Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Chenxuan Liu,

Jicheng Zhu,

Yingwen Weng,

Zilong He.

University of Minnesota

February 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this document is to introduce our voting software. Our software is developed to count the ballots of two types of voting: 1. Instant runoff voting; 2. Party list voting. It will also include features of the software, the interfaces of the software, and the constraints under which it must operate. This document is intended for users of the software and also potential developers.

1.2 Document Conventions

This Document is created based on the IEEE template for System Requirement Specification Documents.

1.3 Intended Audience and Reading Suggestions

User:

- Typical Users, such as election officials at voting stations, who want to use our voting systems for analyzing results of two types of voting (Instant Runoff Voting, IRV & Open Party List, OPL)
- The election result will be shared with media
- Programmers and testers who are interested in working on the project by further developing it to be used for more types of voting or fix existing bugs.

Overview:
In section 1, we give a basic introduction of our software. Section 2 is an overall description including perspectives, functions of our software, user classes and characteristics, constraints of software and hardware, user document and assumptions & dependencies. We introduce the interface of our software in section 3. System features are discussed in section 4. In the end, we go over other nonfunctional requirements.

1.4 Product Scope

This software is a tool developed to get the election results from two voting methods (IRV & OPL). It will be run multiple times during the time for both normal and special elections. An audit file will be generated and shared to the media. For IR, the audit file will include all steps showing how the ballots are redistributed. The software will be run on the most up to date version of CSE lab machines.

1.5 References

Software's Github:
https://github.umn.edu/umn-csci-5801-S21-002/repo-Team11.git

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

Instant-runoff voting, Wikipedia:
https://en.wikipedia.org/wiki/Instant-runoff_voting?oldformat=true

Party-list proportional voting, Wikipedia:
https://en.wikipedia.org/wiki/Party-list_proportional_representation?oldformat=true

GUI, Wikipedia:
https://en.wikipedia.org/wiki/Graphical_user_interface?oldformat=true

RAM, Wikipedia:
https://en.wikipedia.org/wiki/Computer_memory?oldformat=true

CSV, Wikipedia:
https://en.wikipedia.org/wiki/Comma-separated_values?oldformat=true

API, Wikipedia:
https://en.wikipedia.org/wiki/API

2. Overall Description

2.1 Product Perspective

This voting system is developed for everyone who needs a program to summarize plenty of voting data and find the results. It can handle only one kind of data format which is a CSV file and supports two kinds of voting algorithms (Instant Runoff Ballot and Open Party List Ballot). It is an open-source project and it has a very active developer team to support it and provide feedback to users. It was developed to run on Windows, Mac OS X, and Linux.

2.2 Product Functions

- Read: read the input file.
- Decision: determine the vote type.
- Process 1: File processing for Instant Runoff.
- Process 2: File processing for Open Party Runoff.
- Two-side Decision: Coin Flip simulation.
- Display: Display the results on the screen for media.
- Audit: Generalize the auditing file.

More specific information is listed in section 4.1 use cases.

2.3 User Classes and Characteristics

Refer to Section 1.3

- Typical Users, such as election officials at voting stations, who want to use our voting systems for analyzing results of two types of voting (Instant Runoff Voting, IRV & Open Party List, OPL)
- The election result will be shared with media
- Programmers and testers who are interested in working on the project by further developing it to be used for more types of voting or fix existing bugs.

2.4 Operating Environment

OS:
Windows 10
Ubuntu 16.04, 18.04
macOS Mojave (10.14) or up

Hardware:
Memory: 256 MB or higher
Graphics Card: Not required
CPU: Intel Pentium 4 2.00GHz or better
File Size: 1 MB

Software:
Java JRE

2.5 Design and Implementation Constraints

Our voting system is developed in Java. It uses a modular design where every feature is wrapped into a separate module and the modules depend on each other through well-written APIs. There are several APIs available to make plugin development easy. We don't build a visualization interface for our system.

2.6 User Documentation

We don't have a user documentation for this voting system right now.

2.7 Assumptions and Dependencies

The voting system is developed in Java and therefore requires Java to be installed on the user's system. The latest stable version of the voting system requires Java version 11 or higher. This applies to Windows and Linux users. On Mac OS X, Java is bundled with the application.

3. External Interface Requirements

3.1 User Interfaces

The voting system does not provide a graphical user interface, instead, it can be run through command-line tools.
Open the command prompt (Windows), or terminal (macOS, Ubuntu).
Run the command **java votesystem filename**, no further interaction is required.
The voting system will then process the file and print the result on the screen.
The voting system will also create an audit file named votingformat_date_time.txt (e.g., OPL_2020-02-13_13:02.txt)

3.1.1 Instant Runoff Ballot

IR audit file

```
Type of Voting: Instant Runoff (IR)
Number of Candidates: 4
Candidates:
Rosen (D), Kleinberg (R), Chou (I), Royce (L)

First count:
Start ballot distribution,
Ballot 1: Rosen
Ballot 2: Rosen
Ballot 3: Rosen
Ballot 4: Chou
Ballot 5: Chou
Ballot 6: Royce

Rosen (D): 3
Kleinberg (R): 0
Chou (I): 2
Royce (L): 1

No majority
Start Transfer of Royce's Votes,
Ballot 6 does not have other number -> Discord

Second count:
Rosen (D): 3
Kleinberg (R): 0
Chou (I): 2

Rosen is the majority.
```

3.1.2 Open Party List Ballot

OPL audit file

Type of Voting: Open Party List Ballot(OPL)
Number of Candidates: 21
Candidates:
Democratic: Benjamin Pike, Sam Rosen-Amy, Megan Gentzler, Ben Foster, Colin Volz,
Republican: Fram Deutsch, Steve Grolnic, Wendy Berg, Gerald Epstein, Sarah McChurg,
Reform: Steven Wong, Dchorah Gorlin, Brad Crenshaw, Daniel Czitrom, Meryl Fingrutd,
Green: Tom Wartenberg, Juan Hernandez, Beata Panagopoules, Alice Morey, Sarah Pringle,
Independent Candidate: Robert Moll
ballot 1: Benjamin Pike
ballot 2: Sam Rosen-Amy
…
ballot 100000: Robert Moll
Number of Ballots:
Republican: 38000
Democratic: 23000
Reform: 21000
Green: 12000
Moll: 6000
Calculation and Steps:
1. The quota:
The total number of valid votes/the number of seats
100000/10=10000
2. Every 10,000 votes represent a seat for a party.
Counting the number of votes for every party
First Allocation Of seats:
Republican: 3
Democratic: 2
Reform: 2
Green: 1
Moll: 0
3. The remaining votes and the remaining seats
Republican: 38000-3*10000=8,000
Democratic: 23000-2*10000=3,000
Reform: 21000-2*10000=1,000
Green: 12000-1*10000=2,000
Moll: 6000-0*10000=6,000
Remaining seats: 10-(3+2+2+1+0)=2
4. Compare the remaining votes and find remaining seats to be allocated
Republican: 1
Democratic: 0
Reform: 0
Green: 0
Moll: 1
5. List the result:

Party Name	Seats	% of vote	% of seat
Republican:	4	38%	40%
Democratic:	2	23%	20%
Reform:	2	21%	20%
Green:	1	12%	10%

Moll:	1	6%	10%

3.1.3 screen output

IR

```
Counting is over!
Type: IR
Number of ballot: 100000
Klicinberg (R): 52000
Rosen (D): 18000
Chou (I): 30000
The winning elector :
Klcinberg (R)
Thank you for using the vote counting system!
```

OPL

```
Counting is over!
Type: OPL
Number of ballot: 100
Number of seats: 4
Parties:
D: 50, 50%, 2 seats
R: 31, 31%, 1 seats
I: 19, 19%, 1 seats
Candidates:
Pike (D): 30
Foster (D): 20
Deutsch (R): 10
Borg (R): 20
Jones (R): 1
Smith (I): 19
The winning electors :
Pike (D)
Foster (D)
Borg (R)
Smith (I)
Thank you for using the vote counting system!
```

3.1.4 CSV file

Example CSV file format for IR:

```
IR
```

```
4
Rosen (D), Kleinberg (R), Chou (I), Royce (L)
6
1,3,4,2
1,,2,
1,2,3,
3,2,1,4
,,1,2
,,,1
```

1st Line: IR if instant runoff
2nd Line: Number of Candidates
3rd Line: The candidates separated by commas
4th Line: Number of ballots in the file
Each ballot will have at least 1 ranking

Example CSV file format for OPL:

```
OPL
6
[Pike,D], [Foster,D],[Deutsch,R], [Borg,R], [Jones,R],[Smith,I]
3
9
1,,,,,
1,,,,,
,1,,,,
,,,,1,
,,,,,1
,,,1,,
,,,1,,
1,,,,,
,1,,,,
```

1st Line: OPL for open party listing
2nd Line: Number of Candidates
3rd Line: The candidates and their party in [ ]. Notice the name and party are separated by commas.
4th Line: Number of Seats
5th Line: Number of Ballots
Only a single 1 will be placed in the position of the given candidate selected.

3.2 Hardware Interfaces

The minimum hardware requirements of this voting system are an Intel Pentium 2.0 GHz CPU and 256 MB of RAM. No special requirement for the graphics card.

3.3 Software Interfaces

This voting system requires Java to be installed, Java version 11 or newer is recommended to run the program without errors.

This voting system can be used on Windows 10, Ubuntu 16.04, Ubuntu 18.04, and macOS Mojave. Additional information about the operating environment can be found in section 2.4 of this document.

Data is imported from a CSV file and an audit file containing the result is generated after processing. No data is shared across other software.

3.4 Communications Interfaces

The voting system can finish counting and presenting the result locally. No communications functions are required in the system.

4. System Features

4.1 Use Cases

4.1.1 Read the Input File

Name	Read the Input File
ID	VT_001
Description	Read a CSV file that contains the voting information
Actor(s)	User
Organization Benefits	After the voting system reads the file, it can acquire all the necessary information from a single file. It is more efficient and accurate than type ballots into the system manually.
Frequency of Use	Each time we want to get the vote result from a CSV file.
Trigger	The user starts the voting system using command-line tools.
Precondition	A valid CSV file is stored in the same directory as the voting system.

Postcondition	The voting system successfully read the file and was able to start processing.
Main Course	1, User starts the voting system, give the file name as its argument 2, System searches the file from the current working directory (see EX1, EX2) 3, System confirms that the file is successfully read.
Alternate Courses	Ac1 user start the voting system without giving file name 1, Prompt user to type the file name 2, Return to the main course step 2
Exceptions	Ex1 system cannot find the file 1, Notify the user that the file doesn't exist 2, Prompt user to type the file name again 3, Return to the main course step 2 Ex2 permission denied when reading the file 1, Notify the user that he doesn't have permission to read the file 2, Terminate the voting system

4.1.2 Determine the Vote Type

Name	Determine the Vote Type
ID	VT_002
Description	Determine the type of the vote after we read the CSV file
Actor(s)	None
Organization Benefits	Only after we determine the type of vote (IR or OPL), we can process the file with its respective algorithm. It ensures the efficiency and accuracy of our voting system.
Frequency of Use	Every time we read a CSV file, we need to know the type of the vote before we determine the algorithm to further process the file.
Trigger	Occurs automatically after we successfully read the file (see use case VT_001).
Precondition	The file is already loaded
Postcondition	The type of the vote is decided

Main Course	1, Get the first line of the file 2, If the first line is IR, go to file processing for IR (see use case VT_003) If the first line is OPL, go to file processing for OPL (see use case VT_004) Otherwise, go to Ex1
Alternate Courses	None
Exceptions	Ex1 the type is neither IR nor OPL 1, Notify the user that the system only supports IR or OPL 2, Terminate the voting system

4.1.3 File Processing For IR

Name	File Processing for IR
ID	VT_003
Description	Parse the CSV file, count ballots, and calculate the winner based on IR
Actor(s)	None
Organization Benefits	This is the main function in the voting system to calculate the winner for IR. A reliable implementation ensures we can count the vote efficiently and get accurate results. This system can also significantly reduce errors compared to counting the vote manually.
Frequency of Use	Runs every time we determine the type of vote is IR
Trigger	Occurs automatically after we successfully determine the vote type (see use case VT_002), and the type is IR.
Precondition	After reading the file, the type of vote is IR
Postcondition	The count is finished, and the system calculates the winner
Main Course	1, Get the vote information including the number and name of candidates, the number of ballots 2, Scan all the following lines contain ballots, count the first candidates for every vote

	3, If there exists a tie, go to coin flipping simulation (see use case VT_005) 4, If there is no majority, discard the candidate with the least number of ballots and redistribute his/her ballots to the next choice. Go to step 3. 5, If there is a majority, end the counting and go to use case VT_006 (Ac1)
Alternate Courses	Ac1 No majority until there is only one candidate 1, The popularity wins and go to use case VT_006
Exceptions	Ex1 the number of the vote doesn't match the line number 1, Notify the user that the file is not complete 2, Terminate the voting system

4.1.4 File Processing For OPL

Name	File Processing for OPL
ID	VT_004
Description	Parse the CSV file, count ballots, and calculate the winner based on OPL
Actor(s)	None
Organization Benefits	This is the main function in the voting system to calculate the winner for OPL. A reliable implementation ensures we can count the vote efficiently and get accurate results. This system can also significantly reduce errors compared to counting the vote manually.
Frequency of Use	Runs every time we determine the type of vote is OPL
Trigger	Occurs automatically after we successfully determine the vote type (see use case VT_002), and the type is OPL
Precondition	After reading the file, the type of vote is OPL
Postcondition	The count is finished, and the system calculates the winner
Main Course	1, Get the vote information including the number, name and party of candidates, the number of ballots. 2, Scan all the following lines contain ballots, count the vote for both candidate and party 3, Calculate the number of seats for each party (see Ex1)

	4, Give remaining seats to each party based on the remaining votes 5, go to use case VT_006
Alternate Courses	Ac1 there is a tie when allocating remaining seats 1, Go to the coin flipping simulation to decide the winner(see VT_005) 2, Back to main course step 4 to continue allocating
Exceptions	Ex1 the number of seats exceeds the number of people in one party 1, Set spare seats as empty seats 2, Back to main course step 4

4.1.5 Coin Flipping Simulation

Name	Coin flipping Simulation
ID	VT_005
Description	When we meet a tie in the file processing, flip a coin to determine the winner.
Actor(s)	None
Organization Benefits	A simple and fair way to determine the winner whenever there is a tie. No need to run the election again to save time and money.
Frequency of Use	Rarely used, only flip the coin when there is a tie.
Trigger	Automatically triggered when we get a tie in the file processing (see VT_003 and VT_004).
Precondition	A tie occurs
Postcondition	The winner is determined by the coin flipping
Main Course	1, Get the number of the candidates (n) who have the same number of votes from file processing (see VT_003 and VT_004). 2, Generate a random number (x) from 1 to n 3, The x^{th} candidate becomes the winner
Alternate Courses	None

Exceptions	None

4.1.6 Display the Result To the Screen

Name	Display the Result to the Screen
ID	VT_006
Description	The voting system displays the result on the screen
Actor(s)	None
Organization Benefits	Brief information about the vote is displayed on the screen (e.g., the winner, type of election), it provides us an efficient way to know the result without looking at the detailed audit file.
Frequency of Use	Every time we successfully processed the file, the voting system would display the result.
Trigger	Automatically triggered after the voting system finishes file processing (see VT_003 and VT_004).
Precondition	File processing (see VT_003 and VT_004) is finished
Postcondition	Result is displayed on the screen
Main Course	1, Print out the vote information including the winner(s), type of election, and the number of seats. 2, Exit the voting system
Alternate Courses	None
Exceptions	None

4.1.7 Generate the Audit File

Name	Generate the Audit File
ID	VT_007
Description	The voting system writes the result into an audit file
Actor(s)	None

Organization Benefits	A detailed audit file is generated so we can verify the voting result through other systems. It ensures that the whole voting process is transparent and fair.
Frequency of Use	Every time we successfully processed the file, the voting system would display the result.
Trigger	Automatically triggered after the voting system finishes file processing (see VT_003 and VT_004).
Precondition	File processing (see VT_003 and VT_004) is finished
Postcondition	An audit file is generated
Main Course	1, Create an audit file named votingformat_date_time.txt (e.g., OPL_2020-02-13_13:02.txt) in the current working directory. (see Ex1) 2, Write the type of voting, number of candidates, candidates, number of ballots into the audit file. (see Ex2) 3, When processing the file (see VT_003 and VT_004), record every step into the file. (see Ex2) 4, After the file finishes processing, write the winner(s), and how many votes a candidate has into the file. (see Ex2)
Alternate Courses	None
Exceptions	Ex1 failed to create the new file 1, Notify the user that the system cannot create the audit file 2, Terminate the voting system Ex2 failed to write into the file 1, Notify the user that the system cannot write into the audit file 2, Terminate the voting system

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The voting system needs an operation system with at least a 2.0 GHz CPU and 256 megabytes of RAM to meet optimal running efficiency. Best use CSELabs machines running the system. A Java

compiler is also required, Java version 11 or higher is recommended to run the program without errors.

Users need to provide documentation of the information they need to run, the file needs to be a CSV file with a comma separated values and each row is separated by a newline. The first line of the file needs to be the type of voting (IR or OPL).

In addition, the voting system using text provides interaction with the user, thus requiring the user to have a screen on which text can be printed.

5.2 Safety Requirements

There are no specific safety requirements. However, to ensure that users will not lose any data while using the voting system, we encourage users to duplicate the information file before running it in the system. In addition, the developer will update the voting system to fix bugs. Users are also encouraged to report bugs to the developer.

5.3 Security Requirements

No specific security requirements. Ensuring one vote for one person is handled at the voting centers.

5.4 Software Quality Attributes

The voting system provides users with adaptable, correct, simple, reliable, interoperable, reusable, testable and usable features.
The voting system can be used on Linux (Ubuntu 16.04, 18.04), Mac (macOS Mojave (10.14) or up) and Windows (Win10) systems.
Each poll result file can be run any number of times at any time and has the same correct output.
Any output is reliable and can withstand manual verification.
The system is friendly to use, simple and intuitive to operate, and has good interaction effects.

5.5 Business Rules

An election counting person is allowed to enter the system, upload the information file and let the system count the votes.

An audit file is generated by the voting system, and it can be examined by other auditing officers.

6. Other Requirements

None.

Appendix A: Glossary

IRV: Instant-runoff voting, also sometimes referred to as the alternative vote (AV), preferential voting, or ranked-choice voting (RCV), though these names are also used for other systems, is a type of ranked preferential voting counting method used in single-seat elections with more than two candidates. Instead of indicating support for only one candidate, voters in IRV elections can rank the candidates in order of preference. Ballots are initially counted for each voter's top choice.

OPL: Party-list proportional representation systems are a family of voting systems emphasizing proportional representation in elections in which multiple candidates are elected (e.g., elections to parliament) through allocations to an electoral list. They can also be used as part of mixed additional member systems. In this document, we specifically implement an open party list voting.

GUI: A graphical user interface is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

RAM: is a form of computer memory that can be read and changed in any order, typically used to store working data and machine code.

CSV file: A CSV is a comma-separated values file, which allows data to be saved in a tabular format.

APIs: An application programming interface (API), is a computing interface that defines interactions between multiple software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc. It can also provide extension mechanisms so that users can extend existing functionality in various ways and to varying degrees. An API can be entirely custom, specific to a component, or designed based on an industry-standard to ensure interoperability. Through information hiding, APIs enable modular programming, allowing users to use the interface independently of the implementation.

Appendix B: Analysis Models

None

Appendix C: To Be Determined List

None

Appendix D: Division of Work

Team member	Division of work

Jicheng Zhu(zhu0024)	**section3, section 4**
Zilong He (he000229)	**section1, IR example audit file**
Chenxuan Liu(liu00464)	**section2, OPL example audit file**
Yingwen Weng(weng0069)	**section5, Screen output example**