Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Luisa Jimenez Alarcon, Brian Lu, King Yiu Suen, Scott Deyo

Team 12

February 2021

Table of Contents

Introduction 1

- **Purpose** 1
- **Document Conventions** 1
- **Intended Audience and Reading Suggestions** 1
- **Product Scope** 1
- **References** 2

Overall Description 2

- **Product Perspective** 2
- **Product Functions** 2
- **User Classes and Characteristics** 2
- **Operating Environment** 3
- **Design and Implementation Constraints** 3
- **User Documentation** 3
- **Assumptions and Dependencies** 3

External Interface Requirements 3

- **User Interfaces** 3
- **Hardware Interfaces** 4
- **Software Interfaces** 4
- **Communications Interfaces** 4

System Features 4

- **Read Input Filename** 4
- **Bring File into System** 5
- **Count Votes (Instant Runoff)** 6
- **Count Votes (Open Party Listing)** 7
- **Resolve a Tie** 8
- **Announce Results** 8
- **Produce Audit File** 9
- **Produce Media Report** 10

Other Nonfunctional Requirements 11

- **Performance Requirements** 11
- **Safety Requirements** 11
- **Security Requirements** 11
- **Software Quality Attributes** 11

Business Rules **12**

Other Requirements **12**

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This document is a presentation of the Instant Runoff and Open Party List voting system. A detailed description of the system interface, features and how it will be expected to perform is given. This document will also present how the voting system works and the constraints under it must operate. This document is intended for developers, testers, voting officials, instructors of the class, teaching assistants and fellow classmates.

1.2 Document Conventions

The convention used in this document is based on the IEEE template for Software Requirements Specification Documents. Each requirement has a unique identification number and is also assigned a short title that allows it to be identified in the table of contents followed by a description of the intention of the requirement.

This document uses Arial font and abbreviations are typed in capital letters.

1.3 Intended Audience and Reading Suggestions

This document provides instructions and guidance for programmers, testers, election officials and students who want to learn and use Instant Runoff and/or Open Party List Voting System software.

Programmers	Must read all document; The System Requirements Specification (this document), Use Cases, Requirements Table and other requirements documentation.
Election officials	Should read the System Requirements Specification (this document) and the Use Cases for more in depth knowledge.
Testers	As for programmers.
Students	As for programmers.

1.4 Product Scope

The voting system software supports two types of voting, the Instant Runoff and Open Party List voting system. The system is designed to prompt users to choose a voting system and then for a filename with the ballots to start the voting process. After the counting of the ballots the software will produce an audit file, a report and display the winner. In case of a tie the system will flip a coin.

The voting system is developed in C++ programming language and is expected to run 100,000 ballots in under 8 minutes. The system will assume there are no errors in the ballots and security protocols are not needed.

1.5 References

IEEE Template for System Requirement Specification Documents:
https://goo.gl/nsUFwy

UMN CSE labs Hardware requirements:
https://cse.umn.edu/cseit/classrooms-labs

2. Overall Description

2.1 Product Perspective

This system is developed to count ballots and decide the winner(s) of an election. Ballots are expected to be cast in a different system, and are expected to be stored in a comma separated values (CSV) file. When the system finishes processing the ballots, the winner(s) and other information about the election (such as the type of the election, the number of ballots cast and the number of votes each candidate received), will be displayed on the screen. A text file containing the same information will also be generated, so that the user can forward it to media personnels. To allow for replication of the election, an audit file that shows how each ballot is assigned as the election progresses will be created.

2.2 Product Functions

- Read input filename: Read the filename user inputted via command line interface.
- Bring file to system: Read the input file and the setup of the election according to the pre-specified format.
- Count votes (instant runoff): Count the number of votes for each candidate and decide the winner(s) based on the instant runoff algorithm.
- Count votes (open party listing): Count the number of votes for each candidate and decide the winner(s) based on the open party listing algorithm.
- Resolve a tie: If there is a tie, randomly select a winner based on a fair coin toss.
- Announce results: Display the winner(s) and information about the election to the screen.
- Save audit file: Produce a file that shows how the election progressed so that the audit could replicate the election itself.
- Save media report: Produce a file containing the summary of the election to be shared with media personnel.

2.3 User Classes and Characteristics

There are three types of users that may interact with the system:
- election officials who need to know the results of the election and generate reports to be shared with media personnels,

- programmers who are interested in working on the program by further developing it or fix existing bugs, and;
- testers who want to verify that all the functional and non-functional requirements of the program are satisfied.

2.4 Operating Environment

We recommend systems that meet or exceed the following specifications:

Operating System:	Ubuntu 20.04, Windows 10, or Mac OS X 10.15
Processor (CPU):	Intel Core i5 or equivalent
Memory:	16 GB RAM
Storage:	500 MB available (for storing audit files and media reports)

2.5 Design and Implementation Constraints

The system should be able to process 100,000 ballots in under 8 minutes. There are no special safety or security requirements. Invalid ballots are expected to be handled at the voting centers.

2.6 User Documentation

A separate documentation of how to run tests in the system is provided for testers.

2.7 Assumptions and Dependencies

The system is developed in C++ and therefore requires a C++ compiler, GNU Compiler Collection (GCC), to be installed on the user's computer. It is also assumed that the system is only executed once for each election. All ballots in the input file are assumed to be valid and formatted according to a pre-specified format.

3. External Interface Requirements

3.1 User Interfaces

The user interface of the voting system only consists of the command line. At the command line, the filename of the ballot CSV is passed in as a command line argument. If the filename is valid, then the votes will be processed. If the filename points to a file that is invalid or missing, then the voting system will return an error message telling the user to enter a valid filename.
When the ballots have been counted and processed, the command line displays the results of the election, i.e., type of voting, number of candidates, name and party of candidates, number of seats, number of ballots, number of votes each candidate received, and winner(s).

3.2 Hardware Interfaces

As stated in section 2.4, the minimum hardware requirements of the voting system are an Intel i5 processor or an equivalent processor, 16 gigabytes of RAM, and 500 megabytes of available storage.

3.3 Software Interfaces

As stated in section 2.7, the voting system requires the GNU Compiler Collection (GCC) to be installed on the system for compiling the program.

3.4 Communications Interfaces

The voting system does not require an internet connection, and does not interact with any other services over the internet.

4. System Features

This section demonstrates the voting system's most prominent features and explains how they can be used and the results they will give back to the user.

4.1 Read Input Filename

Name	Read Input Filename
ID	UC_01
Description	The system validates the filename inputted by the user.
Actors	Election Officials, Programmers, Testers
Organizational Benefits	This feature ensures the existence of the input file and its file format is CSV before proceeding.
Frequency of use	As necessary per election.
Triggers	The user enters a filename after the system prompts.
Preconditions	N/A
Postconditions	The filename is stored in the system as a string.
Main Course	1. The system prompts the user to enter a filename. 2. The user inputs the name of the ballot file in the command line. 3. The system reads the input as a string and checks if the file name is valid (see EX1, EX2).

Alternate Courses	N/A
Exceptions	EX1: The input file does not exist. 1. Display an error message concerning the existence of the file and back to Main Course #1. EX2: The file extension is not CSV. 1. Display an error message concerning the file format and back to Main Course #1.

4.2 Bring File into System

Name	Bring File into System
ID	UC_02
Description	The system opens the input file and reads the setup of the election.
Actors	Election Officials, Programmers, Testers
Organizational Benefits	This feature allows the system to know which voting algorithm to run and create the appropriate data structures to store the information about the election.
Frequency of use	As necessary per election.
Triggers	The function is triggered automatically after the system reads the input filename.
Preconditions	The input filename is validated.
Postconditions	The system knows which voting algorithm to run, the number of participated candidates, as well as the names and parties of the candidates.
Main Course	1. The system opens the input file in read mode. 2. The system reads which voting algorithm, IR or OPL, to use to decide the winner(s) from line 1. 3. The system reads the number of candidates who participated in the election from line 2. 4. The system reads the names and the parties of the candidates from line 3. 5. If the voting algorithm is IR, proceed to Count Votes (Instant Runoff) (UC_03). If the voting algorithm is OPL, proceed to Count Votes (Open Party Listing) (UC_04).
Alternate Courses	N/A
Exceptions	N/A

4.3 Count Votes (Instant Runoff)

Name	Count Votes (Instant Runoff)
ID	UC_03
Description	This feature processes the results of an Instant Runoff algorithm and passes its results to either Resolve a Tie (UC_05) or Announce Results (UC_06).
Actors	Election Officials, Programmers, Testers
Organizational Benefits	Necessary software requirement.
Frequency of use	As necessary per election.
Triggers	The function is triggered when the first line of the input file is read as "IR".
Preconditions	N/A
Postconditions	Election results are returned to Announce Results (UC_06) or Resolve a Tie (UC_05).
Main Course	1. The system opens the file in read-mode. 2. A variable is derived from the total number of ballots / 2 + 1. This is the number of votes needed to win the election without needing to break a tie. 3. When the ballot section of the file is reached, a data structure will be made for each ballot, with ballot ID and a rank value for each candidate (0 for no rank given for said candidate). a. It is presumed that each ballot ranks at least one candidate as '1'. b. It is presumed each ballot has no errors. 4. All ballots are queued and rank '1' processed, with each '1' rank incrementing the vote count of the respective candidate. 5. After this pass, if there is a candidate with a majority of votes, that candidate is the winner and the results are returned to Announce Results (UC_06). 6. Else the candidate with the lowest number of votes has all ballots associated via ballot IDs processed again, with the next lower ranked candidate on their ballot getting the vote. If there are no more ranked candidates on the ballot, the ballot may be dequeued. 7. Repeat steps 5-6 until a winner is declared in step 5, or two candidates are tied with 50%, in which case call Resolve a Tie (UC_05) with the tied candidates.
Alternate Courses	Count Votes (Open Party Listing)

Exceptions	None

4.4 Count Votes (Open Party Listing)

Name	Count Votes (Open Party Listing)
ID	UC_04
Description	This feature processes the results of an Open Party Listing algorithm and passes its results to either Resolve a Tie (UC_05) or Announce Results (UC_06).
Actors	Election Officials, Programmers, Testers
Organizational Benefits	Necessary software requirement.
Frequency of use	As necessary per election.
Triggers	The function is triggered when the first line of the input file is read as "OPL".
Preconditions	N/A
Postconditions	Election results are returned to Announce Results (UC_06) or Resolve a Tie (UC_05).
Main Course	1. The system opens the file in read-mode. 2. The total number of ballots and total number of seats are read from the file. A variable, 'quota', is derived by dividing the number of ballots by the number of seats. 3. When the ballot section is reached, a data structure is made with the Ballot ID, the party voted for, and the candidate voted for. 4. Ballots are processed: The relevant candidate will have their vote count incremented for each ballot cast for them. a. It is presumed that each ballot has a vote for a single candidate. b. It is presumed each ballot has no errors. 5. Each party is allocated an initial number of seats equal to the floor function of (number of votes for the party / quota). The remainder from the floor function is saved as a variable for each party. 6. If there are seats left over, they are assigned to the party or parties with the highest remainders. 7. If there is a tie, it is sent to Resolve a Tie (UC_05) for resolution. 8. Once the number of seats is determined for each party, the candidates with the most votes in a party will win the seats. In the case of a tie(s), Resolve a Tie (UC_05) is called.

	9. The results are returned to Announce Results (UC_06).
Alternate Courses	Count Votes (Instant Runoff)
Exceptions	None

4.5 Resolve a Tie

Name	Resolve a tie
ID	UC_05
Description	If there is a tie between candidates a fair coin toss is used to randomly select a winner.
Actors	Election Officials, Programmers, Testers
Organizational Benefits	Allow the voting system to choose a winner in case of a tie.
Frequency of use	As necessary per election.
Triggers	There is a tie of votes between candidates.
Preconditions	There must be at least two candidates with the same amount of votes.
Postconditions	A winner is chosen.
Main Course	1. The system tells the user that there has been a tie. 2. The system prompts the user to start the toss of a coin. 3. A coin is tossed. 4. A winner is displayed.
Alternate Courses	None.
Exceptions	None.

4.6 Announce Results

Name	Announce Results
ID	UC_06
Description	After the Voting system algorithm runs the system will display the winner on screen.
Actors	Election Officials, Programmers, Testers

Organizational Benefits	Allow the voting system to clearly display the winner of the election to the actors.
Frequency of use	Once
Triggers	Algorithms have finished counting the ballots.
Preconditions	The system vote algorithm must be completed.
Postconditions	A summary of results and a winner is displayed.
Main Course	1. A Voting System algorithm has started counting the ballots. 2. Results summary and winner is displayed on the terminal.
Alternate Courses	ALT1: There is a tie between candidates. 1. See use case UC_05
Exceptions	None.

4.7 Produce Audit File

Name	Produce audit file.
ID	UC_07
Description	Writes and saves an audit file under a unique name, which will contain information about the election. Specifically, the audit file will contain the type of voting, number of candidates, names and parties of candidates, number of seats, number of ballots, number of votes each candidate received, winner(s), coin flips, calculations, and how the election progressed (i.e. which candidate got which ballot, and when). The audit file is written concurrently with counting and processing the ballots.
Actors	Testers.
Organizational Benefits	The audit file is written so that it can competently replicate the election.
Frequency of Use	Whenever ballots are counted and processed, only one audit file is produced.
Triggers	The ballot file is brought into the system.
Preconditions	The ballot file is valid, i.e., there is no information missing from the file, nor are there any mistakes.
Postconditions	There is an audit log with a unique filename in the assumed directory.

Main Course	1. The ballot file is brought into the system. 2. The system creates an audit file in the assumed directory with a timestamped filename. (See EX1) 3. The system identifies the type of voting, number of candidates, names and parties of candidates, number of seats, and number of ballots from the ballot file. The system writes this information to the audit file. 4. As the ballots are counted and processed, the system writes every vote counted, candidate eliminated, vote transferred, seat allocated, coin flip, and calculation to the audit file. 5. After the results of the election have been announced, the system writes the number of votes each candidate received and the winner(s) to the audit file.
Alternate Courses	None.
Exceptions	EX1: There already exists a filename with the desired timestamp in the assumed directory. 1. The system creates an audit file in the assumed directory with a timestamped filename, followed by an incrementing numeral, which increments until there is an available filename. Continue to Main Course #3.

4.8 Produce Media Report

Name	Produce media report.
ID	UC_08
Description	Writes and saves and a media report under a unique name, which will contain basic information about the election. Specifically, the media report will contain the same information as what is displayed on the screen when the results of the election are announced: the type of voting, number of candidates, names and parties of candidates, number of seats, number of ballots, number of votes each candidate received, and winner(s). The media report is written after the results of the election have been announced.
Actors	Testers.
Organizational Benefits	The media report is meant to be a summary of the election, not containing the fine details of how the votes are counted.
Frequency of Use	Whenever ballots are counted and processed, only one media report is produced.
Triggers	The results of the election are announced.
Preconditions	None.

Postconditions	There is a media report with a unique filename in the assumed directory.
Main Course	1. The results of the election are announced. 2. The system creates a media report in the assumed directory with a timestamped filename. (See EX1) 3. The system writes the type of voting, number of candidates, names and parties of candidates, number of seats, number of ballots, number of votes each candidate received, and winner(s) to the audit file.
Alternate Courses	None.
Exceptions	EX1: There already exists a filename with the desired timestamp in the assumed directory. 1. The system creates a media report in the assumed directory with a timestamped filename, followed by an incrementing numeral, which increments until there is an available filename. Continue to Main Course #3.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Voting System requires a system with at least a 16GB of RAM and 500MB of memory in order to process a small number of ballots without an unexpectedly long delay. Performance depends on the number of ballots to be processed and as such, the system requirements will need to scale accordingly for larger numbers of ballots.

5.2 Safety Requirements

Since the file containing ballot information is opened in read-only mode, and not modified, there is no danger of losing ballot data by running the Voting System.

5.3 Security Requirements

Since voter information is not kept in the file(s) used by the Voting System, there are no security concerns in the voting system.

5.4 Software Quality Attributes

The Voting System requires basic training in order to use it, or at least familiarity with using command line arguments.
The Voting System is intended to be a low-level tool, and as such does not have concerns with adaptability, reusability, testability, or other attributes or a higher-level tool.

5.5 Business Rules

The Voting System software has no business rules.

6. Other Requirements

There are no other requirements.

Appendix A: Glossary

Term	**Definition**
CSV	Comma-separated values file.
Election officials	An official responsible for the proper and orderly voting at polling stations.
GCC	GNU Compiler Collection.
IR	Instant Runoff.
N/A	Not applicable.
OPL	Open Party List.

Appendix B: Analysis Models

None.

Appendix C: To Be Determined List

None.