KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Al-Imam Mohammad University
College of Computer & Information Sciences

المملكة العربية السعودية
وزارة التعليم العالي
جامعة الإمام محمد بن سعود الإسلامية
كلية علوم الحاسب والمعلومات

<u>**Second term 1441/2020**</u>

Software Engineering (CS- 310)
BSCS- Section: 171

Project-Phase No: 02
Automatic Jobs Candidates Selection System (AJCSS)
(Design)

Submitted By
Sultan Attaf Al-Salmi (439013826) – Coordinator
Abdulaziz Derham Asseri (439011531)
Musaad Mubarak Alhammami (439016695)
Yasser Ahmed Almuhaidib (439013620)
Auodh Mohmmed AL-Qahtani (435032042)

Supervisor

Professor Sultan S. Alqahtani

Date: 21/3/2020

Table of Contents

Abstract:

This design document shows the architectural design of the AJCSS. It contains graphical representation about classes and functions, and the representation of requirements is shown in a way that reflects the purpose of the system. Also, it gives an example of how the software will be like.

1. Introduction

Here we will give a scope description of everything included in this document.

AJCSS is an Automatic Jobs Candidates Selection System. AJCSS software will allow users to store vacant jobs' and jobs' seekers data(candidates). It also allows users to apply matching process between jobs and candidates and generate a report of the matched candidates. AJCSS will not perform hiring or recruitment process. AJCSS will automate the selection of the most suitable unprocessed candidates for a vacant job. It will reduce the time and cost the manual process. It will save HR managers and members time and provide what is so-called recruitment's equal opportunity. AJCSS will allow users to apply a matching criterion based on the comparable attributes of jobs and candidates. Such criteria will consider multiple attributes for matching from different data types and values (integers and Boolean values). The criteria will apply exact matching between some attributes and inexact between some others. If exact match failed then no matching found, where in the inexact case, a match is found if attribute's value of a candidate is greater than or equal to the same attribute's value of a job. The matching process will aggregate the exact and inexact matches as a value of 10 points per attribute. Exact match is either 0 or 10 points while inexact is a range from 0 to 10. Boolean attributes are preferences and will not affect the matching process. The result of the matching process will produce a list of matched candidates -if exists- sorted by the highest aggregated percentage of a matched candidate. The matched list will be sent as a report to the HR manager. The status of the selected job for matching will be updated as occupied and the status of the matched candidates will be updated as processed. Regarding the progress that has been made since the SRS document, we have been designing the graphical user interface (GUI) of the application. As well the system model design and the architectural pattern.

2. High Level and Medium Level Design

2.1 System Level Diagram

The system level diagrams are: system's interaction model (Use-case diagram) and system's architectural pattern (pipe and filter pattern).
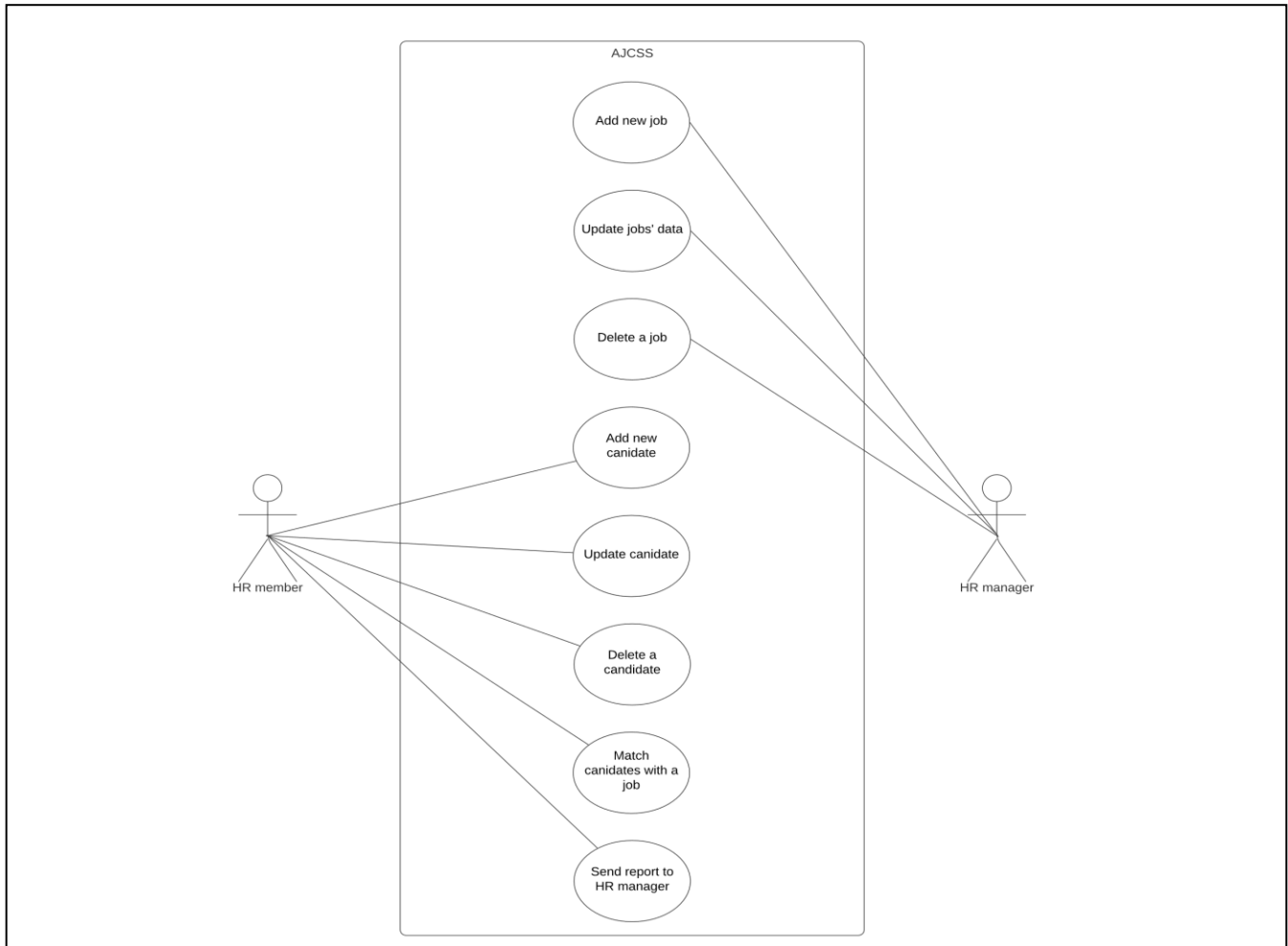

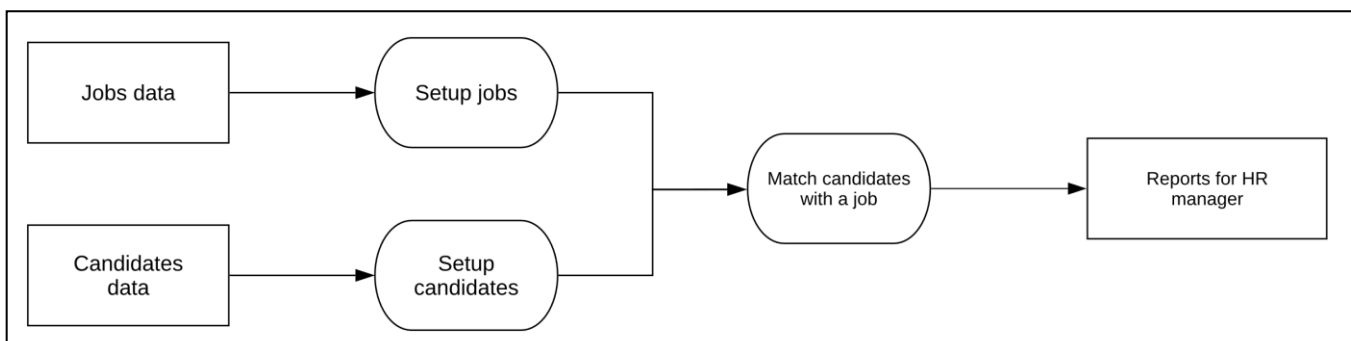**Fig. 1** Use case diagram that depicts the use cases of HR manger and member actors.

Fig. 2 pipe and filter pattern that depicts the data flow to the process of finding a match.

2.2 Class Diagram
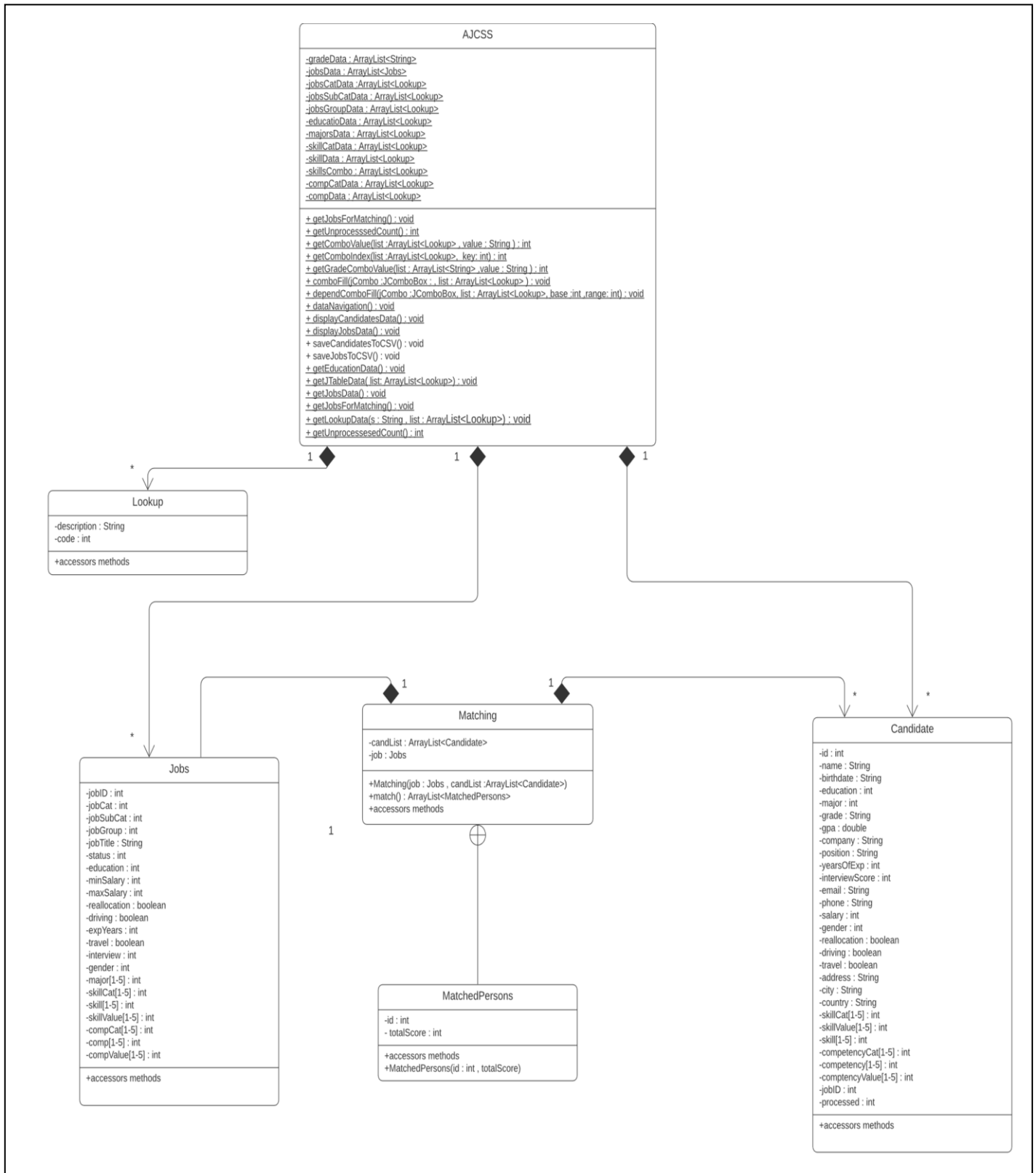
The class diagram shows the structure and relationships of AJCSS software.

AJCSS

-gradeData : ArrayList<String>
-jobsData : ArrayList<Jobs>
-jobsCatData :ArrayList<Lookup>
-jobsSubCatData : ArrayList<Lookup>
-jobsGroupData : ArrayList<Lookup>
-educatioData : ArrayList<Lookup>
-majorsData : ArrayList<Lookup>
-skillCatData : ArrayList<Lookup>
-skillData : ArrayList<Lookup>
-skillsCombo : ArrayList<Lookup>
-compCatData : ArrayList<Lookup>
-compData : ArrayList<Lookup>

+ getJobsForMatching() : void
+ getUnprocesssedCount() : int
+ getComboValue(list :ArrayList<Lookup> , value : String) : int
+ getComboIndex(list :ArrayList<Lookup>, key: int) : int
+ getGradeComboValue(list : ArrayList<String> ,value : String) : int
+ comboFill(jCombo :JComboBox : , list : ArrayList<Lookup>) : void
+ dependComboFill(jCombo :JComboBox, list : ArrayList<Lookup>, base :int ,range: int) : void
+ dataNavigation() : void
+ displayCandidatesData() : void
+ displayJobsData() : void
+ saveCandidatesToCSV() : void
+ saveJobsToCSV() : void
+ getEducationData() : void
+ getJTableData(list: ArrayList<Lookup>) : void
+ getJobsData() : void
+ getJobsForMatching() : void
+ getLookupData(s : String , list : ArrayList<Lookup>) : void
+ getUnprocessesedCount() : int

Lookup

-description : String
-code : int

+accessors methods

Matching

-candList : ArrayList<Candidate>
-job : Jobs

+Matching(job : Jobs , candList :ArrayList<Candidate>)
+match() : ArrayList<MatchedPersons>
+accessors methods

Jobs

-jobID : int
-jobCat : int
-jobSubCat : int
-jobGroup : int
-jobTitle : String
-status : int
-education : int
-minSalary : int
-maxSalary : int
-reallocation : boolean
-driving : boolean
-expYears : int
-travel : boolean
-interview : int
-gender : int
-major[1-5] : int
-skillCat[1-5] : int
-skill[1-5] : int
-skillValue[1-5] : int
-compCat[1-5] : int
-comp[1-5] : int
-compValue[1-5] : int

+accessors methods

MatchedPersons

-id : int
- totalScore : int

+accessors methods
+MatchedPersons(id : int , totalScore)

Candidate

-id : int
-name : String
-birthdate : String
-education : int
-major : int
-grade : String
-gpa : double
-company : String
-position : String
-yearsOfExp : int
-interviewScore : int
-email : String
-phone : String
-salary : int
-gender : int
-reallocation : boolean
-driving : boolean
-travel : boolean
-address : String
-city : String
-country : String
-skillCat[1-5] : int
-skillValue[1-5] : int
-skill[1-5] : int
-competencyCat[1-5] : int
-competency[1-5] : int
-comptencyValue[1-5] : int
-jobID : int
-processed : int

+accessors methods

Fig. 3 AJCSS class diagram

2.3 Class Method Descriptions

AJCSS

The AJCSS class is main class which has the graphical user interface (GUI) components, the methods that initializes all the data and fills the GUI components when the program starts. Also, it has methods which save the progress into CSV files.

Class	AJCSS
Method	getUnprocesssedCount
Visibility	Public
Return type	Int
Parameter, types	None
Description	This method counts how many unprocessed candidates are there.

Class	AJCSS
Method	getComboValue
Visibility	Public
Return type	Int
Parameter, types	list : ArrayList<Lookup> , value : String
Description	This method gets the value of the selected item in a combo box.

Class	AJCSS
Method	getComboIndex
Visibility	Public
Return type	Int
Parameter, types	list : ArrayList<Lookup> , key : int
Description	This Method gets the index of the selected item in a combo box.

Class	AJCSS
Method	comboFill
Visibility	Public
Return type	void
Parameter, types	jCombo : JComboBox , list : ArrayList<Lookup>
Description	This method fills a combo box from an arraylist.

Class	AJCSS
Method	dataInitialization
Visibility	Public
Return type	void
Parameter, types	none
Description	This method initializes all the data at the start of the program from CSV files.

Class	AJCSS
Method	getJobsData
Visibility	Public
Return type	void
Parameter, types	none
Description	This method fetches the jobs data from CSV file. Then stores the fetched data into an arraylist in program.

Class	AJCSS
Method	getLookupData
Visibility	Public
Return type	void
Parameter, types	s : String , list : ArrayList<Lookup>
Description	This method fetches the lookup data from CSV file. Then stores the fetched data into an arraylist in program.

Class	AJCSS
Method	SaveCandidatesToCSV
Visibility	Public
Return type	void
Parameter, types	none
Description	This method saves the candidate's data into a CSV file.

Class	AJCSS
Method	SaveJobsToCSV
Visibility	Public
Return type	void
Parameter, types	none
Description	This method saves the job's data into a CSV file.

Matching

This Matching class is designed to process the matching of the selection criteria by using a method which finds a match for unprocessed candidates with a vacant job.

Class	Matching
Method	match
Visibility	Public
Return type	ArrayList<MatchedPersons>
Parameter, types	none
Description	This method looks for a match between selected unprocessed candidates and a vacant job.

Jobs

The Jobs class is designed to store jobs' data that contains essential data for processing. As well update a job's data. This class contains accessors methods.

Lookup

The Lookup class is designed to reference jobs' data to coded values. A job's data is written with a description; thus, it will reference the job's data description to a coded integer value to ease the operation and process. This class contains accessors methods.

Candidate

The Candidate class is designed to store candidates' data that contains essential data for processing. As well update a candidate's data. This class contains accessors methods.

3. Detailed Design

This section contains diagrams depicting low-level details of pieces of the AJCSS software.

Figure 4 depicts an activity chart at the start of the program. Activity includes loading the files into the program, storing data in program data structures, selecting user type, sending reports and storing data into files.
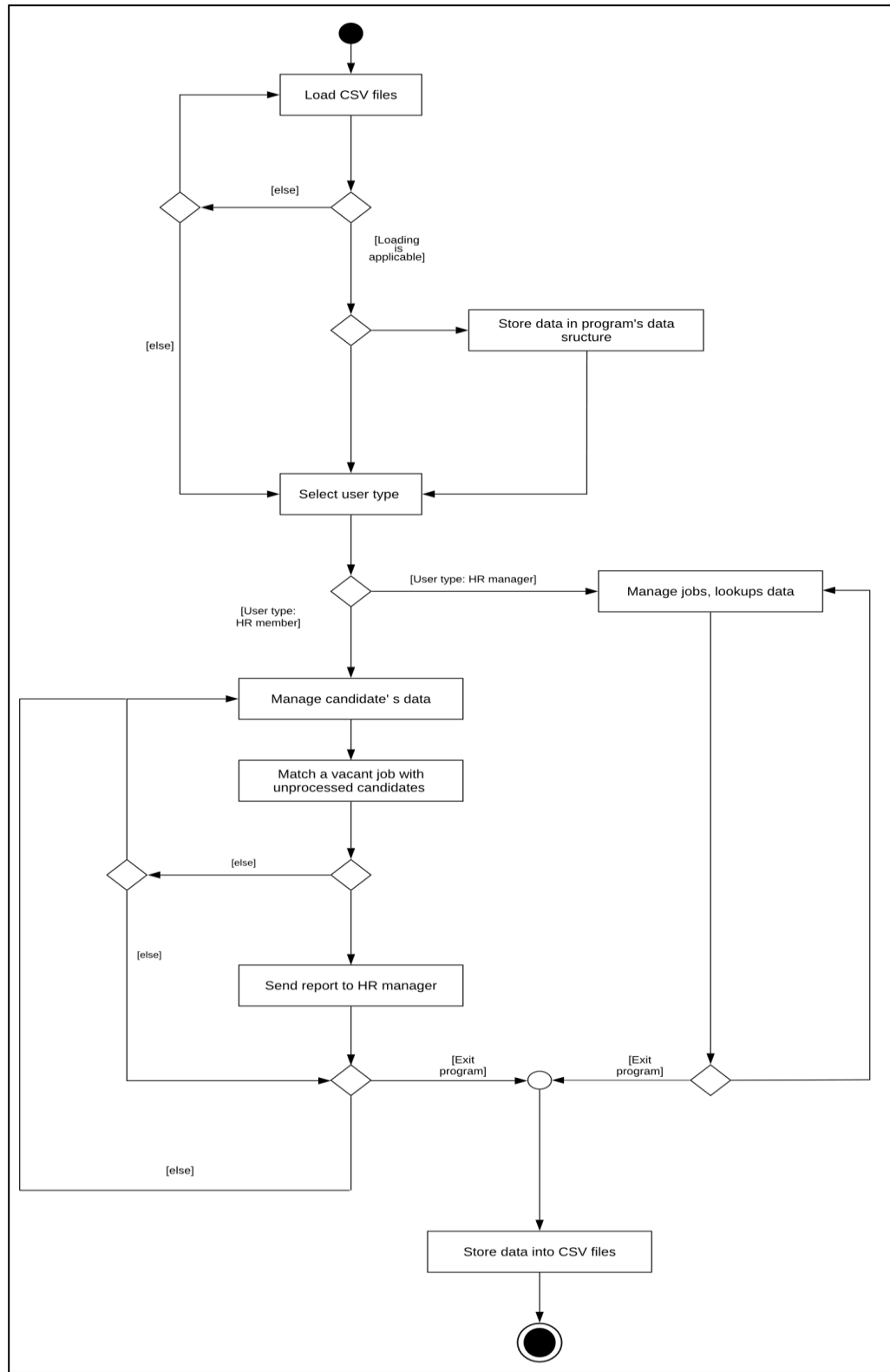
Figure 5 is a flowchart showing the method Mathching:match() which finds a match for unprocessed candidates with a vacant job.
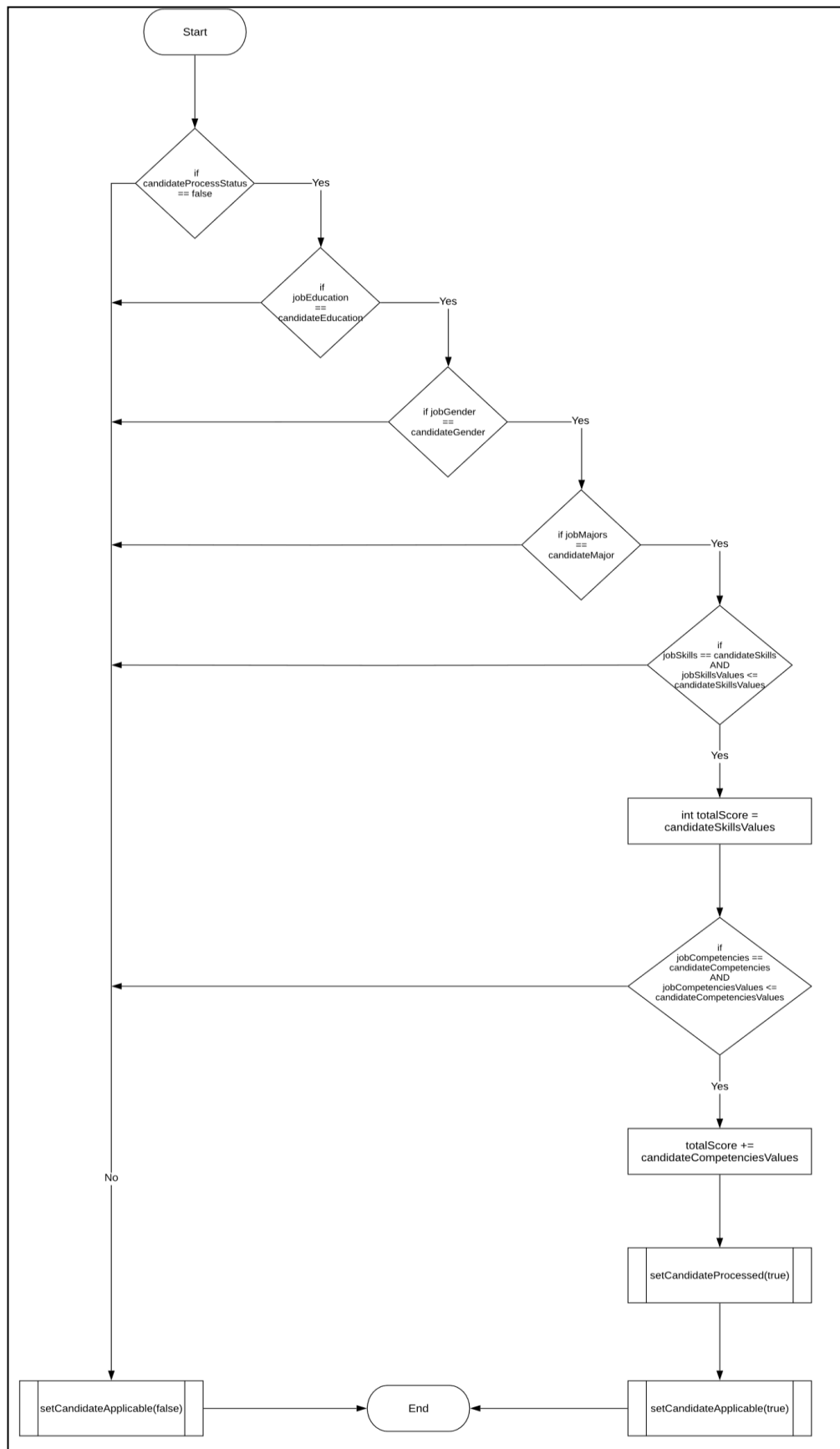
Fig. 5 The match() method from the Matching class.

4. User Interface Design

This section contains screen shots of initial versions for the AJCSS user interface.
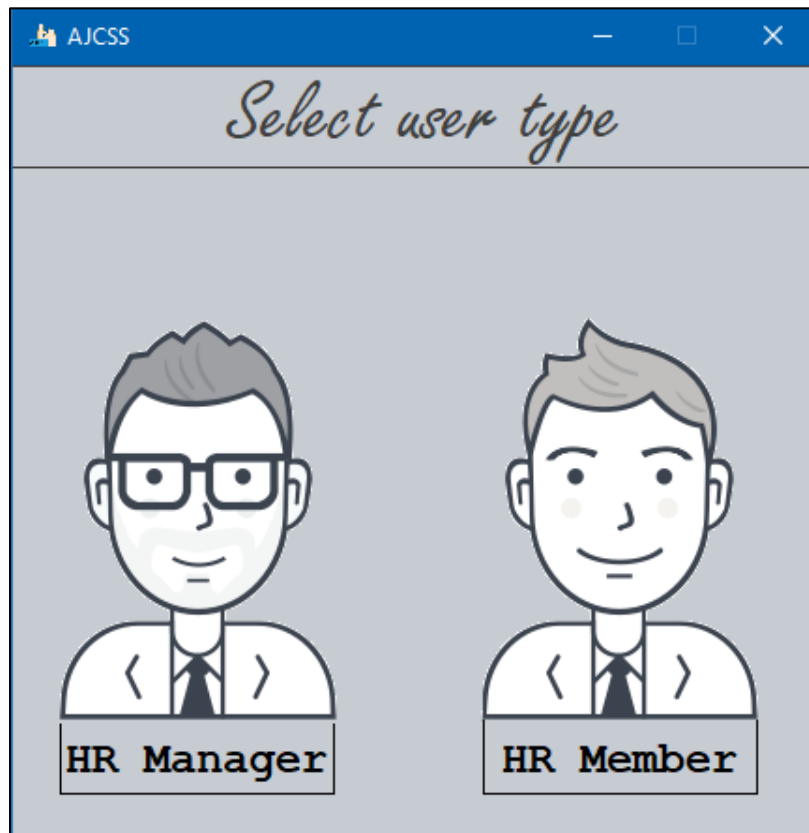
Fig. 6 This window lets you select the user type.
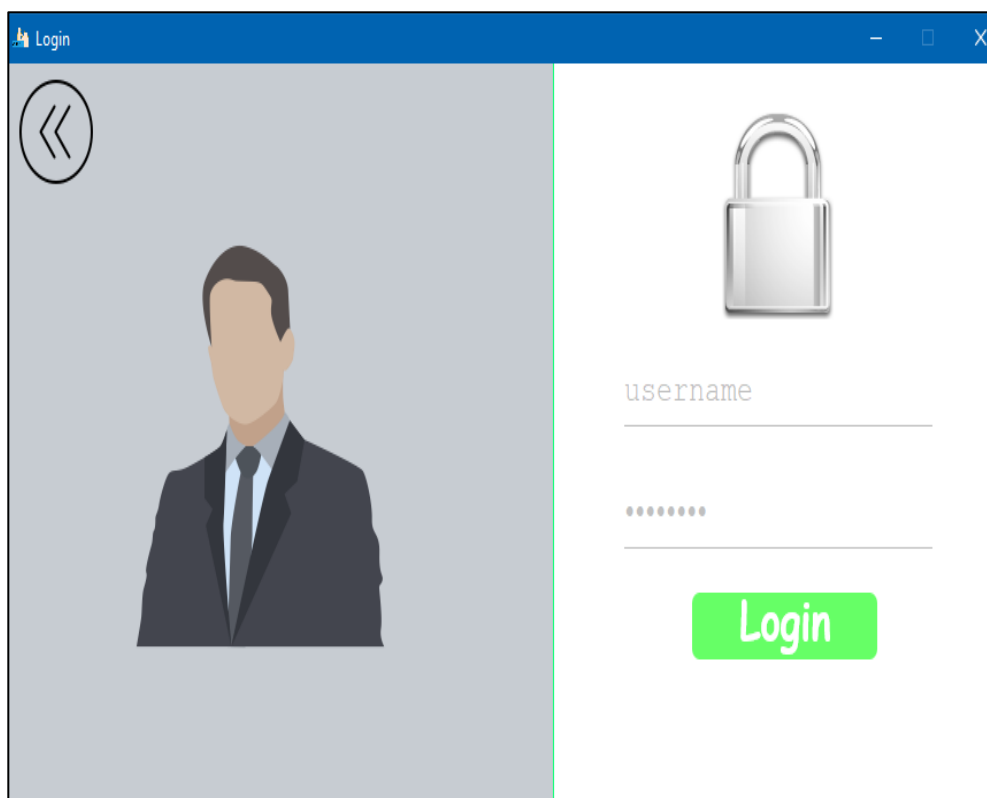
Fig. 7 This window is for HR manager user type which asks for admin credentials.

Fig. 8 A screen shot of the jobs-new job section which let you setup a new job.

Fig. 9 A screen shot of the jobs-vacant jobs section which shows you all vacant jobs.

Fig. 10 A screen shot of the candidates- new candidate section which lets you setup the a new candidate.

Fig. 11 A screen shot of the candidates- unprocessed candidate section which shows you all unprocessed candidates.

Fig. 12 A screen shot of the lookups section which shows you all lookups that were loaded from CSV file.

Fig. 13 A screen shot of the matching section which lets you select unprocessed candidates to match them with a vacant job.

5. Conclusion

In the design document we gave a brief visualization about the AJCSS software, we have clarified the process using high level and medium level design that involve the system level diagram and class diagram. Also, we have specified back-end infrastructure for the AJCSS software, using the detailed design that involves activity chart and flowchart. Finally, we have shown the expected AJCSS graphical user interface in the user interface design section.