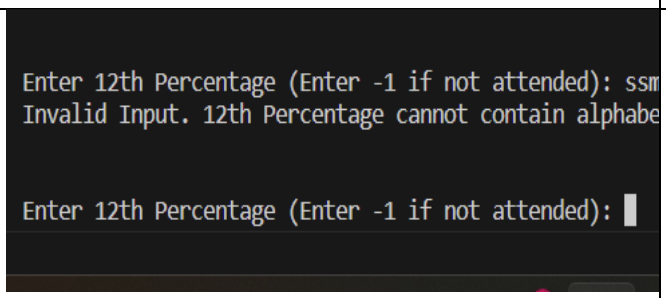
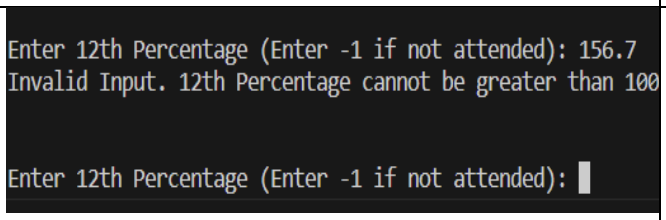
Float value between 0 to 100	10th percent is stored		Successful

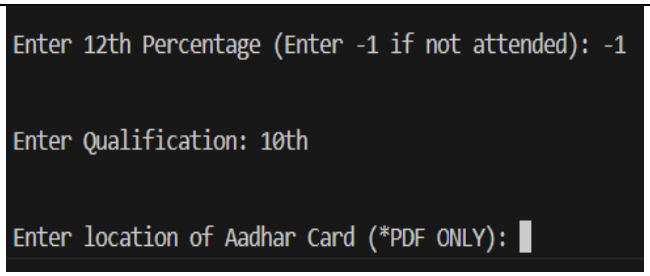
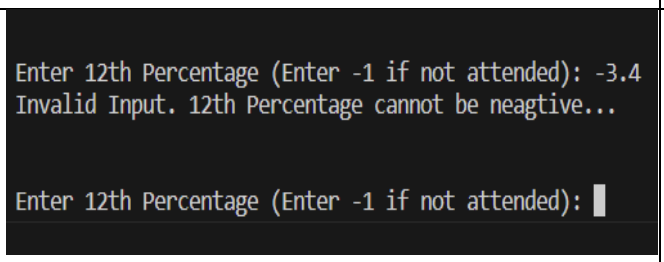
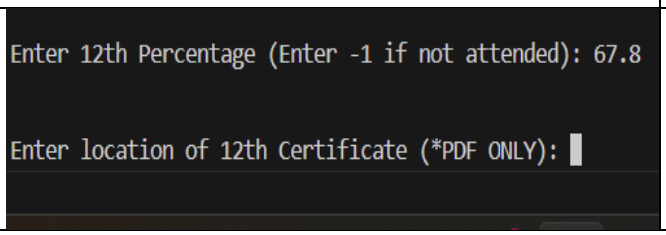
8.3.8. 10th Certificate Location

Input	**Expected Output**	**Actual Output**	**Validation**
Non pdf documents	Warning Prompt: Invalid file format. Please upload a PDF file...		Successful
File is not present in that location	Warning Prompt: Invalid file location		Successful

Pdf document with correct file location	Pdf document is stored in the documents folder.		Successful

8.3.9. 12th Percentage

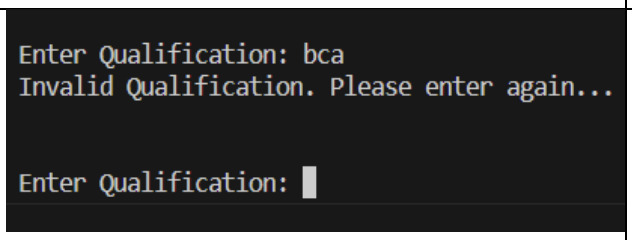
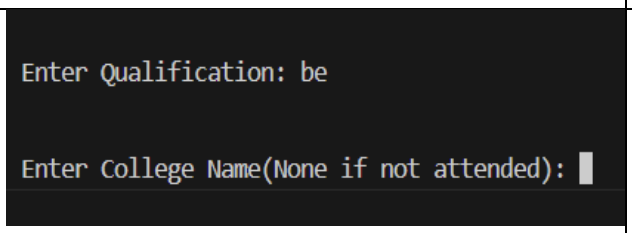
Input	Expected Output	Actual Output	Validation
Any sort of string input	Warning Prompt: Invalid Input. 12th Percentage cannot contain alphabets or special characters		Successful
Float value greater than 100	Warning Prompt: Invalid Input. 12th Percentage cannot be greater than 100		Successful

-1	12th Certificate is not asked, College and CGPA is automatically set as None and -1		Successful
Negative Float value	Warning Prompt: Invalid Input. 12th Percent cannot be negative.		Successful
Float value between 0 to 100	12th percent is stored		Successful

8.3.10. 12th Certificate Location

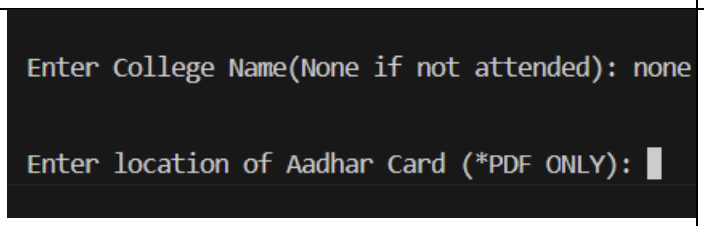
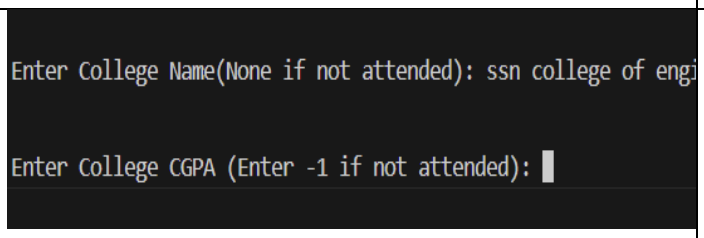
*Same as 10th Certificate Location (7.3.7)

8.3.11. Qualification

Input	Expected Output	Actual Output	Validation
Other than the qual. that are specied in the program	Warning Prompt: Invalid Qualification. Please enter again		Successful
Qual. that are specified in our program	Qualification is stored		Successful

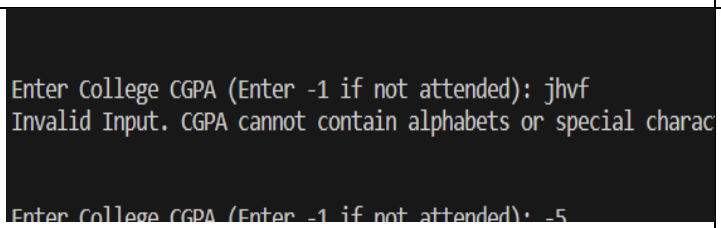
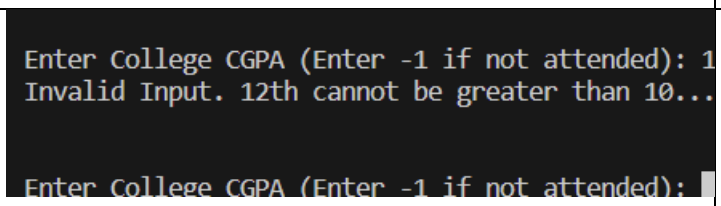
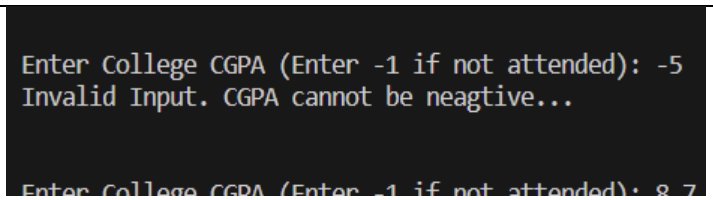
NOTE: The characters in the qualification are automatically converted into uppercase characters

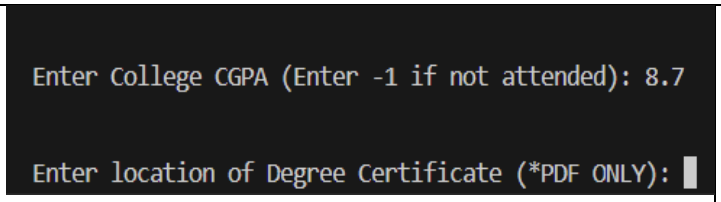
8.3.12. College

Input	Expected Output	Actual Output	Validation
None	CGPA is automatically taken as -1 and degree certificate is not asked		Successful
Any string even if it is not an actual college	College is stores		Not Successful

NOTE: The first letter of each word is automatically converted into uppercase and the rest into lowercase characters

8.3.13. CGPA

Input	Expected Output	Actual Output	Validation
Any sort of string input	Warning Prompt: Invalid Input. CGPA cannot contain alphabets or special characters		Successful
Float value greater than 10	Warning Prompt: Invalid Input. CGPA cannot be greater than 10		Successful
Negative Float value	Warning Prompt: Invalid Input. CGPA cannot be negative.		Successful

Float value between 0 to 10	CGPA is stored		Successful

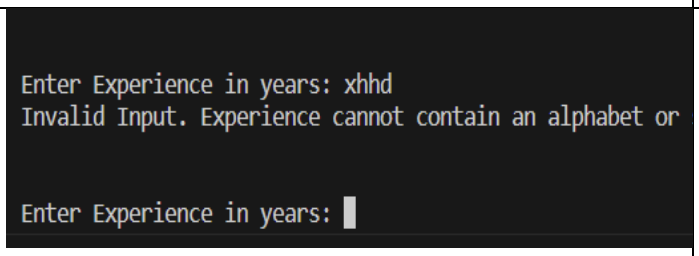
8.3.14. Degree Certificate Location

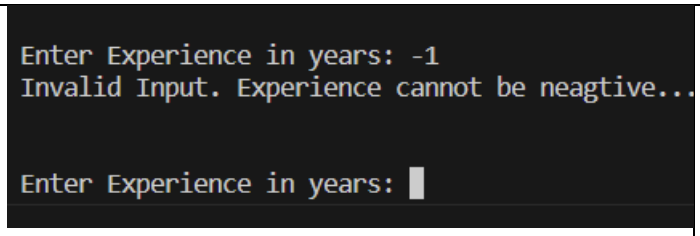
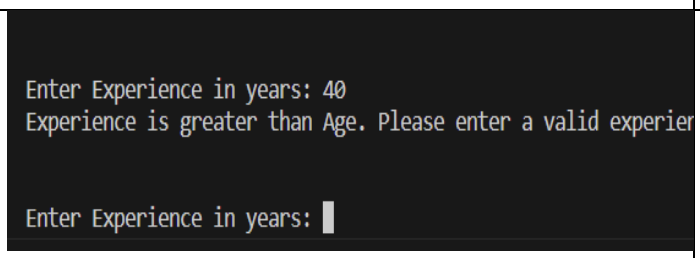
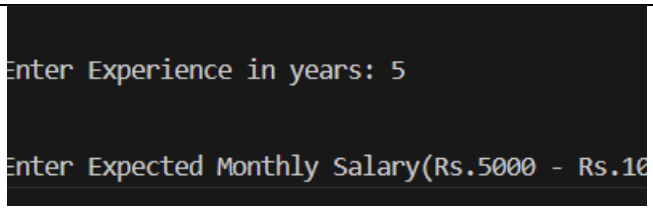
*Same as 10th Certificate Location (7.3.7)

8.3.15. AADHAR Certificate Location

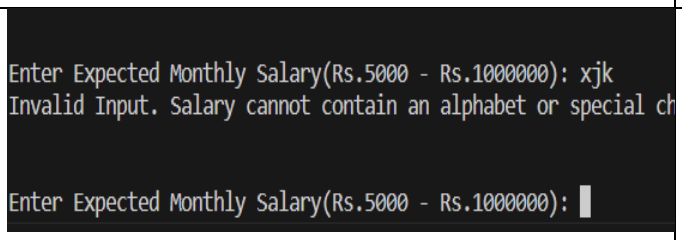
*Same as 10th Certificate Location (7.3.7)

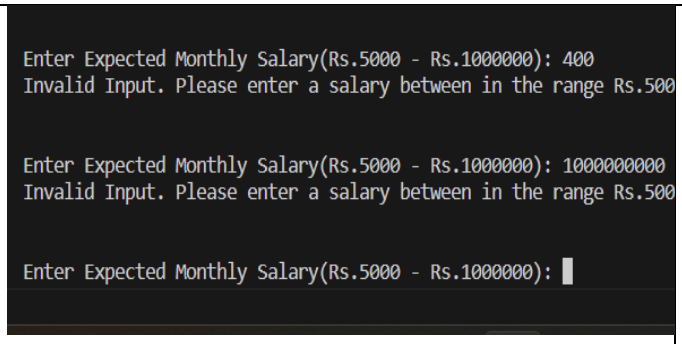
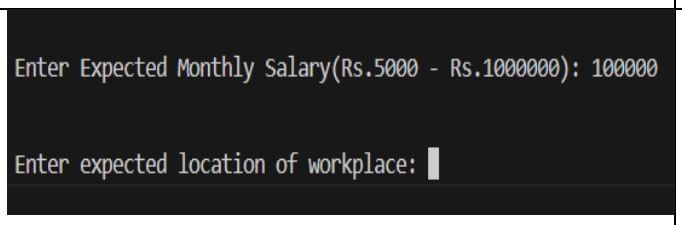
8.3.16. Experience

Input	**Expected Output**	**Actual Output**	**Validation**
Any form of string	Warning Prompt: Invalid Input. Experience cannot contain an alphabet or special character		Successful

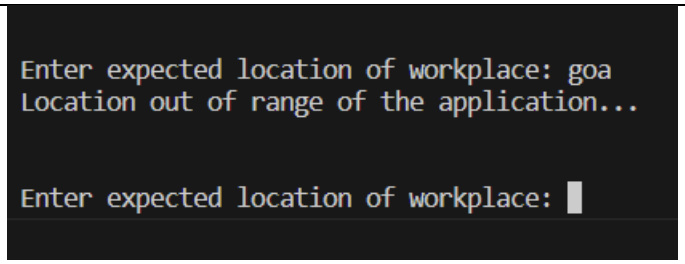
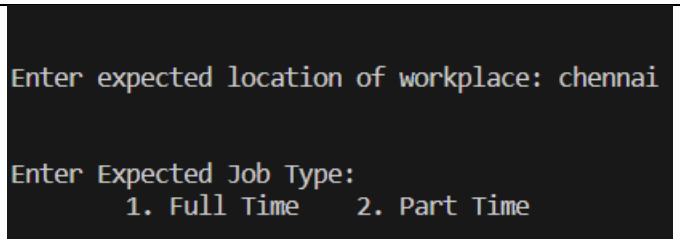
Negative Integer	Invalid Input. Experience cannot be negative		Successful
Integer greater than age	Experience is greater than Age. Please enter a valid experience.		Successful
Integer between 0 - Age	Experience is stored		Successful

8.3.17. Salary

Input	**Expected Output**	**Actual Output**	**Validation**
Any form of string	Warning Prompt: Invalid Input. Salary cannot contain an alphabet or special character		Successful

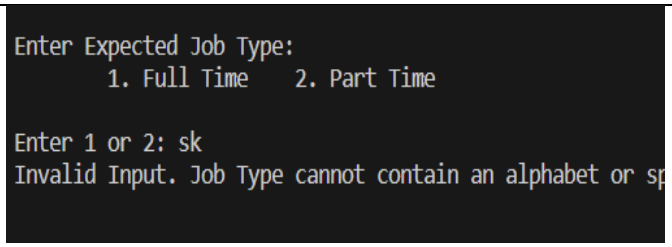
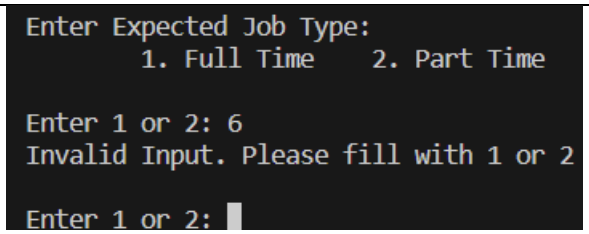
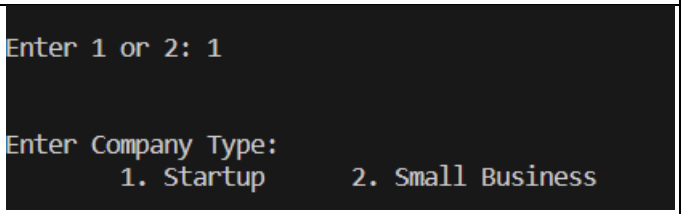
Integer less than 5000 or More than 1000000	Invalid Input. Please enter a salary between in the range Rs.5000 - Rs.1000000		Successful
Integer between 5000 - 1000000	Salary is stored.		Successful

8.3.18. Job Location

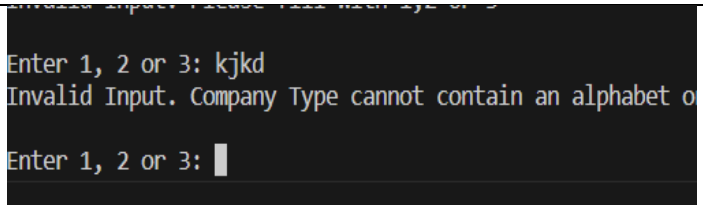
9. Input	Expected Output	Actual Output	Validation
Other than the cities that are specied in the program	Warning Prompt: Invalid Location. Please enter again		Successful
Cities that are specified in our program	Location is stored		Successful

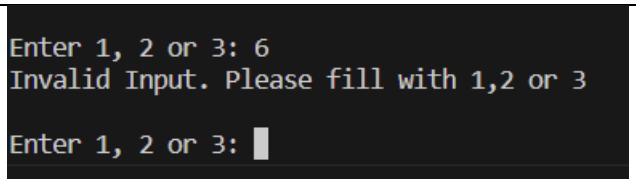
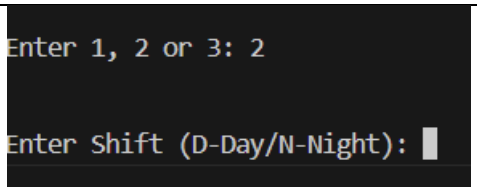
NOTE: The first character of the words is automatically converted into uppercase characters and the rest into lowercase characters

8.3.19. Job Type

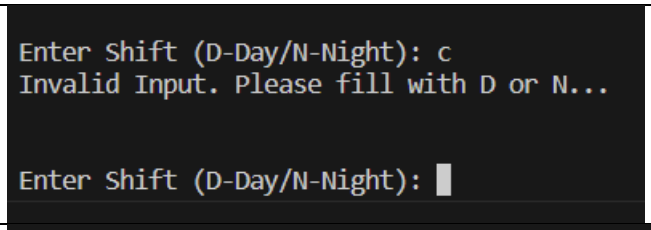
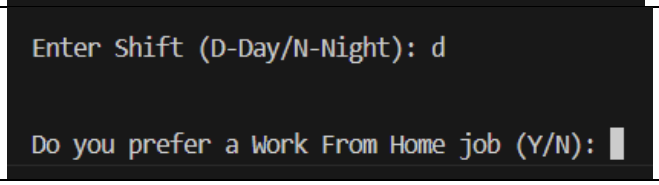
Input	Expected Output	Actual Output	Validation
Any form of string	Warning Prompt: Invalid Input.		Successful
Integer other than 1 or 2	Invalid Input. Please enter 1 or 2		Successful
1 or 2	Respective Job Type is stored		Successful

8.3.20. Company Type

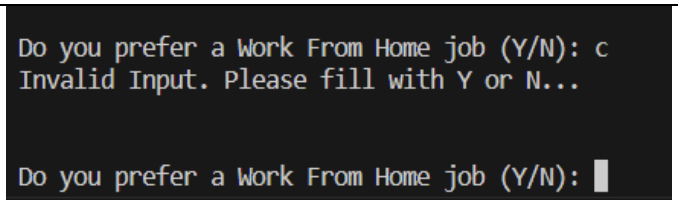
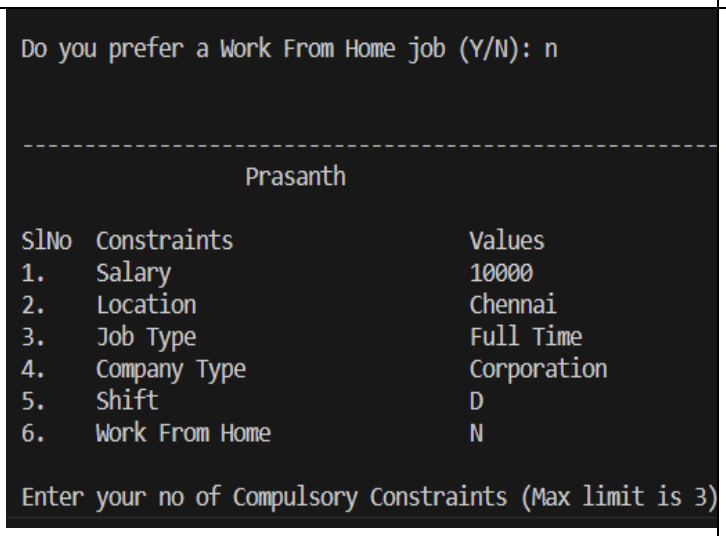
Input	Expected Output	Actual Output	Validation
Any form of string	Warning Prompt: Invalid Input.		Successful

Integer other than 1 ,2 or 3	Invalid Input. Please enter 1,2 or 3		Successful
1,2 or 3	Respective Company Type is stored		Successful

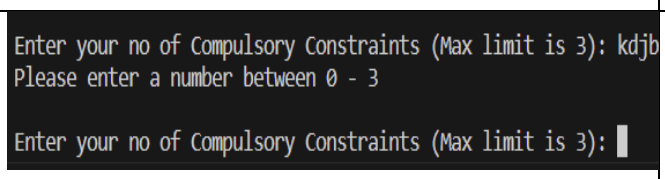
8.3.21. Shift

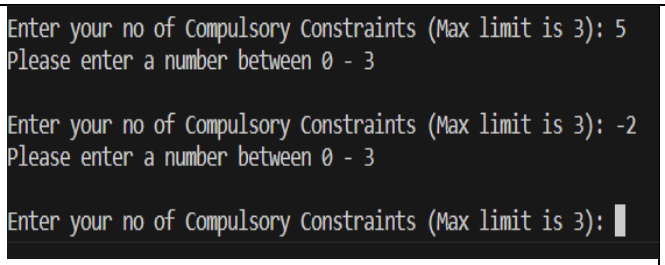
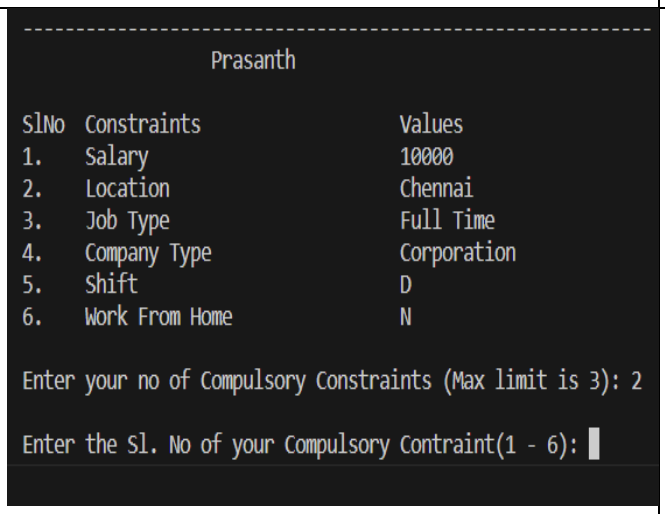
Input	**Expected Output**	**Actual Output**	**Validation**
Other than D and N	Warning Prompt: Invalid Input.		Successful
D or N	Respective Shift is stored		Successful

8.3.22. Work From Home

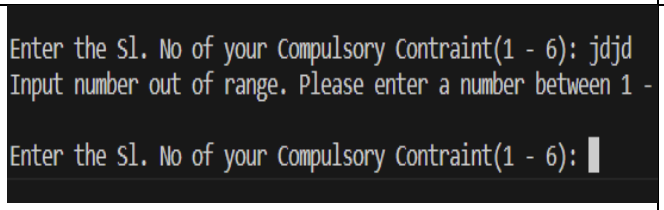
Input	Expected Output	Actual Output	Validation
Other than Y and N	Warning Prompt: Invalid Input.		Successful
Y or N	Respective Work from Home is stored		Successful

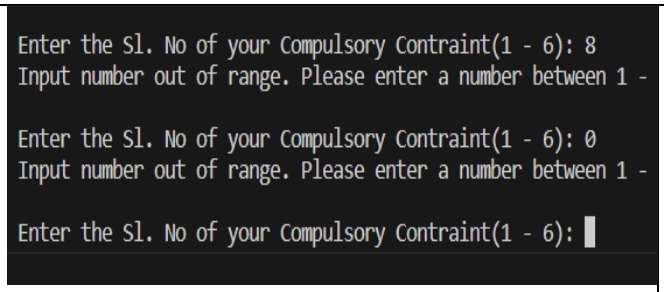
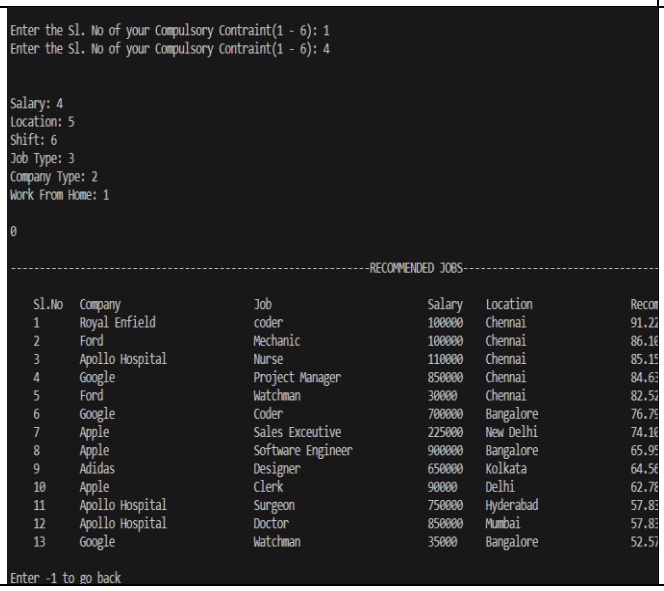
8.3.23. No Of Compulsory Constraints

Input	Expected Output	Actual Output	Validation
Any form of string	Warning Prompt: Invalid Input.		Successful

Integer less than 0 or More than MAX COMP CONST	Invalid Input.		Successful
Integer between 0 - MAX COMP CONST	No of Compulsory Constraints is stored		Successful

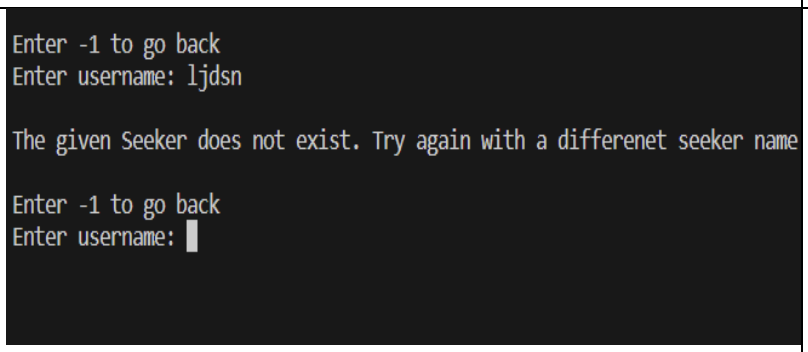
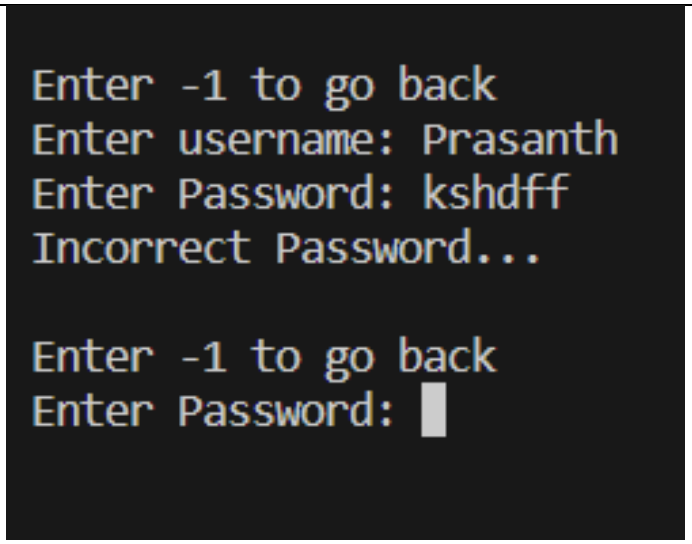
8.3.24. Compulsory Constraint

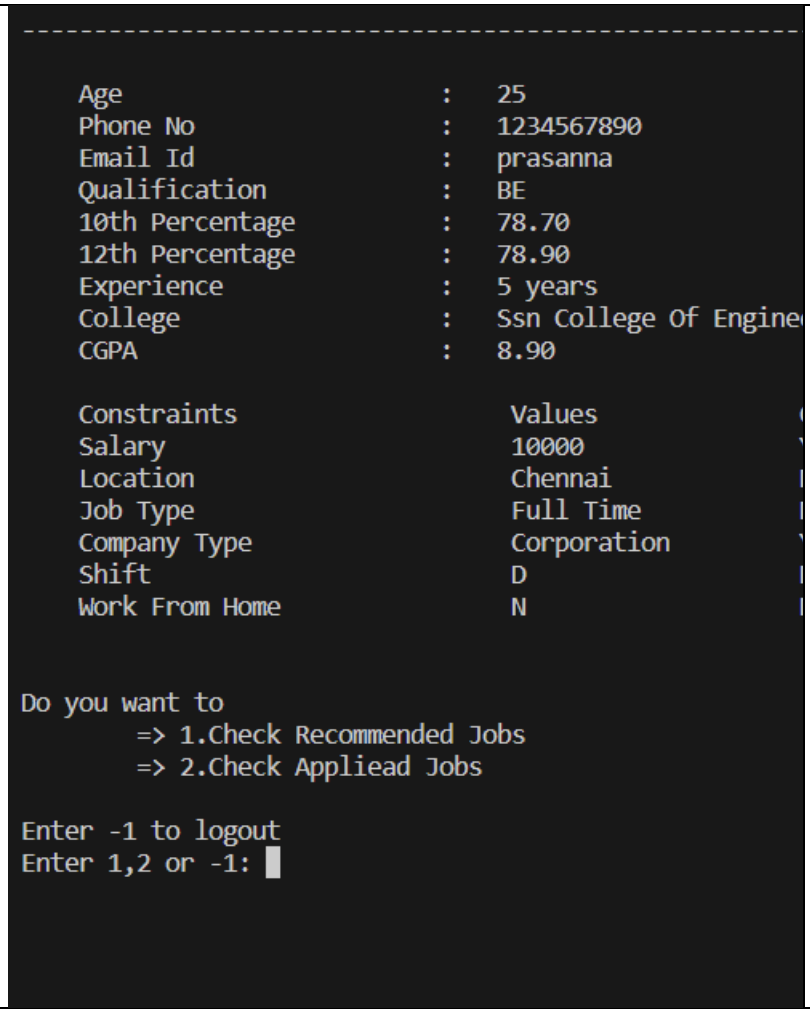
Input	**Expected Output**	**Actual Output**	**Validation**
Any form of string	Warning Prompt: Invalid Input.		Successful

Integer less than 0 or More than NO OF CONST	Invalid Input.		Successful
Integer between 0 - NO OF CONST	Jobs are recommended		Successful

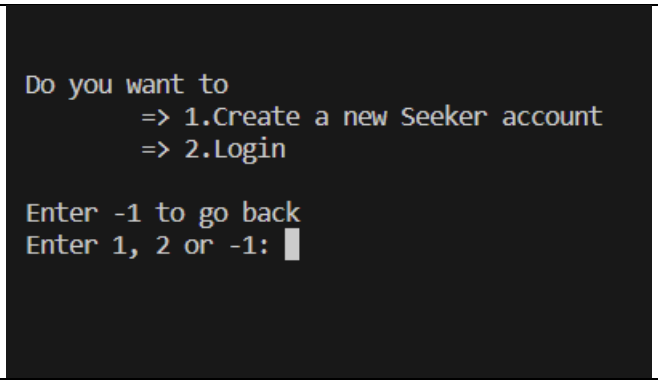
8.4. LOGIN SEEKER

8.4.1. Login Menu

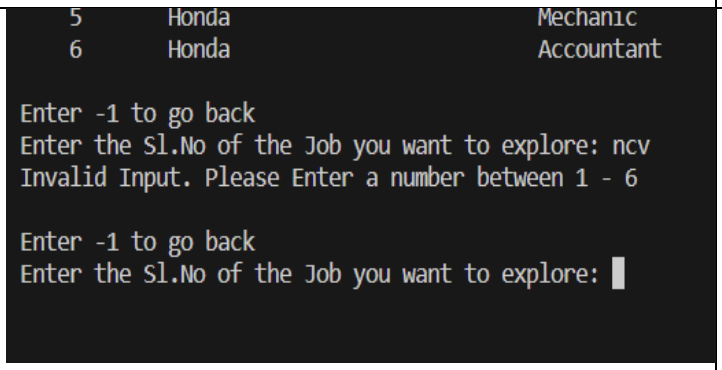
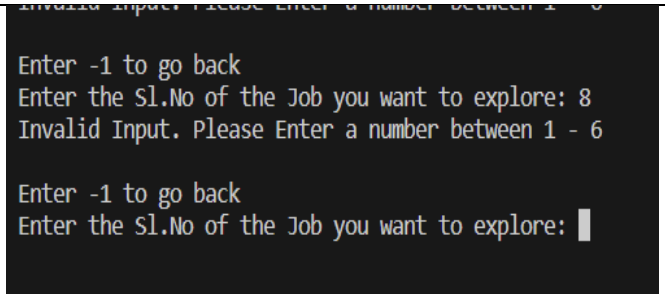
Input	Expected Output	Actual Output	Validation
Non-existent Seeker	Warning Prompt: Seeker Doesn't Exist		Successful
Existing Seeker and Wrong Password	Invalid Password		Successful

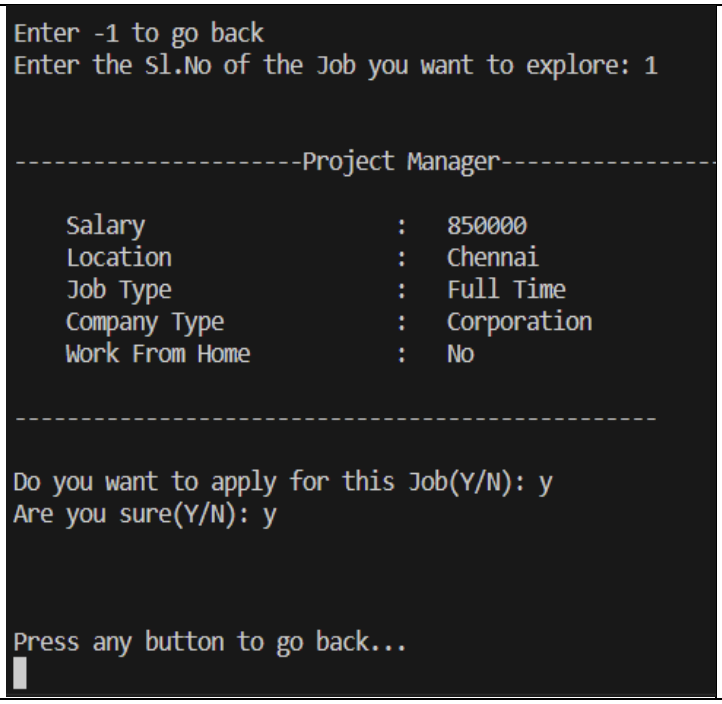
Existing Seeker and Correct Password	Seeker profile is shown		Successful

8.5. SEEKER PROFILE MENU

Input	Expected Output	Actual Output	Validation
1	Jobs are recommended		Successful
2	Applied Jobs list is shown		Successful
-1	Moves to Seeker Menu		Successful

8.6. RECOMMENDATION MENU

Input	Expected Output	Actual Output	Validation
Any form of string	Invalid Input		Successful
Beyond the range provided	Invalid Input		Successful

Within the limit given	Applies for the job		Successful

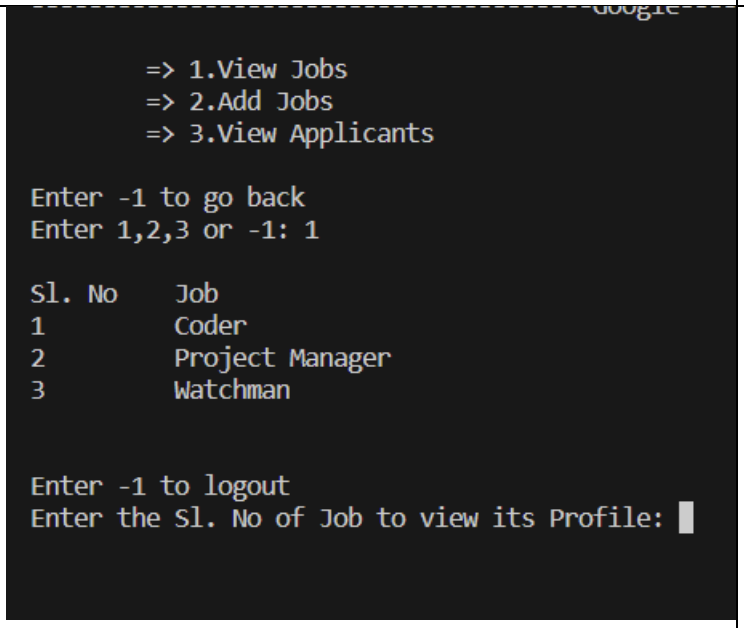
8.7. CREATE COMPANY AND JOB

*Same as CREATE SEEKER

8.8. LOGIN COMPANY

*Same as LOGIN SEEKER

8.9. COMPANY MENU

Input	Expected Output	Actual Output	Validation
1	Displays the jobs of the company		Successful
2	Create job module is called		Successful

3	Displays the applicants of the company		Successful

9. LIMITATIONS

- Scalability of the Locations and Qualifications is very low
- Cannot check if the phone number given exists or not
- Similarly, cannot check if the given Gmail is valid or not
- Cannot correctly compare Qualifications for matching

  Eg: Talent acq. Team's compulsory constraint is 10^{th} grade for qualification, and since we compare only string literals a seeker whose qualification is BE will not satisfy while computing the recommendation percentage
- No option of forget password
- No option for editing the already existing data

10. OBSERVATIONS

10.1. Social Perspectives

Inclusivity:	The flow of the app is solely based on the user preferences and the app gives a lot of importance to the preference of the user by getting inputs
Simplicity:	Our app does not use any complex language that the user might not be able to understand. Everything is in terms of simple English so that the user will have easy access to the app

10.2. Legal Perspective

Hashed Password:	The passwords inputted by the user are hashed using SHA256 hashing algorithm to increase the security aspect of our app
Document: Verification	The app demands documents including AADHAR, 10th certificate, etc to the verify the trueness of the information provided by the user. These documents will be manually verified

10.3. Environmental Perspective

Reduced Code:	We have used modularity techniques and made our code efficient and effective thereby having more functionality for less code thus reducing the energy required which is evident when the number of users increase on a vast number

11. LEARNING OUTCOMES

- Learnt File Handling in detail
- How to integrate Machine Learning in C
- Multi file compilation in C
- Modularity in C
- Make files
- Analysis and design reporting before implementing a project
- Problem solving and algorithm designing
- Better knowledge in C overall

12. REFERENCES

- For Flow Charts

https://www.lucidchart.com/pages/examples/flowchart-maker

- Machine Learning Algorithm (KNN)

https://devcoons.com/k-nearest-neighbours-algorithm-in-c/

- Encryption of Password (SHA-256)

https://github.com/B-Con/crypto-algorithms/blob/master/sha256.c

- What is KNN

https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4