Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Ashton Berg (ber00063), Caleb Tracy (tracy255), Elias Caceres (cacer019), and Garrett Abou-Zeid (abouz009)

University of Minnesota, S23 CSCI 5801 (001)

Team 25

February 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Revision 1.0	2/16/2023	Initial Requirements Specification	1.0

1. Introduction

1.1 Purpose

This document's purpose is to define the first version of a system that can manage elections to unbiasedly and consistently determine the winner of instant runoff votes and closed-party list elections. The document will provide a detailed description of the software describing the goals, parameters, and constraints it will operate under in their entirety.

1.2 Document Conventions

This document followed the IEEE template for System Requirements Specification Documents. Section headers are labeled in 18-point Times font, subsection headers in 14-point Times font, all other headers in 11-point Times font, and body text will appear in 11-point Sans Serif. System features will have their level of priority stated. Functional requirements will hold priority based on length of details given in their respective use cases.

1.3 Intended Audience and Reading Suggestions

This document is organized into six sections, the first of which is the introduction which states important information for the readers. Section two extends to give a system description and expectations for users. Section three outlines the characteristics of each interface. Section four organizes the functional requirements for the product and presents use cases. Section five organizes the non-functional requirements for the product. Lastly, section six defines further requirements if applicable. Each section is organized into subsections, the reader should review the table of contents and determine which subsections are most relevant to their needs.

- The intended users, election officials, who want to use this system for processing ballots and receiving results should read section two which discusses system requirements and use cases, as well as use cases in section four.
- Programmers who wish to contribute to this project or are interested in the work should read the entirety of this document with a focus on sections two through five.
- Testers trying to identify bugs and document expected behavior should review expected use cases in section four.
- Members of the media who wish to learn more about the system can read sections one, two, and three for detailed information.

1.4 Product Scope

The voting system defined in this document can be used to analyze provided ballots and produce results for either close party listing or instant runoff voting elections. Users will provide validated ballots and election information. The system is designed as a tool for election officials to automate the process of counting ballots and calculating losers, which is very labor intensive. The system will consistently and fairly determine election results documenting the process in an audit log to avoid any election tampering or other interference. The benefits of this system include the elimination of unbiased decisions and the reduction of work hours for election officials.

1.5 References

Voting system's GitHub page:

https://github.umn.edu/umn-csci-5801-01-S23/repo-Team25

IEEE Template for System Requirement Specification Documents:

https://goo.gl/nsUFwy

2. Overall Description

2.1 Product Perspective

The Voting System product is being developed for use by election officials to determine the results of an election. It is a self-contained product that will not be integrated into an existing system, and it is developed to run on machines meeting the specifications of a University of Minnesota Keller Hall lab machine. Using the software will guarantee accurate election results that are computed quickly. It is designed to process comma-separated value (CSV) files to determine the election type and winner.

2.2 Product Functions

System flow overview:

- System execution and startup.
- CSV file name identification.
- Collect and identify election details.
- Determine vote results depending on election type.
- Fairly handle non-majority cases for IR.
- Break ties fairly.
- Creation of an audit file.
- Write election information to the audit file depending on election type.
- Display election results to the user.

2.3 User Classes and Characteristics

- Programmers who have interest working on the project through further development and fixing bugs in the code base.
- Testers that are working with the software and making sure that the program is producing accurate and correct results.
- Election officials who want to use the voting system software for determining winners of elections.

2.4 Operating Environment

The voting system will be run on CSE IT Labs machines. The system will be run on their general CSE Ubuntu computing machines. These machines run on an operating system of Ubuntu 20.04.5 LTS.

2.5 Design and Implementation Constraints

- Source files for Java will be run through terminal or eclipse.
- Voting System will run on CSE Labs Machines.
- There will only be one program to handle both voting methods.
- The software will not modify any files outside of the program.
- The system needs to run 100,000 votes within 4 minutes.

2.6 User Documentation

Additional help and information can be found on our GitHub README file:
https://github.umn.edu/umn-csci-5801-01-S23/repo-Team25/blob/main/README.md

2.7 Assumptions and Dependencies

The voting system is written in Java and therefore requires Java 17 and the corresponding Java Runtime Environment in the user's environment. The user's environment is assumed to be a machine in one of the Computer Science and Engineering computer labs within the University of Minnesota Twin Cities. Users must have valid University of Minnesota credentials in order to run the application on the command line. For guaranteed correct program execution, this must be a lab machine running Ubuntu 20.04. For example, one of the lab machines at the specified location.

University of Minnesota, Twin Cities. Keller Hall 1-250.

- Operating System: Ubuntu 20.04
- Intel Core i7 @ 2.5 GHz (x8)
- 32 GB RAM

3. External Interface Requirements

3.1 User Interfaces

A simple graphical user interface (GUI) will be used in order for the user to select the election ballot file to be processed.

3.2 Hardware Interfaces

The program will run on CSE lab machines that compute with a four core Intel i7-8700 running at 3.20 GHz with 32 GB of RAM. This vote processing system will mainly work on computational tasks thus an external graphics processing unit (GPU) will not be necessary.

3.3 Software Interfaces

This voting software requires the Java Development Kit (JDK) to be installed on the computer if it is to be locally compiled. The Java Runtime Environment (JDE) is needed in order to run the voting program.

3.4 Communications Interfaces

The product will have no communicating features. All computation and data exporting is done locally on the system.

4. System Features

4.1 Voting System Input

4.1.1 Description and Priority

The voting system will take in a single CSV file at a time containing necessary election information. Such includes the ballot data, candidates, election type, and other data specified in *B.3.* This feature is high priority because the system as a whole cannot function without it.

4.1.2 Stimulus/Response Sequences

Users are expected to start the program and will be prompted for a CSV filename. See *B.2* for more information on user actions.

4.1.3 Functional Requirements

REQ-1: The user can start the program on the command line. See use case *B.1*.
REQ-2: The user is prompted for the name of the file. See use case *B.2*.

4.2 Ballot Processing

4.2.1 Description and Priority

The system will process the provided CSV file. First, the information within the CSV file is analyzed and stored within data structures with votes related to parties and candidates based on the election type. Once all ballots are read depending on the election type, losers are decided by the system and a winner is chosen.

4.2.2 Stimulus/Response Sequences

Users take no action, the process of counting ballots and determining losers is an automatic response to the system receiving a valid CSV file.

4.2.3 Functional Requirements

REQ-1: CSV files are complete and correct in the information they provide.
REQ-2: Identifying information is automatically recorded without user input. See use case *B.3*.
REQ-3: Losers are determined based on election conditions for victory. See use cases *B.4* and *B.5.*
REQ-4: In the case of a non-majority for CLP elections, a candidate must be determined as the loser. See use case *B.6.*
REQ-5: In the case of a tie, the loser is determined by a fair coin flip. See use case *B.7.*

4.3 Voting System Output

4.2.1 Description and Priority

The process of feeding results to users involves the creation of a read-only audit file to prevent tampering. Results are also shown on the command line for immediate review.

4.2.2 Stimulus/Response Sequences

Users will be prepared and able to see results from a terminal window.

4.2.3 Functional Requirements

REQ-1: The system can write into the audit file different information depending on the election type. See use case *B.8.*

REQ-2: An audit file is produced with all election information and calculations made. Winners are clearly indicated. See use cases *B.9* and *B.10.*

REQ-3: The audit file shows how the election progressed, and how to replicate the ballot processing. See use cases *B.9* and *B.10.*

REQ-4: Winners and election information are displayed to the terminal for review by the user. See use case *B.11.*

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Our voting system software requires at least a 2.5 gigahertz CPU with 8 cores and 32 gigabytes of RAM. No dedicated graphics card is required. Performance depends on the number of ballots being processed and as a result, the system requirements for a large number of ballots are more demanding. The system must be able to handle processing 100,000 ballots in under four minutes.

5.2 Safety Requirements

The voting system has no safety requirements regarding the use of the voting system software.

5.3 Security Requirements

The voting system has no security requirements. Security is left to the user, such as in the case of voters only submitting a single ballot for an election. Users are free to use the software without additional privileges outside of the ability to execute and use the software.

5.4 Software Quality Attributes

The most important quality characteristic for the product is correctness. Integrity within elections is very important to voters, and election results are frequently challenged or questioned. The product needs to reliably return the correct winner of an election every time, as issues stemming from an incorrect result can be detrimental to users. Election results should be clear, and it should be clearly indicated when a tie has occurred and how the winner was chosen to assure users of correctness. The results should also be able to be recreated in case their validity is challenged.

The main users of the product will be election officials, who may not be very experienced in using technology or applications. Usability will be an important quality attribute so that any election official, regardless of their experience, will be able to properly determine election results. Clear messaging should be used to provide simple instructions that make use of the product easy.

5.5 Business Rules

Any user has full access to the system’s services. The system cannot be used to process more than one election simultaneously. An election must be fully processed with results, with an audit file generated, before another election may be processed by the system.

6. Other Requirements

Appendix A: Glossary

IR	Instant Runoff
CPL	Closed Party List
CSV	Comma Separated Value
IEEE	Institute of Electrical and Electronics Engineers
Java	Popular object-oriented programming language.
Java Runtime Environment	Set of components that are required to run an application made with Java.
Ubuntu 20.04	Version of the open-source Linux operating system Ubuntu.
GUI	Graphical User Interface

Appendix B: Use Cases

B.1 UC_001

Name	Run Voting System
ID	UC_001
Description	The user executes the voting system.
Actors	Election officials, System Testers
Benefits	Program startup allows users to interact with services.
Frequency of Use	Program is used multiple times throughout the year during normal and special elections. The system should not be run again before results are provided for the previously or currently being processed election as such behavior is undefined.
Triggers	User executes the program from the command line by running an executable jar file.
Preconditions	User has an environment that meets the requirements listed in subsection 2.7.
Postconditions	The program executes without interruption and is ready to prompt the user for information and the provided CSV filename.
Main Course	1. Locate the executable jar file. 2. Double-click the executable jar file, or right-click and select ‘run’. 3. If a prompt appears requesting permission to execute the jar file, provide permission. 4. The system begins execution.
Alternate Courses	AC1: Execution from terminal 1. Open a terminal through the start menu, or on Ubuntu 20.04 using the shortcut ‘Ctrl+Alt+T’. 2. Move to the desired directory containing the jar file using the ‘cd’ command. 3. Execute the jar file using the command ‘java -jar FILENAME.jar’ where FILENAME is replaced by the executable’s file name.
Exceptions	EX1: Java command is not recognized 1. In the terminal, use the command ‘java -version’ to verify OpenJDK Runtime Environment version 17.0.5 is in use.

	EX2: File is not recognized 1. Verify the correct file name was provided for the executable, the name is case-sensitive.

B.2 UC_002

Name	Identify CSV File Name
ID	UC_002
Description	The user identifies the name and location of the CSV file containing ballot and election information.
Actors	Election officials, System Testers
Benefits	CSV files can be produced from any data collection or spreadsheet program. Users can easily supply election information and the system can provide election processing services.
Frequency of Use	On every use of the system, a valid CSV file name must be provided.
Triggers	Automatic trigger when the program is executed and ready to receive input.
Preconditions	User has a valid CSV file located in the same file system as the executed jar file. The user has terminal access and has run the executable. No more than one file is given. CSV file is read-only.
Postconditions	The CSV file is found by the system, and the system is ready to process the contents of the CSV file.
Main Course	1. The system opens a terminal if needed, and requests input for the file name with its full path name. 2. User provides pathname and CSV file name without spaces in a single argument. 3. System locates the CSV file.
Alternate Courses	AC1: Execution from terminal 1. The user inputs the CSV file's pathname as the first and only argument to the jar file. 2. CSV file must be located in the same directory as the jar file if the full pathname is not provided.
Exceptions	EX1: File not found 1. Verify the correct file name or pathname was provided for the executable, the name is case-sensitive. EX2: Incorrect file type 1. The file is located, but the file type is incorrect or corrupted. 2. System informs the user an error was encountered obtaining the file, prompting for a file of file type CSV. 3. The user is able to fail the upload and receive an error an infinite number

	of times. The program will not self-terminate in this case.

B.3 UC_003

Name	Identify Election Details
ID	UC_003
Description	From the provided CSV file, details on the election such as type, candidates involved, seats, and other information outside of the ballots themselves are extracted.
Actors	System Testers, Developers
Benefits	The process of attributing votes to specific parties or candidates, and the number of seats in the case of CLP, is streamlined by the system.
Frequency of Use	On every use of the system, a CSV file is processed for information.
Triggers	Automatic trigger from the input of a valid CSV filename.
Preconditions	Ballots have no errors as all ballots are assumed valid and correct. Every ballot has at least one ranking. If the election is of type IR, the first line within the CSV file is ‘IR’. The second list holds the number of candidates. The third line identifies each candidate with the candidate's last name, a single space, and their party identifier enclosed in parenthesis. Candidates are separated by a comma and a single space. The fourth line is the number of ballots in the file. CPL elections will have ‘CPL’ on the first line of their generated CSV file. The second line is the number of parties, and the third line will identify party names separated by commas. The following lines identify the candidate’s last names for each party in order of their party name on the third line, with each party having its own line. After the candidate names, the number of seats and the number of ballets are each given their own new line.
Postconditions	Candidates are associated with no more or less than a single party. Candidates are associated with a ranking on the ballots in the CSV file, and the system is prepared to count ballots and associate each one with their specified candidates. In the case of a CPL election, multiple candidates are associated with the same party.
Main Course	1. All steps are performed automatically by the system. 2. The first line of the CSV file is read. The system identifies the election type. 3. The second line of the CSV file is read. The system identifies the number of candidates.

	4. See AC1 if the election type is IR. See AC2 if the election type of CPL. 5. The number of ballots is recorded by the system.
Alternate Courses	AC1: Instant Runoff Election 1. Candidates and parties are recorded internally. 2. Candidates are associated with one party. Multiple candidates are not associated with the same party. AC2: Close Party List Election 1. The number of seats available is recorded internally. 2. Parties are identified, and multiple candidates are associated with each party.
Exceptions	N/A

B.4 UC_004

Name	Determine Instant Runoff Voting Results
ID	UC_004
Description	The product determines the results of an election using an Instant Runoff (IR) voting system
Actors	System testers
Benefits	- Allows for election results to be determined in an IR election - Will provide information necessary for the audit file, including election winner and number of votes each candidate received
Frequency of Use	IR is one of two voting systems supported by the product. We aren't told what portion of the time IR will be used, so we'll assume that half of the time the product is used, this case will be used.
Triggers	Identify Election Details use case determines election type is IR
Preconditions	- Election file is in the same directory as the program - All ballots have at least 1 person ranked - There are no write ins - Each ballot has at least one candidate ranked as their top choice - Identify Election Details use case extracted information correctly
Postconditions	An election winner and number of votes each candidate received is determined
Main Course	1. The system counts the number of first-choice votes each candidate received 2. The system determines who received over 50% of first choice votes (See AC1 and AC2) 3. The winner is chosen
Alternate Courses	AC1: No candidate earned over 50% of first-choice votes - Execute Handling Non-Majority Rankings For IR use case - Go to Main Course step 1
Exceptions	N/A

B.5 UC_005

Name	Determine Closed Party List Voting Results
ID	UC_005
Description	The product determines the results of an election using a Closed Party List (CPL) voting system.
Actors	
Benefits	1) Allows for election results to be determined in an IR election 2) Will provide information necessary for the audit file, including election winner and number of votes each candidate received
Frequency of Use	CPL is one of two voting systems supported by the product. We aren't told what portion of the time CPL will be used, so we'll assume that half of the time the product is used, this case will be used.
Triggers	Identify Election Details use case determines election type is CLP
Preconditions	- Election file is in the same directory as the program - Each ballot has only one party chosen for their vote - All independent groups have a single person in the group - Separate independent groups have different names - There are no write ins - Identify Election Details use case extracted information correctly
Postconditions	An election winner and number of votes each candidate received is determined.
Main Course	1) The system counts the number of votes each party received 2) The system calculates a quota by dividing the number of votes that were cast by the number of available seats 3) The system calculates the quotient and remainder for each party by dividing the number of votes that party received by the quota 4) System gives each party the number of seats equal to its quotient found in step 3 5) System gives out the remaining seats in order of the remainder for each party found in step 3 until all seats have been allocated (See AC1) 6) If a party won X seats, the system chooses the first X candidates from their party list to occupy the seats. This is done for each party.

Alternate Courses	AC1: There is a tie for remainders between parties - System executes Tie use case to determine tie winner - Tie winner is allocated a remaining seat first - Continue Main Course 5
Exceptions	N/A

B.6 UC_006

Name	Handling Non-Majority Rankings For IR
ID	UC_006
Description	Eliminates the candidate with the lowest first place votes. That candidate's ballots are transferred to their next available choices.
Actors	System testers
Benefits	No new ballots need to be filled in because runner-up options are already marked on the ballots.
Frequency of Use	Any time there is not a majority vote in an instant runoff voting.
Triggers	When no candidate has received more than 50% of the vote.
Preconditions	Run Voting System has been executed and ballots have been counted.
Postconditions	A candidate is eliminated and their ballots have been reallocated
Main Course	1) Identify candidate with lowest first-place votes (See AC1 and AC2) 2) Identify the ballots of this candidate 3) Eliminate the candidate 4) The identified ballots have their vote transferred to a remaining candidate they put as their next choice. If their next choice has already been eliminated, then vote goes to the choice after, and so on. 5) The system keeps track of the number of votes that were transferred and to who
Alternate Courses	AC1: There is a tie for fewest first-choice votes - System executes Tie use case to determine tie winner - System eliminates other candidate - Return to next step AC2: All votes have been handed out and there is not a clear majority - System chooses the candidate with the most votes to be the winner (See AC3) AC3: There is a tie for popular votes - Tie use case determines tie winner - Go to Determine Instant Runoff Results use case Main Course step 3
Exceptions	N/A

B.7 UC_007

Name	Tie
ID	UC_007
Description	Used to handle situations where a tie has occurred and one candidate or party needs to be chosen over the other.
Actors	System testers
Benefits	Properly handling ties will ensure that election results are accurate, unbiased, and fair in every situation. It is also more efficient and less biased to have the product determine a winner in the case of a tie than have a person do it.
Frequency of Use	Any time a tie occurs in either an IR or CPL election.
Triggers	- There is a tie for first-choice votes in the Determine Instant Runoff Voting Results use case - There is a tie for fewest votes in the Determine Instant Runoff Voting Results use case - There is a tie for popular vote in the Determine Instant Runoff Voting Results use case - There is a tie for remainder votes in the Determine Closed Party List Voting Results use case
Preconditions	The Determine Instant Runoff Voting Results use case has been executed or the Determine Closed Party List Voting Results use case has been executed.
Postconditions	A tie winner is returned
Main Course	1) System assigns one tied party ‘heads’ and the other tied party ‘tails’ (See AC1) 2) System performs a coin flip 3) The result of the coin flip determines the tie winner
Alternate Courses	AC1: The tie consists of more than two parties/individuals - System performs coin flip ‘tournament’ to determine the first place winner and last place loser
Exceptions	N/A

B.8 UC_008

Name	Create Audit File
ID	UC_008
Description	Creates a .txt file to store election results. The file has a name in the format of <election type><date>.txt
Actors	System testers
Benefits	The audit file will allow election officials to share the results of an election. It will also provide a clear view of how the election proceeded and justify the results to avoid controversy.
Frequency of Use	Every time the system is used
Triggers	Determine Closed Party List Voting Results finishes execution Determine Instant Runoff Voting Results finishes execution
Preconditions	A file with the same name doesn't already exist in the directory
Postconditions	A text file with the name <election type><date>.txt will be created in the same directory as the product
Main Course	1) System creates a text file in the same directory as the product. The text file has the name <election type><date>.txt, Ex. cpl021623.txt for a CPL election ran on February 2, 2023. (See EX1)
Alternate Courses	N/A
Exceptions	EX1: System fails to create file - System displays message that it failed to create the file, trying again - Go to Main Course 1

B.9 UC_009

Name	Write to Audit, IR
ID	UC_009
Description	Information about an IR election including the number of candidates, candidates, number of votes, number of votes per candidate, how the election progressed, and election results will be written to a text file.
Actors	System testers
Benefits	Filling the audit file with every step of the IR election process will allow for a clear representation of how the election proceeded so that the results may be justified.
Frequency of Use	Every time the product is used to run an IR election
Triggers	Create Audit FIle use case finishes execution
Preconditions	- Identify Election Details use case correctly extracted all election information - Determine Instant Runoff Voting Results use case produced correct election results and stored the necessary information (winner, vote counts, eliminations, tie occurrences) - Create Audit File use case named the file correctly
Postconditions	The audit file, ir<date>.txt, will contain all of the needed election information in regards to the CPL election
Main Course	1) System creates a text file for writing to with name in format of ir<date>.txt 2) System writes type of election as IR to text file 3) System writes total number of ballots cast to text file 4) System writes each candidate and their party to text file 5) System writes the number and percent of votes each candidate received to text file 6) System writes the winner to text file
Alternate Courses	AC1: Handling Non-Majority Rankings For IR use case was used - System writes which candidate was eliminated to text file - System writes where the eliminated candidates ballots were dispersed to to text file - Go to Main Course step 5 AC2: Tie use case was used - System writes that a tie occurred to text file

	- System writes where and between who the tie occurred to text file - System writes who won the tie to text file - If the tie determined an election winner, go to Main Course 6 - If the tie determined an eliminated candidate, go to AC1
Exceptions	EX1: System fails to create text file - System displays message that file creation failed - Go to Main Course step 1 EX2: System fails to make a write to text file - System displays message that a write to file failed - Return to same step and reattempt the write

B.10 UC_010

Name	Write to Audit, CPL
ID	UC_010
Description	Information about a CPL election including the number of seats, candidates, number of votes, number of votes per party, how the election progressed, and election results will be written to a text file generated by the system.
Actors	System testers
Benefits	Filling the audit file with every step of the CPL election process will allow for a clear representation of how the election proceeded so that the results may be justified.
Frequency of Use	Every time the product is used to run a CPL election
Triggers	Create Audit File finishes execution
Preconditions	- Identify Election Details use case correctly extracted all election information - Determine Closed Party List Voting Results use case produced the correct election results and stored the necessary information (vote counts, seat allocation order, tie occurrences) - Create Audit File use case named the file correctly
Postconditions	The audit file, cpl<date>.txt, will contain all of the needed election information in regards to the CPL election
Main Course	7) System creates a text file for writing to with name in format of cpl<date>.txt 8) System writes type of election as CPL to text file 9) System writes the total number of available seats to text file 10) System writes total number of ballots cast to text file 11) System writes the number of parties involved to text file 12) System writes all parties that were in the election to text file 13) System writes the nominated candidates for each party to text file 14) System writes number of votes each party received to text file 15) System writes the quota used in the election to text file 16) System writes the initially allocated seats to text file 17) System writes the remainders of each party to text file 18) System writes where remaining seats were allocated to text file (See AC1) 19) System writes final allocated seat counts to text file 20) System writes the candidates chosen from each party to text file
Alternate	AC1: A Tie use case occurred during Determine Closed Party List Voting

Courses	Results use case - System writes that a tie occurred to text file - System writes the results of the Tie use case to text file - Go to Main Course step 13
Exceptions	EX1: System fails to create text file - System displays message that file creation failed - Go to Main Course step 1 EX2: System fails to make a write to text file - System displays message that a write to file failed - Return to same step and reattempt the write

B.11 UC_011

Name	Display Result Page
ID	UC_011
Description	The product displays the results for the operator of the software to view.
Actors	System testers
Benefits	The results will be displayed with the victor of the vote. This will also show voting numbers and ordering of most to least votes.
Frequency of Use	The results will be displayed once per voting file run.
Triggers	When the software is finished counting votes and determining winners, the results will be displayed after.
Preconditions	Run Voting System has been executed without error. All data from the input file is working correctly.
Postconditions	An election winner and other candidate positions determined and displayed for the user to view.
Main Course	- The system determines the winners of the votes. - The system will show if any ties or alternative methods of voting that was used in the software run - The system will display the results with number of votes/or tie results
Alternate Courses	- No alternatives. Results are to be shown every run.
Exceptions	N/A