**Software Requirements Specification**

**for**

**Vote Counting System**

**Version 1.0 approved**

**Prepared by Team 9**

**University of Minnesota**

**02/15/2021**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 3

2.6 User Documentation 3

2.7 Assumptions and Dependencies 4

**3. External Interface Requirements 4**

3.1 User Interfaces 4

3.2 Hardware Interfaces 4

3.3 Software Interfaces 4

3.4 Communications Interfaces 4

**4. System Features 4**

4.1 Input CSV 4

4.2 Run Instant Runoff 5

4.3 Run Open Party Listing 6

4.4 Create Audit 6

4.5 View Audit 6

4.6 Create Media Report 7

4.7 View Media Report 7

4.8 Return Results to Display 7

4.9 Shuffle Votes 8

4.10 Tie Breaker 8

**5. Other Nonfunctional Requirements 8**

5.1 Performance Requirements 8

5.2 Safety Requirements 8

5.3 Security Requirements 8

5.4 Software Quality Attributes 9

5.5 Business Rules 9

**6. Other Requirements 9**

**Appendix A: Glossary 9**

**Appendix B: Analysis Models 9**

**Appendix C: To Be Determined List 9**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| **Vote Counting System** | 2/16/2021 | Initial Release | 1.0 |
|  |  |  |  |

# Introduction

## Purpose

Purpose of this document is to present a detailed description of version 1.0 of the Vote Counting System. It will explain how the system works, what the system will do, the constraints of the system and how people will interact with the system. This system will work for OPL and IR voting types.

## Document Conventions

Document based on the IEEE template for System Requirement Specification Documents. No special fonts will be used for different documentation.

## Intended Audience and Reading Suggestions

This document may be read by typical users such as election officials or the media who are mainly concerned with viewing the results of an election. Programmers and testers who want to make sure that some of the voting algorithm implementations are functionally correct or add additional features such as a GUI. For full comprehension of the system, read the SRS from beginning to end. For understanding of functionality, see the use cases in section 4.

## Product Scope

The product will create a Vote Counting System application. This will be a system where a file that includes the votes and type of Vote Counting System will be entered in and a file listing the winner and methodology will be given. This gives a way to document the voting process and results in an efficient and convenient manner. When dealing with hundreds of thousands of ballots an automated system is essential to reduce errors and increase speed.

## References

IEEE Template for System Requirement Specification Documents:

<https://goo.gl/nsUFwy>

Instant Runoff Ballot:

<https://www.fairvote.org/rcv#where_is_ranked_choice_voting_used>

Proportional voting type:

https://www.fairvote.org/how\_proportional\_representation\_elections\_work

# Overall Description

## Product Perspective

This Vote Counting System was developed as a new, self-contained product. Its design is for the use of determining the winner of an election. It can run one of two Vote Counting Systems, Open Party List voting (OPL) or Instant Runoff Voting (IRV) by counting the votes from a CSV file. It was developed to run on Windows, MacOs, and Linux (more specifications can be found at Section 2.4).

## Product Functions

*2.2.1* System Use Cases

* Return Results To Display: Once a winner is determined by the program, it will need to display the results to the screen/terminal of where the program was run. These results may include the rankings of the election, their names and party, type of election, and number of ballot
* Coin Flip Tie Breaker: In the event of two candidates resulting in a tie, flip a coin to break the tie and proceed with the winner of the coin flip.

*2.2.2* Testers Use Cases

* Test using manual input: User manually inputs type of the vote, number of candidates, name of the candidate, political party’s name and number of votes they have.
* Shuffle Votes: The order in which voters are counted is shuffled.

*2.2.3* Election Officials

* Input CSV File: Allows users to use a CSV file which contains the election votes to determine the winner of an election.
* Generate Audit File: Create an audit for the Vote Counting System.
* Create Media Report: Create a report of the election results for the media.
* Run Instant Runoff Voting: Instant runoff voting will be conducted with the file of votes provided.
* Run Open Party Listing Voting: This is where citizens vote for their preferred candidate of their choice. The candidates may belong to a specific party or independent.

*2.2.4* Observer (Media) Use Cases

* View Audit: The election officials view a detailed audit of the election results.
* View Results and Reports: Create a report of the election results for the media.

## User Classes and Characteristics

Here is a list of the various user classes from most important to least:

* Election Officials will be able to use this software to run elections and determine the winner
* Testers and Programmers such as software engineers or researchers will be able to run, test, and develop the software.
* Observers (Media) such as reporters will be able to view the audit and results of the election

## Operating Environment

To use this Vote Counting System, check this list of System Specifications.

Operating Systems:

* Ubuntu  
  *Minimum:* Ubuntu 16.04  
  *Recommended:* Ubuntu 18.04 or higher
* Windows  
  *Minimum:* Windows 7  
  *Recommended:* Windows 10
* Mac Os X  
  *Minimum:* 10.12 Sierra  
  *Recommended:* 10.15 Catalina or higher

Hardware Requirements:

* Processor  
  *Minimum:* Intel Core i5 3570 or AMD FX 6300  
  *Recommended:* Intel Core i7 4770 or AMD Ryzen 1800X or equivalent or higher
* RAM  
  *Minimum:* 8 GB DDR2  
  *Recommended:* 32 GB DDR3 or equivalent or higher
* Graphics Card:  
  *Minimum:* Nvidia Quadro M2000 or AMD FirePro W5100  
  *Recommended:* Nvidia Quadro RTX 4000 or Radeon Pro W5700 or equivalent or higher

Recommended System:

* Windows 10 Intel Core i7 @ 4.2GHz (x4) 32 GB RAM Nvidia Quadro M2000
* Ubuntu Intel Core i7-4770 3.40GHz 16 GB RAM AMD Radeon HD 8570
* Mac OS X Intel Core i5 3.2 GHz 16 GB RAM 2 x NVIDIA® GeForce GT 755M 1024MB

## Design and Implementation Constraints

The Vote Counting System will limit the user to only run one election at a time and therefore will only require one CSV election file when prompted. Once the election is run, an audit file will be generated and placed in the same directory. For every audit file that is generated (if more than one election is run with its respective CSV election file in the same directory), they will need to be named distinctively so that it can be recorded correctly. The user should be able to input a CSV election file of up to 100,000 ballots, choose the Vote Counting System, count the votes, determine the winner, generate an audit file, and display results in under 8 minutes.

The Vote Counting System is developed in Java, so the compiled executable relies on its class and Java files.

## User Documentation

There are no user documentation components that are available at this time.

## Assumptions and Dependencies

Since the Vote Counting System is its own, new, self-containing product, it will not need to rely on any other products to run. The Vote Counting System is developed in Java and therefore will require Java to be installed on the user’s system. The latest version of this Vote Counting System will require at least Java version 8 or higher to run correctly. This applies to Windows, MacOs, and Linux users. CSE lab machines will have all the required software and standard Java libraries to run the system (refer to section 2.4)

# External Interface Requirements

## User Interfaces

The vote processing software is accessible through the terminal (if using Ubuntu or Mac OS X) or command prompt (on Windows OS). No outside software components will be needed. The program will be run on the vote CSV data and output the results along with the audit into the same file directory as the vote CSV data. The User will run the program executable and submit the path of the election results CSV when prompted. The User will solely use the terminal or command prompt for navigating the *vote counting system*. Output to the user will either appear in the terminal or command prompt or they will be stored as a file in the same directory as the election results.

## Hardware Interfaces

The vote processing software is self-contained on the machine it is run on and does not need to interface with hardware devices.

## Software Interfaces

The vote processing software connects with no other software components at this time. However, it may interface with a Polling System in the future to directly transfer votes. In the case of this interfacing, the Polling System will need to store election information in a file and transfer that file to the Vote Counting System.

## Communications Interfaces

The vote processing software does not communicate with other software or end-points. At this time, the application takes input from the local machine, runs on the local machine, and outputs results to the local machine.

# System Features

**4.1 Input CSV File**

4.1.1 Description and Priority

Election officials will be able to input an election CSV file in the terminal along with the call to the program. The election CSV file must be in the same directory as the program executable.

4.1.2 Stimulus/Response Sequences

The election CSV file must be in the same directory as the Vote Counting System program executable. The user will first write the executable’s name in the command line, then following after a single space, the election CSV file name. If there is no file, or the file is invalid, then the program will not run and will quit immediately. The user will then need to re-enter the executable and the correct election CSV file.

4.1.3 Functional Requirements

The system will allow the user to enter the executable and an election CSV file into the command line to run an election and determine the winner.

(UseCase\_Team9\_I1)

**4.2 Run Instant Runoff**

4.2.1 Description and Priority

Instant runoff voting will be conducted with the file of votes provided.

4.2.2 Stimulus/Response Sequences

The user will enter the program executable along with the election CSV file in the terminal. Once the program runs, counts the votes, determines the winner, and creates the audit file, the results of the election will be automatically displayed to the screen.

System will count and distribute ballots for each candidate starting with the primary choice. All results are logged for an audit report. The least popular candidate will be marked as eliminated and their ballots are distributed among the other candidates based on the next highest vote level. The system logs final results and then finishes when there is only one candidate left. System will display to the screen the results of the election and output an audit log.

4.2.3 Functional Requirements

The system must be able to determine a winner for any given election and show the

repeatable process

(UseCase\_Team9\_L1)

**4.3 Open Party Listing**

4.3.1 Description and Priority

OPL will be conducted with the file of votes provided.

4.3.2 Stimulus/Response Sequences

The user will enter the program executable along with the election CSV file in the terminal. Once the program runs, counts the votes, determines the winner, and creates the audit file, the results of the election will be automatically displayed to the screen.

System will count and distribute ballots for each candidate starting with the primary choice. All results are logged for an audit report.

Also, the system will calculate the number of seats available for each party ( Democratic etc.). The winner will be the ones that have the highest number of votes.

System will display to the screen the results of the election and output an audit log.

4.3.3 Functional Requirements

The system must be able to determine a winner for any given election and show the repeatable process

(UseCase\_Team9\_A1)

**4.4 Create Audit**

4.4.1 Description and Priority

Programmers will run the election program and then can create an audit.

4.4.2 Stimulus/Response Sequences

The programmer will run the election and then be prompted about whether or not they want an audit. If yes they will generate an audit file that will be sent to the election officials for viewing. If no, they will then be brought to the next prompt.

4.4.3 Functional Requirements

Audit will be created.

(UseCase\_Team9\_R2)

**4.5 View Audit**

4.5.1 Description and Priority

This is where election officials can view an audit that details over the election results.

4.5.2 Stimulus/Response Sequences

The programmer will run the election and then be prompted about whether or not they want an audit. If yes they will generate an audit file that will be sent to the election officials for viewing.

4.5.3 Functional Requirements

Election officials will view the audit.

(UseCase\_Team9\_R1)

## 4.6 Create Media Report

4.6.1 Description and Priority

Programmers will run the election program and then can create a report for the media.

4.6.2 Stimulus/Response Sequences

The programmer will run the election and then be prompted about whether or not they want an audit. If yes or no they will then be asked if they want a media report. If yes they will generate a report that then can be sent out to the media that have requested the report.

4.6.3 Functional Requirements

Media report will be created.

(UseCase\_Team9\_R3)

## 4.7 View Media Report

4.7.1 Description and Priority

Media can view a snapshot of the election results.

4.7.2 Stimulus/Response Sequences

Media will request the programmers that they want a media report. If the request is granted the programmer will send them a report.

4.7.3 Functional Requirements

Media can view media reports.

(UseCase\_Team9\_R4)

**4.8 Return Results to Display**

4.8.1 Description and Priority

The system will display the results to the screen. These results may include the rankings of the election, their names and party, type of election, and number of ballots.

4.8.2 Stimulus/Response Sequences

The user will enter the program executable along with the election CSV file in the terminal. Once the program runs, counts the votes, determines the winner, and creates the audit file, the results of the election will be automatically displayed to the screen.

4.8.3 Functional Requirements

The system must be able to display the results of the election to the screen back to the user.

(UseCase\_Team9\_I2)

**4.9 Shuffle Votes**

4.9.1 Description and Priority

The order in which voters are counted is shuffled

4.9.2 Stimulus/Response Sequences

System selects a new order for the voter list to be in. System ensures that it is different from the current and returns the ballots in shuffled order

4.9.3 Functional Requirements

The system must be able to shuffle the order of votes

(UseCase\_Team9\_L3)

**4.10 Tie Breaker**

4.10.1 Description and Priority

A tie breaker is decided among two or more candidates

4.10.2 Stimulus/Response Sequences

System records the list of candidates in a tie. If there is only one candidate in the list, then they are returned as the winner. 1001 coins are flipped to decide between two tied candidates

4.10.3 Functional Requirements

The system should be able to break a tie between two or more candidates.

(UseCase\_Team9\_L2)

# Other Nonfunctional Requirements

## Performance Requirements

The performance requirements for this program should be able to process 100,000 ballots in 8 min, this is a critical requirement due to time constraints (input volume). Duration for each volume (input file) might be slightly different across multiple operating systems, but the output rate will be the same.

## Safety Requirements

There is no safety requirement for this software.

## Security Requirements

From a security standpoint, there are no requirements for this software. But people who will use it need to identify all security protocols of who should do certain steps. For example, they shouldn’t allow all employees to use this software to be able to run certain ballots.

Although the audit and report will only be “Read Only” mode, they need to watch for anyone who wants to delete that file to favor other candidates (the one they voted for).

## Software Quality Attributes

* Able to run on widely used OS around the world ( Linux, Macintosh, Windows etc.).
* The system calculates all the votes and generates audits & media reports to show the accuracy of the system. Also, it will allow you to go back and double check the accuracy of the system.
* Anyone who can read and understand basic english sentences should be able to use this software since it does not need special training. (Self explanatory).
* The system is also reliable and easy to use (usability).
* Resilience due to no server connection is needed and for the case of the programs to fail, the program will output an error message.
* The system will be available as long as you have the required operating system (check 2.4 OS environment).

## Business Rules

There is no business plan for this software. This is a free software built for election officials to use during elections.

# Other Requirements

No other requirements for this software besides those mentioned in Section 2.

**Appendix A: Glossary**

CSV - Comma separated values file

IR - Instant runoff voting, linked in section 1.5

OPL - Open party listing, linked in section 1.5

**Appendix B: Analysis Models**

None at this time.

**Appendix C: To Be Determined List**

None at this time.