SOFTWARE REQUIREMENT SPECIFICATION (SRS)

for

Project #1 - Ballot Counting System (BCS)

Prepared by:
Jason C. Phadnis
Kobin J. Khadka
Brandon C. Schenck
Gavin Huang

Team 5

2/1/2021

1. Introduction

1.1 Purpose

This document describes the Ballot Counting System, henceforth referred to as BCS. More specifically we cover the program that uses input ballot data to determine the results of an instant runoff or open party election. The software then displays the winners and other information such as the sats for each candidate. It will also generate an audit file depicting the process leading to that result. We cover the specific requirements to run the program, how it handles ballot data, and what results it generates in greater detail later in this document. This document is intended to be used by the election officials using the software and potential developers.

1.2 Document Conventions

Major sections are indicated both with a number and larger font sizes. Subsections are written with smaller font sizes than the section header. Furthermore, these subsections have the number of the main section and a number for that particular subsection. For section 4, the features are listed in chronological order when running the completed program. There is a link to the Use Case Documentation in section 1.5 that will take you to an external document containing the use cases. The document will also be available via github.

1.3 Intended Audience and Suggested Reading

The intended audience for this document is for developers, testers, and election officials who have an interest in understanding how this software works. This document can be read in any order. It is divided by multiple sections and there is a table of contents to guide the reader. Testers and election officials would in particular be interested in sections 4.6 to 4.8 as those relate to the results. Election officials may also be interested in section 4.5 to see how ties are handled.

1.4 Product Scope

The BCS is the software that is used in the voting process. In this software, it is capable of running two different types of elections (Instant runoff and Open Party Listing). The user will be able to input the ballots through a csv file. The system will sort and count the votes. When all the votes are counted, the system will determine the winner based on the algorithm it contains. When the software is finished running, it will display the results on the screen and create an audit file. This allows the user to get the results of an election quickly and accurately.

1.5 References

Use Case Documentation

2. Overall Description

2.1 Product perspective

This voting system is going to be used for the purpose of counting the votes, and providing a history of Open Party Listing and Instant Runoff voting. This system will be integrated into a larger online voting system. This system will read in a file that details the type of voting that was used for the election, who was running and for which party, and it will produce an audit file that describes the history of the election and how votes were distributed. The system will determine which voting algorithm to run based on the first line in the CSV file. This system will run on the most up-to-date CSElabs machines.

2.2 Product function

1. Input: User inputs name of the file that has the voting details.

2. Open : Opens file that is in supported file format else shows error message.

3. Calculate election result : If the file is successfully loaded, the system generates the result of the election. If the file is not loaded successfully the user is then reprompt for a valid file name.

4. Save : Generates two files one to show how the results were calculated by the system which can be used to audit later and another file to share the result to the media.

5. Print : Prints the election results. This is only available if the result is generated successfully.

6. Exit: Turn off the system.

2.3 User Classes and Characteristics

- Programmers and testers who build, test and maintain the system.

- Election officials who use the system to determine the election results and collect other related information.

- Media persons who receive election results from the system in a timely manner.

2.4 Operating Environment

The software works with most up to date CSELabs machines with following environment:

- Windows 10 on
  - Dell Precision T7910 with
    - Xeon @ 3GHz (x12)
    - 32 GB RAM
  - Dell Precision T3420
    - Intel® Core™ i5 @ 3.4GHz (x4)
    - 64 GB RAM
- Ubuntu 16.04 on
  - Dell Precision 3630 with
    - Intel Core i7-4790 (Quad-Core) 3.60GHz
    - 32 GB RAM, a DVD+/-RW
    - Intel® UHD Graphics 630
  - Dell Optiplex 9020
    - Intel® Core™ i7 @ 900Hz (x3)
    - 32 GB RAM
  - Dell OptiPlex 7050
    - Intel Core i7-7700 3.6GHz
    - 32 GB RAM
    - DVD+/-RW
    - Intel® UHD Graphics 630
  - Dell Power Edge R720
    - 2 x 8-Core Intel Xeon 2.00GHz
    - 64 GB RAM
    - Xeon Phi cards for MIC Architecture programming
  - Dell Precision T5400
    - 2 x Quad-Core Intel Xeon 2.00GHz
    - 8 GB RAM
    - NVIDIA GeForce graphics for CUDA programming

- Mac OS X
  - iMacs
    - Intel Core i5 3.2 GHz
    - 16 GB RAM
    - DVD-RW
    - 2 x NVIDIA® GeForce GT 755M 1024MB.

The software will work with a comma separated values (CSV) file exported from Microsoft Excel where the first line is the type of election: instant runoff or open party. Furthermore, each row will be a new linand e, this file is expected to be in the same directory as the software.

2.5 Design and Implementation Constraints

The Ballot counting systems are developed in Java. The system uses modular design where every feature is wrapped into a separate module and the modules depend on each other. It needs to be in English. The program needs to be able to process 100 thousand ballots within 8 minutes.

2.6 User Documentation

No further user documentation will be provided.

2.7 Assumptions and Dependencies

The ballot counting system is developed in Java. This means the user must have Java on their system. The up to date version of Ballot Counting System requires Java version 7 or higher. The input file with the ballot information must be in the same directory that this application is running from.

3. External Interface Requirements

3.1 User Interfaces

The user interface will use the command line. It will start with a welcome message stating “Welcome to the Ballot Counting System” the interface will then prompt the user to input a file name. The system will then check to see if the file name is valid. If the file name is not valid, the system will display an error message to the user interface saying “The file name you

inputted is not a valid file name. Please enter a valid file name or press ctrl+C to quit". After receiving a valid file name, the system will print to the screen the type of voting algorithm that is being run. The message will say "Your file was processed using Open Party Listing" or it will say "Your file was processed using Instant Runoff". When the processing is complete, the system will display the results to the screen. The results include the winner(s) of the election, number of seats available, the number of votes cast, and number of votes each candidate received (the format for number of votes each candidate received will be in "[Candidate Name] - [number of votes received]"). If there is a tie and a coin flip needs to take place, the user interface will display on the screen "There is a tie between [Candidate Name] and [Candidate Name]. A coin will be flipped 999 times to determine the winner.", a coin will be flipped 999 times by the system and will display the result to the screen by saying "[Candidate Name] wins the coin toss!". When the system is finished displaying the results to the screen, the system will then print "an audit file and press release file have been created in the same folder as the system" to tell the user there is an audit file they can view with full information about how the election proceeded and there is a press release file with results and public statistics that can be shared with the media. Then the system will re-prompt the user for another file and repeat this process.

3.2 Hardware Interfaces

The hardware requirements for our system will be the most updated CSElabs machine. For more information visit https://cse.umn.edu/cseit/classrooms-labs.

3.3 Software Interfaces

This software requires Java to be installed on the system, more specifically version 7 and above. Additional information can be found in section 2.7 and 2.5 of this document.

3.4 Communications Interfaces

This software will require internet connection to install new plugins, update already installed ones, update important information for the election, update some key components such as modules, and update its security. The program will not use any communication protocol. There will be no communication security or encryption issues. There are no special synchronization mechanisms.

4. System Features

This section demonstrates the BCS features and explains how they can be used and the results they will give back to the user.

4.1 System Feature 1 - User Interface/Starting the program

4.1.1 Description and Priority

The User Interface for the system will be the command line interface. Upon starting the program, the user will be prompted to enter the name of a CSV file that contains the voting information. The interface will then display the results of the election as well as notify the user that files have been generated for their use. This is a feature that is not necessarily a core component for the program to run so it is a medium priority.

4.1.2 Stimulus/Response Sequences

The user will open the command prompt and navigate to the location of the program. They will then run the java program to start the system. After starting the system, it will send a welcome message to the user and prompt them for a file name. After the user types in a filename the system will validate if the filename entered is a valid file. If it is not, the system will display an error to the user and reprompt them for a valid file name. If the filename is valid, the system will display to the user what kind of algorithm will be run on the file and then proceed to process the file and produce results as described in section 4.6. The system will then prompt the user for another filename.

4.1.3 Functional Requirements

1. Prompt the user to enter a filename.
2. Check to see if filename is valid. If not, display an error message and prompt the user again.
3. Display to the user what type of election algorithm is being used if filename is valid.
4. Display prompt to ask the user for another filename.

4.2 System Feature 2 - Read in the Ballot Info File

4.2.1 Description and Priority

To obtain the ballot file info, the user will be prompted by the system to enter a filename. The file should be located in the same directory as the system program. It will contain all of the election information with no mistakes. This file will be provided to the

system via an external voting system, and it is a necessity for the program to run so it is a high priority.

4.2.2 Stimulus/Response Sequences

The user will enter in a filename as stated in section 4.1.2 and the system will check to see if the filename is valid as described in section 4.1.2. When the filename is validated, the system will read in the contents of the file and begin processing the election via either the Instant Runoff Algorithm or the Open Party Listing Algorithm as stated in sections 4.3 and 4.4 respectively.

4.2.3 Functional Requirements

1. The system will check to see if the file is valid.
2. The system will grab the information from the file if it is valid.

4.3 System Feature 3 - Instant Runoff Algorithm

4.3.1 Description and Priority

If the input file/user indicates that the input file has ballot information for an instant runoff election, then the software will calculate the results based on that data. Naturally this will follow the instant runoff procedure. This is a primary functional feature and is thus high priority.

4.3.2 Stimulus/Response Sequences

The user will provide the file of type CSV with the ballot data as specified in section 4.2 of this document. Since it is given to us that the input file itself will have no mistakes and a set format, there are no error messages that can appear at this step. The first line of the input file will be the election type, and if it is “IR” then the software will run the instant runoff algorithm described here. When this happens, the system will report to the user that it is starting to run the instant runoff algorithm. Further system responses are described in sections 4.6, 4.7, and 4.8 of this document. No other user actions are allowed while the algorithm runs.

4.3.3 Functional Requirements

1. Count the number of votes each candidate has using the voters’ first choice option.
2. If a candidate has more than 50% of the votes at any point, declare them as the winner.
3. If no candidate has more than 50% of the votes, remove the candidate with the fewest amount of votes. For all the votes that were counted towards that candidate, move them to the next highest preference candidate who is still in the

running (i.e. if the removed candidate was their first pick and the voter's second pick candidate hasn't been removed, then count the vote to that second pick candidate).

4.4 System Feature 4 - Open Party Listing Algorithm

4.4.1 Description and Priority

If the input file/user indicates that the input file has ballot information for an open party listing election, then the software will calculate the results based on that data. Naturally this will follow the open party listing procedure. This is a primary functional feature and is thus high priority.

4.4.2 Stimulus/Response Sequences

The user will provide the file CSV file with the ballot data as specified in section 4.2 of this document. Since it is given to us that the input file itself will have no mistakes and a set format, there are no error messages that can appear at this step. The first line of the input file will be the election type, and if it is "OPL" then the software will run the open party listing algorithm described here. When this happens, the system will report to the user that it is starting to run the open party listing algorithm. Further system responses are described in sections 4.6, 4.7, and 4.8 of this document. No other user actions are allowed while the algorithm runs.

4.4.3 Functional Requirements

1. Extract the number of votes and the number of seats being filled by this election from the input file..
2. Calculate the quota by dividing the number of votes by the number of seats available in the election.
3. Group all of the independent candidates into a single group for future reference.
4. Count the number of votes for each candidate and also total the number of votes for each party (independents are all in one party).
5. Divide each party's vote total by the quota to get an initial amount of seats for each party. Keep track of the remainders of these divisions.
6. If there are still seats to be assigned, assign one seat to the party with the highest remainder. Repeat this until all seats are assigned but don't assign multiple seats to the same party in this manner. Instead assign the seat to the party with the next highest remainder.
7. For each party assign the candidates of that party with the most votes for as many seats as that party won as the winners. In other words, if the Republican party won 3 seats, then the Republican candidates with the highest, second highest, and third highest vote count respectively are all winners.

4.5 System Feature 5 - Flip Coin to Resolve Ties

4.5.1 Description and priority

When the system counts all the votes for the election and finds that there is a tie between two or more candidates then this feature will be used to randomly select the winner. For Instant Runoff elections, this feature may also be used to determine which candidate is removed from the running when there is a tie between losing candidates. This is a high priority feature because this feature can potentially declare the winner of the election and is required for the Instant Runoff Election algorithm.

4.5.2 Stimulus/ response sequences

Once all the votes are counted successfully if the system finds that there is a tie then this feature will be used. If there is a tie between two candidates one will be eliminated by this feature and the other will be declared the winner. If there is a tie between more than two candidates, The system will randomly pick two candidates and then use this feature to eliminate one. The winner will again participate in the coin flip with another candidate and repeat this process until there is only one winner. However for Instant Runoff elections when a candidate needs to be removed from the running due to there not being a majority, a coin toss determines who is removed if there is a tie for the candidates with the lowest votes.

4.5.3 Functional requirements

1. Select the candidates who received an equal number of votes.
2. Determine whether we need to
    i. Select a winner or
    ii. There is no majority for an Instant Runoff election and there is a tie between the candidates with the lowest amount of votes
3. Flip the coin and determine the winner from selected candidates, or loser in the loser selection case.
4. If there are more than two candidates select the winner from the first coin flip and run coin flip again with another candidate.
5. Repeat the process until there is only one winner.

4.6 System Feature 6 - Results Display

4.6.1 Description and Priority

When the software is complete in counting and calculating all the ballots inputted into the system, the summary of results will be displayed. Since the result is crucial in

determining the winner of the election, it is a high priority feature and must be completely accurate.

4.6.2 Stimulus/ response sequences

When all the votes have been gathered and processed through the system, the results and summary will be displayed on the terminal for the user to see. The user will see the type of election, the number of ballots counted, the candidates, the parties, declaring the winner, and rank the candidates based on percentage.

4.6.3 Functional requirements

1. The software gathers the results and displays them to the terminal screen.

4.7 System Feature 7 - Audit File

4.7.1 Description and priority

Once the system completes all the procedures and determines the winner of the election, it will save all the information in the file. This file is then used to validate the accuracy of the election and the system itself. Since this file is proof of a fair election, it is a high priority feature and must be executed successfully.

4.7.2 Stimulus/response sequences

After successfully computing the result of the election, the system will create a file that includes all the details about the election. After successfully creating a file, it will notify users that the file has been created and provide information about where it is located. If a user quit the process, an audit file will not be created.

4.7.3 Functional requirements

1. Create a file named [Election type_currentDate_currentTime]AuditFile.
2. Populate the file with all the details that includes election information as well as the procedure that system follows to determine the winner of the election.
3. Save the file in system defined location and notify the user that the file has been created and saved.

4.8 System Feature 8 - Press Release File

4.8.1 Description and Priority

After the election results are calculated, some information will be printed to a file. This file will be suitable to give to the media so that they may announce the results of the election. This feature is a low priority as the terminal will display the major pieces of this information. Furthermore, the Instant Runoff and Open Party Listing algorithms must take precedence over this feature as without those the software can’t produce a result.

4.8.2 Stimulus/Response Sequences

Once the software determines the results of the election it will notify the user as such. Then it will create the press release file and write the necessary information to it. Once this is complete, the software will notify the user of such and include the name of the press release file which has a system given name.

4.8.3 Functional Requirements

1. Create a file named "[currentDate_currentTime]PressRelease"
2. Populate the file with any and all information that the media would want. This will include the winners, the parties they represent, the number of votes all candidates got, and the total number of votes. For Instant Runoff, it will include the vote distribution for each candidate at every stage i.e. when the winner is determined and when a candidate needs to be taken out of the running. For Open Party Listing, it will include the number of votes that each party received.
3. Print to the console a message saying that the press release file was successfully created and that gives the name of this file. This file will be located in the same directory as the BCS program.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system needs to be able to process one-hundred thousand ballots within eight minutes. This allows the election official ample time to be able to get the results of the election in order to share them with the media and determine if there needs to be a recount of the votes.

5.2 Safety Requirements

There are no safety requirements we need to fulfill for this software.

5.3 Security Requirements

There are no security requirements we need to fulfill for this software. All these concerns are handled at the voting centers.

5.4 Software Quality Attributes

Usability - The software must be easy to use so election officials with no technical experience can operate with ease.

Reliable - The software must be able to determine the winner of the election with no errors and near accuracy.
Fast - The software must be able to process 100,000 votes in 8 minutes.
Testability - The software must be able to be tested by a user by running a test file
Maintainability/Good Looking - The software code must be visually appealing so it can be understood by testers and developers.

5.5 Business Rules

The election officials will be the only ones to use the ballot counting system. They will be the ones to perform all the functions of the software whenever an election is occurring. The testers will have access and the same permissions as the election officials in special cases.

6. Other Requirements

The BCS program will eventually be merged into a larger online integrated voting system after completion.

Appendix A: Glossary

- BCS (Ballot Counting System): the system which this SRS describes
- IR (Instant Runoff): an election type where voters rank the candidates in the order of their preference and their vote is sent to their top preference after candidates are eliminated.
- OPL (Open Party Listing): an election type where voters vote for their preferred candidate and each party gets seats based on the portion of total votes which went to candidates of that party.
- Audit File: File with information on ballots and that runs through the election process from start to finish.
- Press Release/Media file: File that has the election result and other information that are going to be shared with the media.
- Command Line Interface: The terminal which will prompt the user for information and display information for the user.
- Comma Separated Values File (CSV File): Stores data in a tabular format

Appendix B: Analysis Models

Not applicable

Appendix C: To Be Determined List

Not Applicable