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

Second term 1441/2020

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

Project-Phase No: 01
Automatic Jobs Candidates Selection System (AJCSS)
(Specification)

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: 29/2/2020

Table of Contents

1. Introduction

Here we will give a scope description and overview of everything included in this document. Also, the purpose for this document is described and a list of abbreviations and definitions is provided.

1.1 Purpose

The purpose of this document is to give you a brief explanation of the requirements for the automatic jobs candidates selection system (AJCSS) software. It is a process of recruiting Applicants and job seekers; it will save time and cost for hiring department.it will show HR employees the best options for the selections it also will display the design and functions of the system, it will be easy for HR employees to determine the best option for the job and fast way to communicate with the candidate.

1.2 Scope

(AJCSS) is a software for companies to find best option in the job that they want it to search by the choices of the HR employee to get the best option and there is different percentage that gives you how much is the candidate is suitable for the job and it has other specifications like the candidate salary and communication info such as email or phone number and read pre-sorted data and set interview type. This software will be installed in HR employees' computers.

1.3 Definitions, Acronyms, and Abbreviations

The following is a list of terms, acronyms and abbreviations used by the "Automatic Jobs Candidates Selection System" software package and related documentation.

SRS	Software Requirements Specification
UESR	Someone who interacts with The Website
HR	Human Resources
CSV	Comma Separated values
AJCSS	Automatic Jobs Candidates Selection System
LOOKUP	CSV file contains list values (code, description)

1.4 Overview

The next chapter (The General Description section) of this document gives an overview of the functionality of the product and present all main functions that the system should do. This chapter is intended mainly for stakeholders so that does not contain much technical terms. The third chapter (Requirements specification section) of this document is written primarily for the developers. It describes in technical terms the details of the functionality of the system. NOTE: both sections of the document describes the same software product in its entirety, but they are intended for different audience and thus use different language.

2. General Description

Here we will give an overview of the whole AJCSS software, how it will be explained in its context, how it interacts with other systems and introduce the basic functionality of it. It will also describe what type of users that will use the system and what functionality is available for each type of users. At last, the constraints for the system will be presented.

2.1 Product Perspective

The AJCSS is a desktop-based system. The system interfaces have two parts, the HR Manager interface and the HR member interface. The system provides a secure environment for storing and reclaiming of confidential Managers, Members and Jobs candidate's information.

Our product depends on data, it needs somewhere to store the data. That's why CSV files will be used. The desktop application will communicate with the CSV files (see Figure 1), it will add and modify data in an offline environment.

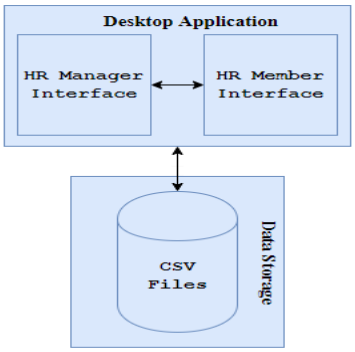

Figure 1 - Block diagram

There are some restrictions in the application regarding the use, the HR manager set up lookups and jobs information, while the HR member can only fill in the information of job candidates then submit them to the HR manager.

2.2 Product Functions

The AJCSS allows the HR manager to manage lookups' data either by inserting new values (code and description for attribute list values) or changing the description of the available lookup. Also, to add new jobs' data by filling its attributes - some attributes have a lookup as pre-stored list values- according to job's properties. In order to facilitate the usage of the system, HR manager can also update and delete jobs' data from the system. HR member can add new candidates' data according to each candidate profiles, education details, skills, competencies, and experiences. HR member can also update and delete candidates' data. The most important part of AJCSS is the matching part where HR member selects a vacant job and apply the matching part of the system to produce a list of all matched candidates of the selected job's properties. If a match found, then a report will be generated from the system including all matched candidates sorted by the highest matching criteria percentage. Then HR member can send the report to the HR manager.

2.3 User Characteristics

The two main groups of AJCSS are HR members and HR managers. A HR manager will have to setup the jobs that they need, HR manger will also have to setup the interview's questions for the jobs and put the threshold for accepting candidates. HR members would do the interviews with the candidates and at the end of the day they will send a report to the HR manger about the candidates.

2.4 General Constraints

This system will be accessed only by HR managers and HR members, the interface will be very simple that it will not need any training to use and it will save time and money for HR department. All the information about the candidates will be store in a CSV files and will be sent at the end of the day to the HR manager to review it.

3. Specific Requirements

Here we list all the functional and non-functional requirements of the AJCSS software. The function will be described in detailed description of the AJCSS software and all its features.

3.1 Functional Requirements

3.1.1 <Functional Requirement or Feature #1>

3.1.1.1 **Function**: upload stored data from CSV files.

3.1.1.2 **Description**:

Upload data from CSV files where data already stored for lookups, jobs, and candidates.

3.1.1.3 **Input**:

CSV files.

3.1.1.4 **source**:

CSV files stored in the same directory where the main program stored.

3.1.1.5 **Outputs**:

Insert the successfully loaded data into data structure in the JAVA program.

3.1.1.6 **Destination**:

ARRAYS, linked lists in JAVA class.

3.1.1.7 **Action**:

Program must check data loading process and show a successful message if all files were loaded successfully, otherwise an error message must be displayed using exception handling. In case of failure program give the choice for the user to use the system without having data loaded where user must add data for lookups, jobs, and candidates.

3.1.1.8 **Requirements**:

Data must be stored in CSV files properly according to the data structure formats.

3.1.1.9 **Pre-condition:**

CSV files must available in the same directory where the main program stored.

3.1.1.10 **Post-condition:**

Loaded data to be stored in data structure in the JAVA class. In case the user

3.1.1.11 **Side effect**:

Data structures limit.

3.1.2 <Functional Requirement or Feature #2>

3.1.2.1 **Function**: Update the lookups data

3.1.2.2 **Description**:

Updates the lookups data according to any new additions. Update includes inserting new values (codes and descriptions) and changing the description of the available contents. Deletion is not included as it is not allowed due to need to integrity.

3.1.2.3 **Input**:

In the case of insertion:

Code: which is an **integer;**

Description: text (string of length 50 characters).

In case of change:

Description: text (string of length 50 characters).

3.1.2.4 **source**:

New recruitments and educational data.

3.1.2.5 **Outputs**:

Store the updated data into data structures in the JAVA program.

3.1.2.6 **Destination**:

Lookup data structure in the JAVA program.

3.1.2.7 **Action**:

User can traverse all available data in a certain lookup and select either to insert or change data. In the insert case, user must have 2 items namely: new code and new description. The new code must:

- follow the sequence and format of the code in the selected lookup,
- not be exist in the lookup before to preserve the uniqueness feature,

The new description must:

- follow the format and size of the selected lookup,
- not to exist in the lookup before to prevent duplication.

In case of change, **code** must not be changed to prevent any integrity loss as **codes** could be already used in jobs and candidate's data. Description could be changed according to the format and size of the description field in the selected lookup.

New data to be inserted, or description to be updated in a certain lookup's Data structure in the JAVA class.

3.1.2.8 **Requirements**:

New code, new description, updated description to be unique and to follow the sequence, format, and size of the lookup's data.

3.1.2.9 **Pre-condition**:

The capacity of the data structure allows the insertion/ replacement.

3.1.2.10 **Post-condition**:

Data structure to be maintained according to the insertion/replacement.

3.1.2.11 **Side effects**:

Data structures limit.

3.1.3 <Functional Requirement or Feature #3>

3.1.3.1 **Function**: Add new job

3.1.3.2 **Description**:

Add a new job's data to the system. Job's data are the attributes which describes the details of a job.

3.1.3.3 **Input**:

All compulsory attributes to be inserted according to the data's type, format, and size of each attribute. Optional attributes may be left blank as it has null values by default.

3.1.3.4 **source**:

Details of a new vacant job.

3.1.3.5 **Outputs**:

New job's data which will be inserted in the data structure in the JAVA program.

3.1.3.6 **Destination**:

Job's data structure in the JAVA program,

3.1.3.7 **Action**:

Job's attributes to be entered in the specified fields according to the data's type, format, and size of each attribute. Some attributes are compulsory were others are optional and will accept null values. Some attributes need to be inserted as text, digits, dates, and Boolean values, where others may be selected from pre-stored lookups. The default status of the inserted job will be vacant.

Job number will be generated by the system and will not be inserted by user. It is a functional requirement by itself.

3.1.3.8 **Requirements**:

Job details not to be inserted before in the system.

3.1.3.9 **Pre-condition**:

The capacity of the data structure allows the insertion.

3.1.3.10 **Post-condition**:

Data structure to be maintained according to the insertion.

3.1.3.12 **Side effects**:

Data structures limit.

3.1.4 <Functional Requirement or Feature #4>

3.1.4.1 **Function**: Update job's data.

3.1.4.2 **Description**:

Update an existed job's data. Job must be vacant to be updated.

3.1.4.3 **Input**:

Attributes to be updated by new data.

3.1.4.4 **source**:

An existed vacant job in the system.

3.1.4.5 **Outputs**:

Store the updated job's attributes into data structures in the JAVA program.

3.1.4.6 **Destination**:

Jobs' data structure in the JAVA program.

3.1.4.7 **Action**:

User can update job's attributes according to the data's type, format, and size of each attribute.
Optional attributes may be left blank as it has null values by default.
Job number will remain the same and will not be updated.
Status of the updated job will remain vacant and will not be updated.

3.1.4.8 **Requirements**:

At least 1 vacant Job exists in the system.

3.1.4.9 **Pre-condition**:

- Compulsory attributes must have data (not to be left blank).
- Entered data must match attributes' type, format, and size.

3.1.4.10 **Post-condition**:

None.

3.1.4.12 **Side effects**:

None.

3.1.5 <Functional Requirement or Feature #5>

3.1.5.1 **Function**: Delete a job.

3.1.5.2 **Description**:

Delete an existed job's data from the system.

3.1.5.3 **Input**:

A vacant job to be selected for deletion using job number.

3.1.5.4 **source**:

An existed vacant job in jobs data structure in the JAVA program.

3.1.5.5 **Outputs**:

Jobs' data structure to be changed accordingly.

3.1.5.6 **Destination**:

Jobs data structure in the JAVA program.

3.1.5.7 **Action**:

The selected job's data to be deleted from jobs' data structure in the JAVA program.

3.1.5.8 **Requirements**:

At least 1 vacant Job existed in the system.

3.1.5.9 **Pre-condition**:

Jobs' data structure must have vacant jobs.
Job must be vacant to be deleted.

3.1.5.10 **Post-condition**:

Data structure to be maintained accordingly.

3.1.5.12 **Side effects**:

None.

3.1.6 <Functional Requirement or Feature #6>

3.1.6.1 **Function**: Add new candidate

3.1.6.2 **Description**:

Add a new candidate data to the system. Candidate's data are the attributes which describe the details of the candidate.

3.1.6.3 **Input**:

All compulsory attributes to be inserted according to the data's type, format, and size of each attribute. Optional attributes may be left blank as it has null values by default.

3.1.6.4 **source**:

New candidate data.

3.1.6.5 **Outputs**:

New candidate's data which will be inserted in the data structure in the JAVA program.

3.1.6.6 **Destination**:

Candidate's data structure in the JAVA program.

3.1.6.7 **Action**:

Attributes to be entered in the specified fields according to the data's type, format, and size of each attribute. Some attributes are compulsory where others are optional and will accept null values. Some attributes need to be inserted as text, digits, dates, and Boolean values where others may be selected from pre-stored lookups. The default status of the inserted candidate will be **unprocessed**.

Candidate number will be generated by the system and will not be inserted by user. It is a functional requirement by itself.

3.1.6.8 **Requirements**:

candidate details not to be existed in the system.

3.1.6.9 **Pre-condition**:

The capacity of the data structure must allow the insertion.

3.1.6.10 **Post-condition**:

Data structure to be maintained according to the insertion.

3.1.6.12 **Side effects**:

Data structure limit.

3.1.7 <Functional Requirement or Feature #7>

3.1.7.1 **Function**: Update candidate's data.

3.1.7.2 **Description**:

Update an existed candidate's data.

3.1.7.3 **Input**:

Candidates' attributes to be updated by new data.

3.1.7.4 **source**:

An existed candidate in the system.

3.1.7.5 **Outputs**:

Updated attributes to be stored into data structure in the JAVA program.

3.1.7.6 **Destination**:

candidate's data structure in the JAVA class,

3.1.7.7 **Action**:

Attributes must be updated according to the data's type, format, and size of each attribute.
Optional attributes may be left blank as it has null values by default.
Candidate number will remain the same and will not be updated.
If Status of the updated candidate is **processed** it will be set to **unprocessed**.

3.1.7.8 **Requirements**:

At least 1 candidate exists in the system.

3.1.7.9 **Pre-condition**:

- Compulsory attributes must have data (not to be left blank).
- Entered data must match attributes' type, format, and size.

3.1.7.10 **Post-condition**:

Candidate's statues will be set to **unprocessed**.

3.1.7.12 **Side effects**:

None.

3.1.8 <Functional Requirement or Feature #8>

3.1.8.1 **Function**: Delete a candidate.

3.1.8.2 **Description**:

Delete an existed candidate's data.

3.1.8.3 **Input**:

User can select a candidate to be deleted using candidate's ID.

3.1.8.4 **source**:

An existed candidate in candidate's data structure in the JAVA program.

3.1.8.5 **Outputs**:

Candidate's data structure to be modified accordingly.

3.1.8.6 **Destination**:

Candidates data structure in the JAVA class,

3.1.8.7 **Action**:

The deleted candidate's data to be deleted from candidate's data structure in the JAVA program.

3.1.8.8 **Requirements**:

At least 1 candidate existed in the system.

3.1.8.9 **Pre-condition**:

Candidate's data structure must have candidates' data.

3.1.8.10 **Post-condition**:

Data structure to be maintained accordingly.

3.1.8.12 **Side effects**:

None.

3.1.9 <Functional Requirement or Feature #9>

3.1.9.1 **Function**: match candidates with a job.

3.1.9.2 **Description**:

The user selects a vacant job and match it with the unprocessed candidates to find the suitable ones for that job.

3.1.9.3 **Input**:

User can traverse all vacant jobs and select a job for matching.

3.1.9.4 **source**:

An exist job in job's data structure in the JAVA class.

3.1.9.5 **Outputs**:

System will display a list of all candidates who meet the comparison criteria.

3.1.9.6 **Destination**:

A report of all matched candidates to be sent to HR manager.

3.1.9.7 **Action**:

Matching process will compare certain attributes in job's data with some identical attributes in candidate's data. All candidates with attributes value greater than or equal to the job's compared attributes will be assigned as a match. Any matched candidate will be assigned an overall percentage based on the comparison result of his attributes and will be prioritized based on that percentage.

The comparison criteria will be as follows:

Candidate's attribute	Match criteria
Educational level	Exact match or higher level must be found
Specialization	Candidates specialization must Match at least 1 of job's specializations
Interview grade	Must be greater than or equal to job's interview required value.
Skills	Exact match for skill category must be found Candidates' matched skills values must be greater than or equal to job's skills required value.
competencies	Exact match for competency category must be found Candidates' matched competencies values must be greater than or equal to job's competencies required value.

If match found, then the status of matched candidates will be changed to **processed** and a report of them will be stored in a data structure in the JAVA program in order to be sent to HR manager by mail, otherwise, system will display a message showing no match found.

3.1.9.8 **Requirements**:

At least 1 unprocessed candidate exists in the system.
At least 1 vacant job exists in the system.

3.1.9.9 **Pre-condition**:

None.

3.1.9.10 **Post-condition**:

A report of all matched candidates picturized by the most suitable ones.

3.1.9.11 **Side effects**:

Capacity of the system.

3.1.10 <Functional Requirement or Feature #10>

3.1.10.1 **Function**: generate job number

3.1.10.2 **Description**:

System generates a unique number for the new inserted job.

3.1.10.3 **Input**:

New inserted job's data.

3.1.10.4 **source**:

Details of a new vacant job.

3.1.10.5 **Outputs**:

New job number will be inserted in the data structure in the JAVA program.

3.1.10.6 **Destination**:

Job's data structure in the JAVA program,

3.1.10.7 **Action**:

Once job's data is entered and confirmed, and pre insertion of job's data to data structure, system searches jobs data in data structure to find the latest inserted job number. System adds 1 to that number then insert it into job number attribute. For the first job, job number attribute will be set to 1000. Then all attributes will be stored in the data structure.

3.1.10.8 **Requirements**:

Job's data entered properly and confirmed valid.

3.1.10.9 **Pre-condition**:

Insert event handler is activated.

The generated number must not exceed the job number attribute's range (integer).

3.1.10.10 **Post-condition**:

The generated number will be placed in job number attribute.

3.1.10.12 **Side effects**:

Job number attribute range could be exceeded.

3.1.11 <Functional Requirement or Feature #11>

3.1.11.1 **Function**: send a report of matched candidates to HR manager

3.1.11.2 **Description**:

System will send a report of all matched candidates to a certain vacant job.

3.1.11.3 **Input**:

Matched candidates list.

3.1.11.4 **source**:

The stored matched candidates in the data structure.

3.1.11.5 **Outputs**:

A report of the matched candidates to a certain job.

3.1.11.6 **Destination**:

HR manager email.

3.2.11.7 **Action**:

System will check the matching data structure for a certain job, if match found, HR member will activate the send event's handler where the matched candidate data in data structure will be collected and sent to HR manager by email. The email of HR manager will be entered manually before send. If no match found, system will display a notification message accordingly.

3.1.11.8 **Requirements**:

Matched candidates to be already stored in data structure.

3.1.11.9 **Pre-condition**:

Send event handler is activated.

Match found.

Correct HR manager entered.

Network/internet availability.

3.1.11.10 **Post-condition**:

None.

3.2.11.12 **Side effects**:

None.

3.2 Non-Functional Requirements

3.2.1 <Non-Functional Requirement or Feature #1>

operational requirement:

- The system must be easy and simple to use for the staff.
- The system will operate in Windows environment.

3.2.2 <Non-Functional Requirement or Feature #2>

Performance (response time) requirement:

- The response time must be fast to serve largest numbers of operations.

3.2.3 <Non-Functional Requirement or Feature #3>

integrity and flexibility requirement:

- Easy in dealing with the system.
- The system should be flexible to help the users.

3.2.4 <Non-Functional Requirement or Feature #4>

Design requirement:

- The system is designed as a JAVA program connected with CSV files in windows.
- Design, color and font size must be appropriate, user friendly.

3.2.5 <Non-Functional Requirement or Feature #5>

storage requirement:

- The system should have 10GBs (minimum) storage space.

3.2.6 <Non-Functional Requirement or Feature #6>

Quality:

- The system must implement a proper data structure based on the system requirements.

4. Conclusion

In SRS document we gave a brief explanation about the AJCSS software, we describe the purpose and we locate the desired scope, we explain how the user characteristics should be and we define the expected general constraints that we may be encounter thought the AJCSS software, we clarify the product perspective and function. Also, the functional and non-functional requirements are specified in full detailed description.