Software Requirements Specification

for

<Voting System>

Version 1.8 approved

Prepared by <Akar Kaung, Hao Wu, Moyan Zhou, Yiming Yao>

<University of Minnesota - Twin Cities>

<02/19/2021>

Table of Contents

Revision History

Name	**Date**	**Reason For Changes**	**Version**
Moyan Zhou	02/06/21	Start editing initial version SRS	1.0
Yiming Yao	02/13/21	Add the system features	1.1
Akar Kaung	02/14/21	Added overall description	1.2
Hao Wu	02/15/21	Add External Interface Requirement	1.3
Yiming Yao	02/15/21	Change the format of the document	1.4
Yiming Yao	02/16/21	Add User Case 4.6 - 4.9	1.5
Hao Wu	02/18/21	Add User Case 4.10, 4.11	1.6
Akar Kaung	02/18/21	Add User Case 4.1, 4.5, 4.12	1.7
Moyan Zhou	02/19/21	Add Rest of Users Case and go over the entire project.	1.8

1. Introduction

1.1 Purpose

The purpose of this software requirements specification is to document a detailed description of the voting system development. Two kinds of voting systems will be developed: Instant runoff voting and Party list voting. This document will explain the features of the system, the design and implementation constraints of the system, the environment under which the system should be run, and expected outputs of the system with given inputs. It is intended for both the users and developers of the voting system.

1.2 Document Conventions

This document was created based on the IEEE template for System Requirement Specification Documents.

1.3 Intended Audience and Reading Suggestions

- Typical users, such as election officials and media representatives, will use this voting system for calculating the voting results with given ballots.

- Programmers and testers are interested in working with the voting system further implementation and maintenance and testing the functionalities.

1.4 Product Scope

The voting system is a tool for users to calculate election results with two kinds of algorithms. The two available algorithms are Instant Runoff Voting and Open Party Voting. Users can specify the ballots (represented as .csv files) and the system will generate an audit file depending on the algorithms. There will be specific illustrations for the calculation if asked from the system.

1.5 References

IEEE Template for System Requirements Specification Documents:

http://goo.gl/nsUFwy

The Voting System Requirements

https://canvas.umn.edu/courses/217849/files/18790482?wrap=1

2. Overall Description

2.1 Product Perspective

This software is new software and it is going to be developed for the media and election officers to use. Media will be able to view the result and audit file of the election process while election officers will be able to use it for counting the votes of election and determining the results. It can only handle the file that includes the voting ballots which follows a specific styling structure with the file format of .csv.

This software system can be only changed or updated if the congress agrees to change the way that the election votes are counted toward the election.

2.2 Product Functions

System perform

- Read the voting file
- Handle tie (flip coin assuming only two candidates will have the same vote)
- Generate result
- Record/Display result
- Generate audit file
- Export audit file (each time the system runs, it will generate a new report)
- handle IR
- handle OPL

User perform

- Close Project: Close the current project
- Import ballot file: Import ballots file with specific file format type
- View:
    - View audit report: View latest audit report
    - View election result: View latest election result

2.3 User Classes and Characteristics

- Election officers - use voting file to generate reports
- Programmers

- Testers
- System

2.4 Operating Environment

Operating Systems (OS)

- Mac OS X
  - Intel Core i5 @ 3.2GHz
  - 16GB RAM
- Windows 10
  - Intel Core i5 @3.4GHz
  - 64GB RAM
- Windows 10
  - Xeon @3GHz
  - 32GB RAM
- Windows 10
  - Intel Core i7 @1.6GHz/ 3.2GHz/ 3.4GHz/ 3.5GHz/ 3.6GHz/ 4.2GHz/ 900Hz
  - 32GB RAM
- Windows 10
  - Intel Core i5 @3.4GHz
  - 32GB RAM/64GB RAM
- Ubuntu
  - Intel Core i7 @ 3.4GHz
  - 16GB RAM
- Ubuntu 20.04
  - Intel Core i5 @800Hz
  - 32GB RAM
- Ubuntu 20.04
  - Intel Core i7 @3.6GHz
  - 32GB RAM

RAM

- Minimum 512 MB

Processor (minimum)

- Intel Core i5
- AMD Ryzen 5
- M3

2.5 Design and Implementation Constraints

Design constraints,

- This software will be developed in Java.

Implementation constraints,

- Input file required .csv format
- It should be able to run 100,000 ballots in under 8 minutes.
- Once the software is released, the main algorithm of the election system cannot be changed unless the election committee/congress agree to do so.
- This program will be run multiple times during the year at normal election times and special elections, thus the program will not be able to update during the election times.

2.6 User Documentation

- This program is usable for both types of election
- There is only 1 voting file for each type of election
- The voting file must be in .csv format
- For open party listing, all independents are grouped into one party.
- If there is ever a tie in the election result, the system will flip a coin which will randomly select the winner in a fair coin toss.

2.7 Assumptions and Dependencies

- The election type will be indicated through the voting file.
- Will be expected to get only one file per election (i.e. all the ballots, candidates, and election type will be merged into one single file).
- The file format will be predetermined and cannot change the structure of the file. (See election_file_format_Team22.pdf)
- There are no errors in the ballots.
- The election file will be located in the same directory as the program.
- Maximum number of 2 candidates for tie voting
- Media will not have access to this software, however, they can view the result/audit file from the election officials
- The machine that is going to be used for this software has latest Java installed
- There is at least one ballot in the file

3. External Interface Requirements

3.1 User Interfaces

All of the following steps will be accomplished in the terminal.

3.1.1 Open by double click Run.sh

After the user double clicks Run.sh, the system will automatically start by prompting the user with the following instructions: “Please tell me what you want to do (import, exit).”

3.1.2 Import file by typing “import”

If the user types in “import,” the system will import the file and start to read; if “exit,” the system will be ended - after reading in the file, prompt the user again by “Reading file complete, do you want to calculate the result? (yes, no)”

3.1.3 Execution:

If “yes,” the system will run for the result automatically.

3.1.4 Result output and location:

After successful execution, the result will be shown in the terminal as well as exported to the system directory automatically with the name of “*result*”.

3.1.5 Generating Audit File :

The audit file will be generated and exported to the system directory automatically with the name of “*auditFile*”.

3.2 Hardware Interfaces

The voting system requires a minimum memory of 512MB and a minimum disk space of 250MB. CPU architecture requires X86 for Windows, X64 or X86 for Linux. For more information about running system requirements, see https://docs.oracle.com/cd/E19830-01/819-4707/abqae/index.html The software is supposed to run on UMN CSE labs machines.

3.3 Software Interfaces

The voting system is able to run on Windows 10, MacOS, and Linux. Latest versions of the operating system are optimal. All ballots will be recorded in a CSV format file. To ensure that the file can be read correctly, the voting system requires Java 7.0 or higher to be installed on the system. For more information about the version of Java, please visit https://www.java.com/en/download/manual.jsp The election file is supposed to be located in the same directory as the program.

3.4 Communications Interfaces

It might require an Internet connection for the latest Java installation and update. Normally, no other APIs or plugins are required.

4. System Feature

4.1 Read Voting File/Ballots

Name:	Read the voting file/ballots
ID:	VTS_01
Description:	Read data from the voting files
Actors:	The system
Organization Benefits:	In order to start the system or run the algorithm, this piece of information is Required from the user.
Frequency of Use:	Everytime when the file is imported.

Trigger:	The system is opened and the format of file is .csv
Precondition:	.csv file is loaded
Postcondition:	Data from the election will be available to be executed.
Main Course:	1. Read 1st line and store it for election type 2. Read 2nd line and store it in number of candidates 3. Read 3rd line and store in candidates info 4. System check if election type is IR (see AC1) 5. Read until last line and for each line store the data into ballot
Alternate Courses:	AC1 election type is not IR 1. read 4th line and store in number of seats 2. read 5th line and store in number of ballots 3. return to Main Course step 5
Exceptions:	This use case doesn't have an exception since we assume that there is no error in this file.

4.2 Handle Tie

Name:	Handle tie
ID:	VTS_02
Description:	Handle tie when there is a tie between two candidates by flipping a coin
Actors:	The system
Organization Benefits:	In order to show the users the result, this is required under circumstances when it is needed
Frequency of use:	Every time when there is a tie between two candidates
Trigger:	When two candidates have the same amount/percentage of votes from the ballots
Precondition:	Assuming the tie will only happen to two candidates
Postcondition:	One candidate will be selected

Main Course:	1. check if two candidates have the same number of votes 2. if yes, then flip a coin 3. assign the result
Alternate Courses:	if two candidates do not have the same number of votes 1. move on to next step
Exceptions:	if the computer where the system runs powers off, the user needs to re-run the system

4.3 Generate Result

Name:	Generate voting result
ID:	VTS_03
Description:	generate voting result with given ballots
Actors:	The system
Organization Benefits:	this case is required by the system
Frequency of use:	every time the user decides to run the voting result
Trigger:	when the user decides to run the voting result
Precondition:	the ballot records have no mistakes and successfully stored in the system
Postcondition:	a correct result with be generated
Main Course:	If IR voting is used: 1. calculate the total votes each candidate gets and it percentage 2. if there is a 50%+ candidate, this candidate won the election 3. if there is not, find the candidate with the lowest percentage of votes 4. eliminate this candidate by transferring this candidate’s second choice votes to the next candidate 5. check again if there is a 50%+ candidate 6. if not, do step 4 again (we see that the nature of this process is a loop from the step description)

Alternate Courses:	If OP voting is used 1. calculate the total votes of each candidate in percentage 2. round up the percentage 3. calculate how many seats the party gets given total available seats 4. arrange top candidates in the party to be elected
Exceptions:	if the computer where the system runs powers off, the user needs to re-run the system

4.4 Record/Display Results

Name:	Record/Display Result
ID:	VTS_04
Description:	display and record the voting result
Actors:	The system
Organization: Benefits:	this case is required by the system
Frequency of Use:	every time the user wants to see the voting result
Trigger:	when the voting result is computed
Precondition:	the system finishes computing the voting result
Postcondition:	A formatted result that is easy to read will be print out in terminal
Main Course:	1. format the result with calculation procedures 2. print to the console/terminal
Alternate Course:	there is no alternative course
Exceptions:	if the computer where the system runs powers off, the user needs to re-run the system

4.5 Generate Audit File

Name:	Generate Audit File
ID:	VTS_05
Description:	Record the log of how the votes goes to each candidates in election
Actors:	The system
Organization Benefits:	Can prove that the voting that goes to the candidates was not fraud
Frequency of Use:	Everytime each vote was counted for candidate
Trigger:	Any candidate vote increases or decreases
Precondition:	Know the type of election to audit
Postcondition:	Audit file is generated
Main Course:	1. Check if audit file exist (see EX1) 2. Check currentFile is true (see AC1) 3. Record all the log that the system run into the audit file 4. If all the log is recorded, then set currentFile to false
Alternate Course:	AC1 currentFile is false 1. Clear all the data in audit file 2. Set currentFile to true 3. Return to Main Course step 3
Exceptions:	EX1 file does not exist 1. Create new file with name audit file 2. Set currentFile to True 3. Return to Main Course step 2

4.6 Export Audit File

Name:	Export Audit File

ID:	VTS_06
Description:	This feature is to export the audit file after election results are completed. Audit file is a document for users to check the processes of the election, it contains many details of election processes, the system allows users to export into the .txt file. The degree of significance is 8, which is priority.
Actors:	The system
Organization Benefits:	This feature can help us
Frequency of Use:	Everytime
Trigger:	User views the audit file then has the option to export.
Precondition:	Users successfully view the audit report after the election is completed.
Postcondition:	The user can see the audit file with detailed information in the same folder.
Main Course:	1. Users successfully view the audit file. 2. The system generates the audit file into the same folder.
Alternate Course:	None
Exceptions:	1. System error causes the exported file lost .

4.7 Close Project

Name:	Close Project
ID:	VTS_07
Description:	This feature is to exit programs by users after evaluating the election results or they can close immediately. The degree of significance is 6, which is normal. The system has the option to close projects after coming out with election results or close immediately. The system will

	ask users to keep running the program or exit immediately. No matter how the system will close by user's instruction.
Actors:	The system
Organization Benefits:	The feature allows the users exit program at any time. The system will not store the results unless the final file is exported. Users will not worry about the file loss or wrong output.
Frequency of Use:	Everytime
Trigger:	No trigger, by users decision.
Precondition:	The program is running normally.
Postcondition:	The program exits normally, and can rerun the program next time.
Main Course:	1. User selects "quit" from the menu 2. The system will close immediately
Alternate Course:	1. The system will automatically quit when the tasks are completed.
Exceptions:	Program shuts down abnormally, checking the hardware usage or JDK error.

4.8 Import Ballot File

Name:	Import Ballot File
ID:	VTS_08
Description:	This feature is to allow users to import ballot files into the system. The degree of significance is 9, which is a high priority. The system will ask the user to import necessary sources. If the system receives the file from the user, the system will let the user know and continue the next step. Otherwise, the system will ask again to the user until all the sources are received.
Actors:	The system

Organization Benefits:	The feature is a basic requirement for this system. System asks the data from the imported file and evaluates the election result for the user.
Frequency of Use:	Everytime
Trigger:	When the system starts normally, the users will be asked to import the ballot file automatically. Unless, users quit immediately.
Precondition:	System will be opened/ compiled by users normally.
Postcondition:	System correctly receives the file and goes to the next step process.
Main Course:	1. System prompt user to input the vote file name 2. System get input from user 3. System search the file from the system file directory (see EX1) 4. System import the file from the system file directory
Alternate Course:	1. Users click Run.sh to compile the program. 2. Users choose “exit,” the system will be ended.
Exceptions:	EX1 File does not exist 1. System prompt user that file does not exist 2. System prompt user to enter the file name again 3. Return to Main Course 2

4.9 View

Name:	View
ID:	VTS_09
Description:	This feature is to allow users to review the latest audit report and election result report. The degree of significance is 9, which is a high priority. After counting votes in each candidate, the system will generate an audit report and election result report for users to view. System will allow users to view all the processes during evaluation. The results must follow the election rule and authority by voting system.

Actors:	Election officers, testers
Organization Benefits:	The system provides a relative audit report and election report, users accept the election result by receiving the detailed information.
Frequency of Use:	Very Frequent once the election is completed.
Trigger:	After the evaluation process is approved and complete. The System will allow users to view this information.
Precondition:	Election evaluation process completed.
Postcondition:	The user can see the audit/ result report on their screen.
Main Course:	1. User types“view” from the menu 2. User selects which reports to see 3. The selected report information on the screen 4. user close the report/ select another report.
Alternate Course:	1. User does not want to view the reports on terminal 2. The election result will be exported into a .csv file in the same dictionary.
Exceptions:	Evaluation process in this system has errors, users cannot enter the “view” section.

4.10 Handle IR

Name:	Instant Runoff Voting
ID:	VTS_10
Description:	voters rank the candidates in the order of their preference. All the number one preferences of the voters are counted. The candidate who receives over 50% of the total votes will win.
Actors:	The system
Organization Benefits:	System can count much faster than humans and reduce the risk of mistakes.

Frequency of Use:	Each time for Instant Runoff election
Trigger:	When type of election is Instant Runoff
Precondition:	.csv file is imported and have read by system
Postcondition:	Result will be shown in terminal and audit file is generated
Main Course:	1. voters rank the candidates in the order of their preference. All the number one preferences of the voters are counted. 2. If a candidate receives over 50% of the first choice votes, he or she is declared elected. 3. If step 2 did not happen. The candidate with the fewest votes is eliminated. The ballots of supporters of this defeated candidate are then transferred to whichever of the remaining candidates they marked as their number two choice. 4. repeat step 2 and 3, until one candidate receives a majority of the counting votes and wins the election.
Alternate Course:	None
Exceptions:	If there is ever a tie, flip a coin. You must randomly select the winner in a fair coin toss.

4.11 Handle OPL

Name:	Open Party List Voting Algorithm
ID:	VTS_11
Description:	Voters must cast a vote for an individual candidate and vote counts for the specific candidate as well as for the party. The order of the final list completely depends on the number of votes won by each candidate on the list. The most popular candidates rise to the top of the list and have a better chance of being elected.
Actors:	The system
Organization Benefits:	System can count much faster than humans and reduce the risk of mistakes.

Frequency of Use:	Each time for open party list election
Trigger:	When type of election is open party list
Precondition:	.csv file is imported and have read by system
Postcondition:	Result will be shown in terminal and audit file is generated
Main Course:	1. voters only can vote for individual candidates, votes in the same party will be counted totally for determining seats. 2. quota is calculated by (total votes) / (number of seats) 3. First allocations of seats for each party will be calculated by (total votes) / (quota), the quotient is the number of seat party gains in first allocation. 4. If there were n seats left after first allocations. system will rank parties based on remaining votes which is calculated after first allocation. Top n party(s) will gain one seat each.
Alternate Course:	None
Exceptions:	If there is ever a tie, flip a coin. You must randomly select the winner in a fair coin toss.

4.12 Program start

Name:	Program start
ID:	VTS_12
Description:	Start the program
Actors:	Election officer, testers
Organization Benefits:	System will now know what to do
Frequency of Use:	Everytime the program start
Trigger:	User open the program

Precondition:	- RAM availability - Storage size availability
Postcondition:	- system know what user want to do
Main Course:	1. System prompts "Please tell me what you want to do (import, exit)" 2. Get user input 3. Check if the input is "import" (see AC1, EX1) 4. Go to use case with ID: VTS_08 5. System prompts "Do you want to run this program again? [y/n]" 6. Get user input 7. Check if the input is "y" (see AC2, EX2) 8. Go to use case with ID: VTS_07
Alternate Course:	AC1 Input is "exit" 1. Redirect to use case with ID: VTS_07 AC2 Input is "n" 1. Return to Main Course step 1
Exceptions:	EX1 Input doesn't match with either "import" or "exit" 1. System prompts "[!] Invalid input, please type again." 2. Return to Main Course step 1 EX2 Input doesn't match with either "y" or "n" 1. System prompts "[!] Invalid input, please type again." 2. Return to Main Course step 5

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The voting system requires a minimum memory of 512MB and a minimum disk space of 250MB. CPU architecture requires X86 for Windows, X64 or X86 for Linux. Besides, 128 megabytes of RAM is required as minimum. The election file is supposed to be located in the same directory as the program. An election should be able to run 100,000 ballots in under 8 minutes. For more details, see section 2.4 and 3.2.

5.2 Safety Requirements

There is no particular safety requirement for this program.

5.3 Security Requirements

- There is no particular security requirement for this program.
- Ensure one vote for one person is handled at the voting center.

5.4 Software Quality Attributes

The Voting System provides a simple interface (through terminal). Users don’t need any knowledge of programming to use it; however, having some familiarity with election algorithms will help them to understand the results from the system.

5.5 Business Rules

There are no particular functions under specific circumstances; the system itself is straightforward.

6. Other Requirements

Appendix A: Glossary:

References: https://docs.oracle.com/cd/E19830-01/819-4707/abqae/index.html

https://www.java.com/en/download/manual.jsp

http://goo.gl/nsUFwy

https://canvas.umn.edu/courses/217849/files/18790482?wrap=1

Appendix B: Analysis Models

N/A

Appendix C: To Be Determined List

N/A