Software Requirements Specification

for

Voting System

Version 01

Prepared by:

Sami Frank (fran0942)
Jake Waro (warox001)
Declan Buhrsmith (buhrs001
Allison Miller (mill7079)

University of Minnesota- Twin Cities: CSCI 5801

2.12.2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Sami	2/14/20	Initial additions to section 5	0.0.01
Sami	2/15/20	Adds use cases 4.3 and 4.4	0.0.02
Jake	2/15/20	Add use cases 4.1 and 4.2, Sections 1.1 - 1.5	0.0.03
Allison	2/16/20	Add use cases 4.5 and 4.7	0.0.04
Jake	2/18/20	Add terms to Appendix A & ToC Formatting	0.0.05
Allison	2/19/20	Section 2	0.0.06
Sami	2/20/20	Additions to section 5	0.0.07
Declan, Sami, Allison, Jake	2/20/20	Clean up and review, Section 3	0.0.08
Jake	2/20/20	Add use cases 4.8 Quit and 4.9 Start Program	0.0.09
Sami	2/21/20	Finalizes section 5, use cases 4.3, 4.4	0.0.10
Jake	2/21/20	Finalize section 1 and use cases	0.0.11
Allison	2/21/20	Final submission checking, last minute formatting	0.0.12

1. Introduction

1.1 Purpose

This document outlines the requirement specifications for the Voting System. The Voting System is a single, stand-alone system. The purpose of this program is to process ballots for an election and display the results of that election. The program can process ballots for elections using either plurality voting or single transferable voting (STV).

1.2 Document Conventions

This document follows the Institute of Electrical and Electronic Engineers (IEEE) software requirements specification (SRS) template. **Bold** text has been used to indicate section headers and to highlight important context.

1.3 Intended Audience and Reading Suggestions

This project is designed for use by **voting election officials**. Other audiences that read this document may include **developers/testers** that are tasked with upholding election integrity, who will read this document to validate the election processing methodology and the required summary information from the ballot processing. They will utilize the testing functionality and will also be responsible for the program's maintenance.

A suggested reading path for the voting election official is to start with section 1. Introduction to get a brief introduction to the product. Secondly, they should continue onto 4. System Features to gain an understanding about the different interactions they may experience with the software and what the expected courses of action are. Lastly, the user should continue onto step 5. Other Non-functional Requirements to gain an understanding about big picture requirements for the program.

A suggested reading path for a developer/tester, would be to read the document in a sequential order. The testers will gain an introduction to the product, understanding how each functional requirement will work, how the program will work with any other interfaces, as well as how the user will interact with the program and any big picture requirements on the software.

1.4 Product Scope

The Voting System processes election ballots to determine the winner(s) and loser(s) of an election using either a plurality or an STV algorithm. It keeps an audit trail that can be followed to understand the given results of an election. The program returns summary information including the type of vote, the number of ballots, the number of seats, and the number of candidates. It also presents information on the winners and losers accompanied by their respective percentage of votes received (for plurality voting) or the ordering of winners and losers (for STV).

The Voting System is intended to be an easy program to learn how to use, with an expectation that it will take less than 15 minutes to teach a user how to use it.

This product creates many benefits for the election process. To start, the Voting System is reusable for future elections. It is versatile and can be used for both plurality and STV models. The program allows for testing the processing of ballots, as well as automating the task of processing the election, which will decrease the amount of time it takes to process ballots. The tool is also beneficial due to its ability to summarize the results and also provide an audit trail to essentially walk through how those results were generated.

1.5 References

IEEE Software Requirements Specification Template: https://ieeexplore.ieee.org/document/278253

2. Overall Description

2.1 Product Perspective

The voting system is a new, self-contained system used for calculating the results of an election with either a plurality voting or a single transferable voting algorithm. The product is not part of a larger system, though ballots will be transferred to the system by secure means outside of this system.

2.2 Product Functions

Users:

- Vote selection: User chooses the voting algorithm to use for the election (plurality or STV).
- Seats: User enters the number of seats for the election.
- Files: User enters the names of the files containing the ballots.
- Run: User runs the election.
- Test: (Developer only) User selects the option to turn the STV shuffle off, enabling testing as the result is predictable given the ballots

System:

- Number of candidates and ballots: System determines the number of candidates and the number of ballots from the given ballot files
- Prompting: System prompts the user to input information, such as the number of seats, algorithm to use, and the names of the ballot files
- Run plurality election: System runs an election using the plurality algorithm; the system will assign the ballots to each candidate and declare the candidate with the most ballots the winner
- Run STV election: System runs an election using the STV algorithm; the system first calculates the Droop quota and then proceeds to use the STV algorithm to distribute ballots and declare winners and losers accordingly. See Appendix A.
- Results: System displays a screen to the user with a snapshot of the results, including the winners and losers with their vote percentages
- Audit: System stores a verbose account of the election in an audit file, including the result of the shuffle if STV is used as well as all winners and losers and the ballots they received
- User interface: System communicates with user through a text-based user interface for ease of use
- Help: System provides a help window upon request if a user requires assistance with the program

2.3 User Classes and Characteristics

The two user classes that will use the voting system are election officials and developers/testers. Both types of users are expected to be able to use a simple user interface and to have knowledge of the election process.

- **Election Officials** - adults in charge of overseeing an election. Election officials will be able to run an election using either the plurality algorithm or the single transferable voting algorithm. Officials may or may not have used the system before; a help window will be provided for those who need extra assistance.
- **Developers/Testers** - software developers who help maintain the voting system and may have assisted in its development. Developers will also be able to run an election using either of the two algorithms; however, developers will also be able to run tests on the STV algorithm to ensure proper functionality by turning off the ballot shuffle before running the test election.

2.4 Operating Environment

Using the newest JDK and SDK released by Oracle, Java SE 12, the following platforms and operating systems will have the capabilities to utilize the product:

- Windows:
  - o Windows Server 2019, (64-bit)
  - o Windows Server 2016, (64-bit)
  - o Windows Server 2012 R2, (64-bit)
  - o Windows Server 2012, (64-bit)
  - o Windows 10, (64-bit)
  - o Windows 8.x, (64-bit)
  - o Windows 7 version SP1, (64-bit)

- Linux:
  - o Oracle Linux 7.x, (64-bit)
  - o Oracle Linux 6.x, (64-bit)
  - o Red Hat Enterprise Linux 7.x, (64-bit)
  - o Red Hat Enterprise Linux 6.x, (64-bit)
  - o Ubuntu Linux 18.10, (64-bit)
  - o Ubuntu Linux 18.04 LTS, (64-bit)

- macOS
  - o 10.12+ excluding version 10.15, x64

- Java virtual machines (JVMs):
  - o Must be 64-bit

2.5 Design and Implementation Constraints

At this time there are no corporate, regulatory, or hardware limitations. Voting data will be stored within comma-separated value files, and results will be saved in another file; no database will be required. Security and the secure transport of ballots to the voting system is handled outside of the system. The developers will continue to maintain the software after its release.

2.6 User Documentation

Sufficient documentation will be provided to the users to enable them to run the program successfully. Within the system itself there will be a help screen option to assist new users (or returning users who need a reminder) in learning how to use the voting system. The help screen will include information such as the format of user commands and what the various options mean, as well as examples of user inputs.

2.7 Assumptions and Dependencies

The voting system will use the most recent release of Java (Java SE 12). Any dependencies related to the Java version or to APIs and libraries used in the development of the software itself will have to be taken into consideration.

3. External Interface Requirement

3.1 User Interfaces

The most fundamental user interface is something that is easy to use and approachable. Our group proposes a text-based interface approach. In accordance with the other functional requirements, a tutorial / help screen is necessary for completeness.

Error handling is another thing that the system is required to handle. With a text-based interface, an error message will be printed in the console upon incorrect use of the system.

The text interface will consist of three main prompts to input election information, with the option to get help or quit at each prompt. You can see the order of operations in the system below:

1. Vote Selection -- options to quit, help window
2. Seats -- options to quit, help window
3. Files -- options to quit, help window
4. Confirm information -- options for help window, correct information
5. Run

3.2 Hardware Interfaces

The system will be developed in Java, thus making any computer that is able to compile Java code a compatible runtime environment. A list of acceptable platforms can be found in section 2.4.

3.3 Software Interfaces

Any operating system that has the ability to compile Java code is also required, which means that the Java SDK is a required library for this application.

3.4 Communications Interfaces

An audit file should contain the elections results once the election is complete. File transfer will be done on site; therefore, there is no required file transfer protocol for this system.

4. System Features

4.1 Vote Selection

Name	Vote Selection
ID	USER_01
Description	The system should prompt the user to choose the type of voting process the system should perform. The options are plurality or single transferable voting (STV).
Actors	User
Organizational Benefits	Allows the system to process multiple methods of conducting an election. This extends the programs usability to be used in both a plurality environment (such as the United States) or a STV environment (such as Ireland). This also allows for the comparison of how an election would result if using one or the other.
Frequency of Use	User selects vote choice once per program execution.
Triggers	User starts the program.
Preconditions	System must exist. System is running.

Postconditions	The system understands the correct voting algorithms to use when generating the election results.
Main Course	1. System prompts the user for the type of voting process to use a. 1. Plurality b. 2. Single Transferable Vote (STV) 2. User enters their choice 3. System records the type of election 4. System moves to use case *Inputting number of seats to fill* (4.2)
Alternate Courses	AC1 User enters a different case of the string "Plurality" or "Single Transferable Vote". E.g. "plurality" or "SINGLE TRANSFERABLE VOTE" 1. Program is case-agnostic, and can handle all versions of the respective strings. 2. Return to main course step 3 AC2 User enters choice of "2" 1. System recognizes both the text of the selection or the selection number to indicate the choice 2. Return to main course step 3 AC3 User decides they are not ready to run the program anymore 1. See use case 4.8 "Quit Program" AC4 User selects help option 1. See use case 4.6 "Help Window"
Exceptions	EX1 User gives invalid input that does not correspond to either of the choices 1. System responds telling the user to please make a selection of either 1. Plurality, or 2. Single Transferable Vote (STV) 2. Return to main course step 1

4.2 Inputting number of seats to fill

Name	Input Number of Seats
ID	USER_02

Description	The system should prompt the user to choose the number of seats to fill in the election.
Actors	User
Organizational Benefits	Allows the system to process election results in which there are a varying number of seats to fill. This promotes the program's reusability.
Frequency of Use	Once per program execution.
Triggers	The user has entered a valid vote type from use case 4.1.
Preconditions	System must exist. System is running. The type of vote has been selected.
Postconditions	The system understands the number of seats to fill, which determines the number of winners to consider during the election processing.
Main Course	1. System prompts the number of seats to fill 2. User enters the number of seats 3. System records the number of seats 4. System moves to use case *Inputting file name(s)* (4.3)
Alternate Courses	AC1 User decides they are not ready to run the program anymore 1. See use cas 4.8 "Quit Program" AC2 User selects help option 1. See use case 4.6 "Help Window"
Exceptions	EX1 User gives a non-numerical input 1. System responds telling the user to please input a positive integer number 2. Return to main course step 1

4.3 Inputting file name(s)

Name	Prompt user to enter file name(s)
ID	USER_03
Description	The user will need to enter the file name(s) that contain the ballots in order for the system to have data to run on.

Actors	User
Organizational Benefits	Allows for the system to have the pertinent data to use. The system will be able to correctly tabulate the results and will eliminate the need for manual ballot counting, which is error prone.
Frequency of Use	Once per program execution.
Triggers	User has entered the number of seats to fill to complete use case 4.2
Preconditions	System has to exist. There must be at least one CSV file in the same directory as the systems program files. Use case 4.1 and 4.2 must be completed.
Postconditions	The system will have the data needed to tabulate a winner.
Main Course	1. User is prompted to enter the CSV file name(s) to be ran. 2. User inputs file name(s). 3. System stores file names to be used. 4. Moves to use case 4.5
Alternate Courses	AC1 User decides they are not ready to run the program anymore 1. See use cas 4.8 “Quit Program” AC2 User selects help option 1. See use case 4.6 “Help Window”
Exceptions	EX1 User makes a spelling mistake while inputting one or more files. 1. System informs the user that one or more files inputted were not found. 2. System shows the user which files were found. 3. System asks the user if they want to enter more file names. a. if yes: i. Return to main course 1. b. if no: i. See use case 4.5

4.4 Shuffle switch

Name	Shuffling switch
ID	STV_01

Description	The system should allow for the option of shuffling the ballots or leaving the ballots unshuffled. When the switch is off (unshuffled), it allows for testing and calibration of the STV algorithm. When the switch is on (shuffled), the STV algorithm works properly.
Actors	Developer/tester
Organizational Benefits	The shuffling switch is used for testing the STV algorithm. This feature will ensure the system is calibrated properly, and allows for unit testing.
Frequency of Use	Used multiple times during testing.
Triggers	The developer/tester has indicated via CLI that shuffle should be on/off.
Preconditions	System has to exist.
Postconditions	The system has either turned on or off the functionality of shuffling ballots.
Main Course	1. Actor will initiate the program with the shuffle flag. 2. System will inform the actor the shuffle switch is on. 3. Moves to use case 4.1
Alternate Courses	AC1 Shuffle flag was not set 1. Moves to use case 4.1
Exceptions	EX1 System fails to interpret shuffle flag 1. System informs the user nothing has changed. 2. Moves to use case 4.1

4.5 Run Program

Name	Run Program
ID	USER_04
Description	The user runs the voting system program. The program uses the selected algorithm to find the winners of the election.

Actors	User
Organizational Benefits	Allows election results to be calculated in a simple and accurate manner
Frequency of Use	Once per program execution
Triggers	User completes Use Case 4.7 (“Confirm Options”)
Preconditions	Use Cases 4.1, 4.2, and 4.3 have been successfully completed
Postconditions	Election results have been saved to an audit file on the system.
Main Course	1. The program reads the first line of a ballot file to determine the names and number of candidates. 2. The program runs the ballot files through the selected voting algorithm. 3. The program saves the results of the election into an audit file for record keeping. 4. The program displays a snapshot of the results to the user
Alternate Courses	AC1: User has chosen the STV algorithm. 1. After Main Course step 1, the program reads the ballot files to determine the total number of ballots. 2. Program calculates the Droop quota prior to processing the ballots. 3. Return to Main Course step 3.

4.6 Help Window

Name	Help Window
ID	HELP_01
Description	The system should give users the ability to get assistance on how to use the voting system.
Actors	User
Organizational Benefits	This allows the voting system to be self-sufficient so the voter doesn’t need to ask any one person for assistance with how to use the voting system.

Frequency of Use	Used as many times as required, generally no more than one time per user
Triggers	The user types “help” in the text-interface.
Preconditions	System has to exist. User is either using use case 4.1, 4.2, or 4.3
Postconditions	The user has a better understanding of how to use the program.
Main Course	1. User is at respective use case prompt (4.1, 4.2, 4.3, or 4.7) 2. User types “help” 3. Screen displays tutorial 4. The text-interface asks the user if it wants to view the tutorial again or return to the prompt

4.7 Confirm Options

Name	Confirm Options
ID	USER_05
Description	User confirms their selected options before running the election process
Actors	User
Organizational Benefits	Adds a level of verification for the user, allowing them to confirm the vote type, number of seats, and the files to use.
Frequency of Use	Once per program execution
Triggers	Use case 4.3 is successfully completed
Preconditions	Use cases 4.1, 4.2, and 4.3 must be completed.
Postconditions	Program is ready to run with correct user inputs.
Main Course	1. User is shown a screen with their previous selections for the voting algorithm, number of seats, and file names 2. User is prompted to confirm choices 3. After user confirmation, go to Use Case 4.5

Alternate Courses	AC1 User decides to not confirm the results 1. Return to use case 4.1 to allow the user to re-input options. AC2 User selects help option 1. See use case 4.6 “Help Window” AC3 User enters input inconsistent with available activity options 1. System prompts user to enter valid input 2. Return to Main Course step 2

4.8 Quit Program

Name	Quit Program
ID	USER_06
Description	User elects to quit the program
Actors	User
Organizational Benefits	Before submitting ballots to process the election, if a user decides they do not wish to be running the program yet, they may decide to instead quit the program.
Frequency of Use	Once per program execution
Triggers	User types “quit” while on one of the prompts for use case 4.1, 4.2, or 4.3.
Preconditions	User is at a prompt for one of the following use cases: 4.1, 4.2, or 4.3.
Postconditions	Program is terminated.
Main Course	1. User is at respective use case prompt (4.1, 4.2, or 4.3) 2. User types “quit” 3. Program terminates
Alternate Courses	AC1 User enters a different case of the string “quit”. E.g. “QUIT” or “Quit” 1. Program is case-agnostic, and can handle all versions of the string “quit”. 2. Program terminates

4.9 Start System

Name	Start System
ID	USER_07
Description	User uses the command line interface (CLI) to start the program.
Actors	User
Organizational Benefits	Allows for the start of the program, which is then used to process the election.
Frequency of Use	Once per program execution. This would likely be used multiple times during testing, and most likely once during the election processing (being that the user can quit the program early before running the election process). Once the election process has been run (use case 4.5), this use case should no longer be run.
Triggers	The command line interface receives a command to start the program.
Preconditions	User is on a CLI, in the project's directory.
Postconditions	Program is running.
Main Course	1. On CLI, user enters command: javac VotingSystem.java 2. User enters command: java VotingSystem
Alternate Courses	AC1 The code is already compiled 1. User enters command: java VotingSystem
Exceptions	EX1 File not found error 1. Console will give the user an error stating the file was not found. 2. The user should verify they are in the correct directory and check the file name is spelled correctly. EX2 Could not find or load main class _____ error 1. Console will give the user an error stating it could not find / load the main class for the project specified. 2. The user should verify they are in the correct directory, the java source code has been compiled, and they should check the name of the project being specified is correct.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Election results must be able to be generated in 5 minutes or less, given no more than 100,000 ballots in all files combined. This performance requirement will ensure prompt ballot counting for any election using this system.

5.2 Security Requirements

There are no security issues that the software has to handle. Each individual voting precinct will handle voting security. The CSV files received will have been transmitted via a secure channel.

5.3 Software Quality Attributes

The software will have interoperability with a Windows CSV file. This allows for streamlined, straight forward usage of the software, as there is only one standard file format the system uses.

There will also be functionality to test the STV algorithm. This will allow users to ensure the system is calibrated properly and will also allow for unit testing.

After an election has been tabulated, an audit file with a record of the election will be generated for the user to view. These statistics include how many ballots were counted, the number of seats to fill, which algorithm was used (plurality or STV), candidate percentages of the vote, verified winner(s) and loser(s), and which ballots went to which candidates, as well as (if STV was used) the ballot order before and after the shuffle.

In addition to the audit file produced, there will also be a results screen that is generated for the user. This screen is meant to be a quick snapshot of the election results. Some information included will be the type of election, number of total ballots, number of seats to fill, number of candidates. It will also include the winner(s) and loser(s) and pertinent statistical information such as the percentage of votes received (if plurality was used), or order of winning/losing (if STV was used).

5.4 Business Rules

Since the system does not have any security constraints, the system can assume only the correct, authorized election officials have access to the system.

6. Other Requirements

Appendix A: Glossary

CSV File - Format of a tabular data file. It uses commas to delineate between values. Our system will be using the Windows CSV format.

Text-based Interface - Interface that is used within the console of a program. If running the program in an integrated development environment (IDE), this console will likely be within the IDE application. Any interaction with the interface will be through text prompts and keyboard inputs.

Plurality Voting - Given x empty seat(s) to fill, the x candidate(s) with the highest votes win the seat(s). Given a tie, the winner will be chosen randomly. Each ballot in a plurality election contains a single vote for one candidate.

Single Transferable Vote - Shuffles the ballots, calculates the Droop quota using the formula floor(numBallots/(numSeats+1)) + 1, then distributes the ballots one at a time to each candidate. Whenever a candidate reaches the Droop quota, they are marked as a winner and they and their ballots are removed from the election. When all of the ballots have been distributed, the candidate with the lowest number of ballots is marked as a loser and removed from the election; their ballots are then redistributed to the other candidates. This cycle continues until every candidate has been marked a winner or a loser. If not enough candidates reach the quota, losers will be re-marked as winners, starting with the person who lost most recently.