Software Requirements Specification

for

Project 1 - Voting System

Version 1.0 approved

Prepared by Team 14
Liam O'Neil (oneil779), Bilal Osman (osman353),
Hyehwan Ryu (ryu00032), Soorya Sundravel (sundr030)

CSCI 5801

16 February 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Liam O'Neil	1/30	Created Document	0.10

Bilal Osman	2/5	Wrote Overall Description	0.20.1
Liam O'Neil	2/8	Wrote System Features	0.21
Soorya Sundravel	2/11	Wrote Nonfunctional Requirements	0.22.1
Hyehwan Ryu	2/11	Wrote External Interface Requirements	0.22.2
Bilal Osman	2/15	Wrote Introduction	0.23.1
Soorya Sundravel	2/15	Wrote additional information and Glossary	0.23.2

1. Introduction

1.1 Purpose

The purpose of this document is to show users a detailed description of the features and use cases of the voting system. The document describes the functionality, the different voting algorithms and calculations, and the constraints limiting the use of the software. It will also go over the system's interface as well as the accessibility of the software. This document is intended for officials who want to run elections from voting data, as well as developers and testers who want to learn about the workings of the program.

1.2 Document Conventions

This document was created from the IEEE template for system requirements specification documents provided.

1.3 Intended Audience and Reading Suggestions

- Programmers can use the listed formal specifications and software constraints to develop their own voting program. They can implement new features as they desire.
- Testers may use this document to test limitations and performance of the system mentioned later in this document.
- Election officials will use this document to learn all functionality of the voting systems and requirements to run the product successfully. They can also use this document to identify constraints of data collection and calculations.

1.4 Product Scope

The voting system developed is intended to be used to congregate the results of an election. The system is run once the user provides documentation on election information, such as the voting method being used and the tallies of ballots. The program is capable of calculating results of two types of methods: closed party-list voting and instant runoff voting. These two methods will handle other information that is inputted through a list of ballots and parse the data to output election results to an audit file. The audit file will contain calculations and collected external data of the ballots and display candidates who have various ranges of vote percentages. The product is intended for elected officials to handle large collections of ballots and election information and produce wanted results. This software will only be used for election-related purposes and aimed to reduce complexity in ballot counting and evaluation.

1.5 References

- Information regarding closed party-list and instant runoff voting:

http://FairVote.org

- CSE lab machines information:

  https://cse.umn.edu/cseit/classrooms-labs

- IEEE Template for System Requirement Specification Documents:

  https://goo.gl/nsUFwy

2. Overall Description

2.1 Product Perspective

The aim of the voting system is to provide election officials with a way to determine the elected candidate through a list of ballots. It was developed to simulate two types of voting methods: closed party list and instant runoff voting. The list of ballots will be provided as a CSV file. The actual voting is done separately from thevoting program. Anyone interested in the software such as programmers and testers, can use the documentation to add on new features and capabilities.. This also makes it a self-contained product aimed at providing one service, which is simulating how ballots are counted through specific voting methods. This system is run on CSE lab machines through a command prompt.

2.2 Product Functions

The system will consist of three main functions. These are file parsing and determining the winning candidates through the two voting systems. The end result of any voting method is to produce an XML-based text file containing information of the outcome of the election simulation.

File-Parsing

- Input: File is passed in as argument. The user must provide a file.
- Format: Only in CSV format.
- Extracting: Information parsed from file includes
  - Type of voting system: Either IR for instant runoff voting or CPL for closed-party-list voting. Always on the first line.
  - Number of Candidates / Number of Parties Running: separated by commas
  - Candidates / Political Parties: separated by commas
  - Number of Ballots
  - Ballots cast by voters
  - Number of Seats - only in CPL voting
- Organizing: Information will be collected in various ways depending on the voting system.
- Additional Information: More information may be gathered if the user inputs additional information in the command line.

- Generate elected candidate(s): Additional functions are called to simulate a specific voting system based on what was provided in the CSV file.

Instant Runoff-Voting System

- Collecting Information: Information is passed in through as argument and spread through:
    - Data containing candidates with number of votes who chose them as first preference
    - Number of voters
    - Winning candidate
    - Losing candidate per round
- Producing audit file: contains information of all events and outcomes of election
- Tie-Breaker: Determines a loser if two or more candidates have the same number of ballots counted for them
- Determine winner: finds if candidate has more than half of ballots in support of it
- Determine losing candidate: finds losing candidate if there is not a candidate with more than half support from ballots.
- Redistribute Ballots: Moving ballots of those who chose the losing candidate to their next preferred candidate.

Closed Party-List Voting

- Collecting Information: Information is passed in through as argument and spread through:
    - Data containing political parties with number of votes who voted for the party
    - Number of voters
    - Winning political party
    - Number of seats
    - Data containing candidates in list order for each political party
    - Data of available seats and those who will occupy them
- Producing audit file: contains information of all events and outcomes of election
- Allocate Seats: Determines which candidate will sit for each seat available
- Calculating Percentages: Using “largest remainder formula” to find the amount of seats that a political party will take

Overview

- Audit file: distributed at the end of the election simulation. XML-based text file.
- Command-Line: Where users can input the file name of the ballot and election information, as well as additional information.

2.3 User Classes and Characteristics

- Programmers can run and use this program to determine functionality. As well as implementing additional features to improve simulation performance or improve accuracy of results. Can also be used for additional analysis of ballots cast.
- Testers may use this program to identify bugs and unresolved issues before distribution.

- Election officials will be the main users of this software product. Mainly running and using this program to collect results inputted by CSV file containing ballots of a current or previous election.
- Media Personnel can use and run the program to record results. Specifically for public viewing and analysis.

2.4 Operating Environment

The operating environment for the program will be on linux running on CSE lab machines distributed by the University of Minnesota CSE labs. The program will only be run through this operating environment.

2.5 Design and Implementation Constraints

The program will be written in Java utilizing no other external modules or APIs. All source files and class files will be provided. Will utilize Object-Oriented-development with some polymorphism. Program will be run through the command prompt and may be ran through Eclipse if wanted.

2.6 User Documentation

The program will be distributed to those who wish to determine election results through a CSV file of ballots already made. This is not for users who want to collect ballots for voters. That must be done beforehand and separate from this program. Also, this program is aimed to have one specific purpose in simulating a type of voting system, thus anyone who wants to simulate a voting system not supported by the program cannot use it. Only those who wish to use the program's specific purpose can use it to its fullest capability.

2.7 Assumptions and Dependencies

Since the program may only run on CSE lab machines, the user must already have access to the operating environment through a CSE labs account. Also the user must have Java installed as the program can run on the Java 7 version or higher. Finally, in order for the program to be of function, the user must provide a valid CSV file containing the ballots that are cast. Also, the file must not have any numbering or invalid structure as the program parses information in a specific format.

3. External Interface Requirements

3.1 User Interfaces

The user interacts with the software program through the command-line terminal. The program accepts command-line arguments through keyboard input. The system first prompts the user to specify the type of voting system for the election. It then asks the user to provide a file with

the voting data. If the file is unable to open, a message is printed out displaying "Failed to open '*file_name'*. Please provide the correct file." In the end, the system provides the user the audit file with printing " Winner is '*winner's name*' and this file includes the process and result of the election."

3.2 Hardware Interfaces

The machine the program is being run on must have the system resources available to run Java programs and libraries without issues. The program must have access to the machine storage containing the input files, and must have access and permissions to save the audit file to the working directory of the program.

3.3 Software Interfaces

The program is intended to run on x64 Linux platforms, in particular: the University of Minnesota CSE lab desktops. The command line terminal is used to run the voting program. The software requires Java 7 or newer to be installed on the system. A spreadsheet application such as Excel is necessary to view and process CSV files.

3.4 Communications Interfaces

An internet connection is required if accessing the CSE Linux machines through Remote Desktop (VOLE). The software itself runs through the command line. There are no other communications interfaces used by the voting program.

4. System Features

4.1 File parsing

4.1.1 Description and Priority

The system prompts for a CSV file containing ballot information, determines the type of election (instant runoff or closed party list), and collects the data from the file. High priority.

4.1.2 Stimulus/Response Sequences

The system prompts the user for a filename through the terminal.

The user inputs a filename and the system attempts to access the given file.

The system reads the first line of the file to determine the election type.

The system collects the ballot data from the file.

4.1.3 Functional Requirements

REQ-1: Capability to prompt the user for a filename through the terminal.

REQ-2: Access and read the data from the input file.

REQ-3: Produce error messages through the terminal if access to the file is not allowed or the file is unreadable. These messages will be “Access to the file was denied” and “File is unable to be read”, respectively.

REQ-4: Prompt for another filename if an error occurs (restart input process from beginning).

REQ-5: Collect and insert information from the input file into an appropriate data structure (TBD).

4.2 Gather additional information

4.2.1 Description and Priority

The input file may not contain all the information necessary to process the ballot, in which case the user will be prompted for any additional information. High priority.

4.2.2 Stimulus/Response Sequences

The system determines some necessary information is missing or cannot be extracted from the input file.

The system prompts the user to manually input the specific information that is needed.

The system collects the data from the user’s input.

4.3.3 Functional Requirements

REQ-6: Validation that the data extracted from the input file is complete.

REQ-7: Capability to recognize when data is missing and what additional information is necessary.

REQ-8: Prompt for user input indicating the specific data that is necessary. This can be any of the following depending on the election type: election type, number of candidates, candidate names, number of ballots, number of parties, party names, number of seats, individual ballots.

REQ-9: Append input data to the appropriate data structures.

REQ-10: This feature should behave such that all information expected in an input file (4.1) can be retrieved through manual user input.

4.3 Calculate winner of IR election

4.3.1 Description and Priority

For an instant runoff election, the system will count the number one preferences of the voters, check for a majority winner, eliminate the candidate with the fewest votes, and distribute the eliminated candidate’s votes to the voters’ number two preferences. This process is repeated until a candidate holds a majority. High priority.

4.3.2 Stimulus/Response Sequences

The user’s file input or manual input contains a ballot for an instant runoff election.

The system processes the ballot data and calculates the winner.

4.3.3 Functional Requirements

REQ-11: Check the ballot data for the appropriate election type.

REQ-12: Count number one preference votes.

REQ-13: Calculate ratio of votes for each candidate.

REQ-14: Indicate a winner if a candidate has a majority vote.

REQ-15: Determine the candidate with the lowest first preference vote.

REQ-16: Distribute second preference votes to appropriate candidates.

REQ-17: Repeat calculations and eliminations until a winner or tie is determined.

REQ-18: Execute appropriate behavior in the event of a tie or if there is no clear majority (4.5).

REQ-19: Save progression data through each step in the election process for output to the audit file.

4.4 Calculate winner of CPL election

4.4.1 Description and Priority

For a closed party list election, the system will calculate a quota, determine the first allocation of seats, count the remaining votes, determine the second allocation of seats, and indicate the winning candidates. High priority.

4.4.2 Stimulus/Response Sequences

The user's file input or manual input contains a ballot for a closed party list election.

The system processes the ballot data and calculates the winner.

4.4.3 Functional Requirements

REQ-11: Check the ballot data for the appropriate election type.

REQ-20: Calculate the quota by summing the votes and dividing by the number of seats to be filled.

REQ-21: Count and store the votes for each party.

REQ-22: Calculate first allocation of seats.

REQ-23: Remove appropriate number of votes from each party depending on the quota and first allocation of seats.

REQ-24: Calculate second allocation of seats.

REQ-25: Determine winning candidates by number of seats and order of the candidates, and the final seat total for each party.

REQ-19: Save progression data through each step in the election process for output to the audit file.

4.5 Determine the winner of a tie

4.5.1 Description and Priority

An IR election can result in a tie if two or more candidates remain and each receive the same number of votes. This feature remedies a tie and selects a winner. High priority.

4.5.2 Stimulus/Response Sequences

The user's file input or manual input contains a ballot for an instant runoff election.

The ballot is processed as specified in 1.3.

At any point in the process, two or more candidates with the lowest number of votes have the same number of votes.

The system randomly selects the candidate to be eliminated from the election and returns to the ballot processing.

4.5.3 Functional Requirements

REQ-26: Determine which candidates have the same number of votes.

REQ-27: Assign integers values to each candidate.

REQ-28: Randomly select an integer in the appropriate range.

REQ-29: Return to the process of calculating the winning candidate while indicating which candidate is eliminated by the tie-breaker.

REQ-30: Save progression data through each step of determining the winner of a tie for output to the audit file.

4.6 Produce the audit file

4.6.1 Description and Priority

The audit file contains all input election information, the winner(s), and the election progression data. The file is saved to the machine. High priority.

4.6.2 Stimulus/Response Sequences

A user inputs ballot information for an election.

A file object is created for the audit file.

The input information is stored and appended to the audit file.

At each step in the process of calculating a winner, the appropriate data is appended to the audit file to show the election's progression.

The winners of the election are indicated on the audit file.

The audit file is output to the "audits" folder of the program directory.

4.6.3 Functional Requirements

REQ-31: The system creates an entity to store the data of the audit file.

REQ-32: The information provided as input is copied to the audit file entity, including type of voting, number of candidates, candidates, number of ballots, number of parties, and parties depending on the type of election.

REQ-33: Each ballot is assigned to the appropriate candidate or party in the audit file.

REQ-34: Each step in the calculation is appended to the audit file.

REQ-35: The winners of the election are clearly shown in the audit file.

REQ-36: The system saves the audit file entity as a text file in the "audits" folder of the program directory.

4.7 Display results

4.7.1 Description and Priority

The system automatically displays the winners and information about the election to the terminal. High priority.

4.7.2 Stimulus/Response Sequences

The user inputs ballot information for an election.

The system creates a data structure to hold the information that will be displayed.

The data is processed and the type of voting and number of seats are saved by the system.

The election results are calculated.

The system appends the number of ballots cast, the winners, and the number and percentage of votes received by each candidate/party to the data structure.

This information is displayed in the terminal.

4.7.3 Functional Requirements

REQ-37: The system creates an entity to store the information to be displayed OR utilizes the audit file entity created in 1.6 (TBD).

REQ-38: The system displays the number of ballots cast, the winners, the number/percentage of votes for each candidate and party, the type of voting, and the number of seats to the terminal with clearly marked labels.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The program has a minimum runtime constraint of processing 100,000 ballots in under 4 minutes.

5.2 Safety Requirements

There are no special safety considerations that can affect the program to cause any possible loss, damage or harm.

5.3 Security Requirements

There are no special security requirements in terms of authentication or privacy policies.

5.4 Software Quality Attributes

Extensibility of the program would be a beneficial characteristic for developers. In the case that Open Party List had to be implemented, the functionality can be added without modifying the original program. Extensibility ensures that the additional capabilities can be programmed without causing issues to the original code. Object-Oriented-Programming (OOP) enables the creation of classes and methods, which can store and process data. This helps implement new functionality with ease.

Scalability is an important quality of the voting software. Considering the opportunities of future applications in major elections, this program must be able to handle processing information from millions of data entries in an efficient manner. Currently, the minimum runtime constraint is to run 100,000 ballots in under 4 minutes. Scalable software makes sure the program does not take exponential system resources or runtime for processing larger sets of data.

5.5 Business Rules

The election officials have access to the ballots and voting data, and are the users of the program. They can specify the type of voting-system that they want to run, and provide file input of the voting data through comma-separated-value (CSV) files.

6. Other Requirements

Appendix A: Glossary

CSV File - A text file that allows data to be stored in a tabular format. Can be opened and edited through the use of a spreadsheet program. Stands for comma-separated values.

IR - Instant runoff voting or election. A type of election in which voters select candidates in order of preference. First preference votes are tallied and the losing candidate’s second preference votes are distributed to the proper remaining candidates. This process is repeated until a candidate receives a majority of the votes.

CPL - Closed party list voting or election. A type of election in which one or more seats are up for election and voters vote for a specific party. Each party lists their candidates in the order they would be elected should they win seats.

Appendix B: Analysis Models

No pertinent analysis models to include.

Appendix C: To Be Determined List