Software Requirements Specification

for

IR and CPL Voting System

Version 2.0 approved

Authors:

Codey Camerer (camer441)

Zoë Foster (foste924)

Adelaide Granse (grans036)

Hailey Koster (koste116)

University of Minnesota - Twin Cities

February 2023

Table of Contents

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 **2**

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 **5**

3.4 Communications Interfaces **5**

4. System Features **5**

4.1 Accepting an Election Data Input File **5**

4.2 Parsing the Input File **7**

4.3 Prompting User for Additional Election Information **9**

4.4 Validate Election Type **10**

4.5 Creating Audit File **12**

4.6 Writing to Audit File **13**

4.7 Run an Instant Runoff Election Calculation **14**

4.8 Run a Closed Party List Election **16**

4.9 Determining seat allocation for Closed Party List Voting **17**

4.10 Running Popularity Election Calculation **19**

4.11 Leveling a Tie by Fair Coin Toss **20**

4.12 Leveling a Tie by Fair Coin Toss **21**

4.13 Finalizing Audit File **22**

4.14 Display Winner and Percentage **23**

5. Other Nonfunctional Requirements **25**
- **5.1 Performance Requirements** **25**
- **5.2 Safety Requirements** **25**
- **5.3 Security Requirements** **25**
- **5.4 Software Quality Attributes** **26**
- **5.5 Business Rules** **26**

6. Other Requirements **26**

Appendix A: Glossary **26**

Appendix B: Analysis Models **27**

To Be Determined List **28**

Revision History

Name	Date	Reason For Changes	Version
Initial Draft	2/13	Initial draft of documentation	1.0
IR and CPL Voting System	2/16	Rewording and reformatting content	2.0

1. Introduction

1.1 Purpose

The purpose of this document is to describe, in detail, a project to determine election winners via two voting algorithms, Instant Runoff Voting and Closed Party List Voting. This document will cover the purpose and features of the software. This includes all interfaces of the system, the details of all that the system will do, as well as a cohesive description of the constraints the system will operate within. This document is intended for the developers of the system, potential future developers, as well as the users of this system.

1.2 Document Conventions

The IEEE template for System Requirement Specification was used as a basis for the creation of this document.

1.3 Intended Audience and Reading Suggestions

This document will be most informative and useful when read in completion, however, several sections may be of special interest to certain readers.

There are several intended audiences for this document, the main and most important user being election officials. Election officials would be using this system as a portion of the election process, and may therefore find this document useful in understanding how the system works. The section of most pertinence to the future developers would be section 2, the overall description, and this section will be the most digestible to an average user.

The second intended audience would be any future developers or engineers who may be looking to improve-upon or update this software. Reading this document will be helpful in understanding how the original system was developed, as well as understanding how the system functions. Likely, a future developer would need information included in all sections of this document.

The final intended audience for this document would be testers. Having a thorough understanding of the system will help in their work to test the functionality of the product. The section of most pertinence to the testers would be section 4, the system features, especially the use cases described in that section.

1.4 Product Scope

This software will be a voting system, including two voting algorithms: Instant Runoff Voting and Closed Party Voting. The system is designed for the use of election officials when all votes have been casted and counted. The system will fairly calculate the election winners based off of the chosen voting algorithm, either Instant Runoff Voting or Closed Party List Voting, and user-provided ballot and cast votes. The system will ensure all votes are counted according to the algorithms in a fair and consistent manner, and the system will provide the information pertaining to the winners of each election.

1.5 References

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

Project 1 - Waterfall Methodology. CSCI 5081, Software Engineering 1. University of Minnesota, 2023.

Frost, J. (2022, August 26). The Monty Hall Problem: A Statistical Illusion. Statistics By Jim. Retrieved February 15, 2023, from https://statisticsbyjim.com/fun/monty-hall-problem/

2. Overall Description

2.1 Product Perspective

This system is designed to aid in the election process. It is designed to handle a specific portion of the electoral process; the counting of votes and determination of a winner. All pre and post services will not be handled by this product. It is not a component of a larger system, though it has the potential to be added to a larger automated electoral process, and this document may be helpful in that possible future development.

See Appendix B.1

2.2 Product Functions

The program will be broken into 3 major components, accepting an input file or ballot, running one (and only one) of two voting algorithms, and delivering the election results to the user. These processes can be seen visually in the diagrams of Appendix B, and are listed in brief below.

Accepting the electoral information:
- Input file: user will submit a closed, readable file to the system.
- Determine voting type: system will read the first line of the file to determine the voting type.
- Additional information: any information not received from the ballot will be prompted to the user.

Instant Runoff Voting:
- Read file: read file to determine number of parties and the number of votes.
- Count votes (first): determine number of votes cast ranking a candidate as first.
- Determine winner: select the candidate with a 50% majority as the winner.
  - see Appendix B.2 for a more detailed explanation.
- Repeat until the election is complete.

Closed Party Voting:

- Read file: read file to determine number of parties and the number of seats.
- Count votes: determine the number of votes cast for each party.
- Determine quota: divide the number of votes by the number of seats.
- Determine winner: divide each candidate's vote by the quota to determine the number of seats that party will receive. Use the remainder of this division to allocate any remaining seats.
- Allocate candidates to seats: according to the set ranking of the candidates, assign candidates to their party seats.

Deliver the election results:

- Write audit: system will compile audit file with electoral result and metadata for the user
- Deliver audit: system will deliver the audit file to the user
- Display result: system will display the election result(s) for the user

2.3 User Classes and Characteristics

There are two main users of this product. The first, the typical user, is the election officials. This product is to be used by election officials at the conclusion of voting and counting of votes. The user will provide the input file consisting of all counted votes, and the user will receive an audit file containing election results. The election officials are the main and most important users.

The second expected user would be the testers of this product. Testers will provide an input file either in the same method as the election officials, or they may provide input via the terminal or script. They will also receive results at the termination of the system. Though their purpose is to test the system for proper function, their experience should be closely similar to that of the typical user.

2.4 Operating Environment

The program will be designed to run on the UMN CSE Computer labs that operate with the following properties:

- Operating system: Ubuntu 20.04, Windows 10
- Kernel version: #64~20.04.1-Ubuntu SMP Fri Jan 6 16:42:31 UTC 2023
- Processor architecture: x86_64
- CSE lab machines: csel-kh1254-#.cselabs.umn.edu
- C++ version: gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
- Make version: GNU Make 4.2.1

2.5 Design and Implementation Constraints

This system is designed in C++. The organization is responsible for maintaining and ensuring future operating system changes are compatible with the system.

2.6 User Documentation

There will not be any user documentation components.

2.7 Assumptions and Dependencies

The following points are assumed factors and dependencies:

- The system is developed to be used on certain operating systems, see section 2.4.
- This system is to be used in elections, and will promise a fair result delivered at the completion of the program, but only when it is assumed that the information imputed to the system, the ballot, is fair.
- It is assumed that the system is protected and only accessed by authorized personnel to the election, the system will not provide those security measures.
- The input file must be honest and representative of the election. Our system is not responsible for false election results due to data tampered prior to the file's input to the system.

3. External Interface Requirements

3.1 User Interfaces

The user will be prompted to upload a file to the system. They can do this in one of two ways:

1. Providing file path when prompted
    a. System is started by running the program file
    b. After the program is running, the user is prompted to enter the file path or file name through the terminal.
        i. File should already be in the same directory as the program file
2. Providing file path through the terminal
    a. System is started by running the program through the terminal
    b. When running the program the file path and name is provided as an argument

After the file has been provided to the system, the user will also be asked to provide the date of election. This will be used for naming the audit file:

1. User is prompted to enter the date of the election through the terminal
2. The date should be provided in MM/DD/YYYY format

3.2 Hardware Interfaces

To interact with the system the user will be required to have a monitor or screen of some kind to view the program's display components. Additionally they will need a keyboard and mouse in order to interact with the program.

In addition to the above requirements, the program will be required to run on all of the UMN CSE lab computers and servers with hardware specifications as follows:

- Dell Precision 3630, Intel® Core™ i7 @ 3.2GHz (x6), 32 GB RAM
- Dell Precision T3420, Intel® Core™ i5 @ 3.4GHz (x4), 32 GB RAM
- Dell OptiPlex 9020, Intel® Core™ i7 @ 4.2GHz (x4), 32 GB RAM
- Dell Precision 3650 Tower, Intel Core i7 @ 2.5 GHz (x8), 32 GB RAM
- Lenovo Workstation TS P620, Threadripper Pro 5945WX 12 Core Processor, 32 GB RAM
- Dell Precision T3620, Intel® Core™ i5 @ 800Hz (x3), 32 GB RAM
- Dell Precision T3420, Intel® Core™ i5 @ 3.4GHz (x4), 64 GB RAM
- Dell OptiPlex 9020, Intel Core (Quad-core) i7 @ 3.6 GHz
- Dell Precision Tower 3620, Intel® Core™ i5 @ 4.2GHz (x4)
- Dell Precision T1700 Intel® Core™ i7 @ 3.6GHz (x4) 32 GB RAM
- Dell Precision tower 3620, Intel® Core™ i5 @ 900 MHz (x3), 32 GB RAM
- Dell PowerEdge R740, Tesla T4 GPU, 2 x Intel® Xeon® Gold 6148 CPU @ 2.40GHz (40 cores per machine), 96 GB RAM
- Intel® Xeon Phi(TM) CPU 7290F @ 1.50GHz (72 cores, 4 threads/core)
- VMWare virtual machine, 2x Intel® Xeon® CPU E5-2695 v3 @ 2.30GHz, 16 GB RAM
- Dell PowerEdge R815, 4 x AMD® Opteron® 6220 (Eight-Core) @ 3.00GHz, 192 GB RAM
- Dell PowerEdge R330, 4x Intel® Xeon® CPU E3-1220 v6 @ 3.00GHz, 64 GB RAM
- Dell XPS 8910, Intel Core i7-6700 CPU 3.40GHz, 16GB RAM, Nvidia GeForce GTX 1080
- Dell Precision 3630, Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, 32 GB RAM, DVD+/-RW, Intel® UHD Graphics 630
- Dell OptiPlex 7050, Intel Core i7-7700 3.6GHz, 32 GB RAM, DVD+/-RW, Intel® UHD Graphics 630
- Dell Precision Tower 7910, Intel Xeon e5-2460 2.4ghz 20 cores, 32 GB RAM, DVD-RWX drive, Nvidia Quadro M200

3.3 Software Interfaces

The program will support the Windows 10 and Ubuntu 20.04 operating systems. Prior to use of the system an external program will be required for creation of a CSV file (Notepad or Excel). This file will be stored in the same directory as the program file.

3.4 Communications Interfaces

The system will not require an internet connection in order to run, and therefore will not require any communication interfaces with external systems. The file used to run the system will be required to be in the same directory as the program file, eliminating the need for an internet connection.

4. System Features

4.1 Accepting an Election Data Input File

4.1.1 Description and Priority

The system must be able to open a file that will be parsed for information about the election. (*you can not manually input votes*) This functionality is high priority since the data from the file will be needed for all other parts of the system.

4.1.2 Stimulus/Response Sequences

1. Upon starting the program the system will check if a filename was passed through the command prompt. If a filename is available, the system will skip step 2 and move to step 3.
2. If a filename was not provided in the command prompt, the system will prompt the user to enter a filename.
3. The system will attempt to open the file. If the file is opened, the system will acknowledge to the user that the file was opened successfully. Else, if the system could not open the file, it will inform the user of the failure and will prompt the user for a new filename. It will then attempt to open the file with the new filename.
4. The system will parse the file for election information and store information in memory.

4.1.3 Functional Requirements

REQ-1.1: The actual voting will be done separately from the voting system you are developing. Ballots will be cast online and a comma delimited (CSV) text file will be provided to you.You may assume that there are no numbering mistakes in the file (e.g. a voter will not make any mistakes on the ballot.)

REQ-1.2: You can pass the name of the file into the program as a command line argument or ask for the name within the program itself.

REQ-1.3: There will never be more than one file given to you per election.

REQ-1.4: The election file will be located in the same directory as the program.

4.1.4 Use Case for Accepting an Election Data File

Name	Accepting an Election Data Input File
ID	UC_001
Description	After an election has occurred a file with information about the election and the ballot data needs to be uploaded to the program to determine the type of election and the winner of the election.
Actors	Election official, System testers, System Engineers.
Organizational Benefits	Receive election data for processing.
Frequency of Use	Once per election (normal elections and special elections).
Triggers	Program is run.
Preconditions	● Voting has already been completed. ● File name is known by the user. ● File is in the same directory as the system. ● File must be of CSV read-only format.

Postconditions	The file is parsed for election information.
Main Course	1. System prompts user for filename 2. User enters filename 3. System accesses the file and opens it for parsing.
Alternate Courses	File name passed through the command line as an argument: 1. System accesses the file and parses it.
Exceptions	System fails to find file 1. System notifies the user that an error has occurred. 2. Return user to Main Course step 1.

4.2 Parsing the Input File

4.2.1 Description and Priority

The system will take the opened input file and parse it for election information it needs to make an election calculation. This information will be stored in memory for easy access during the election calculations.

4.2.2 Stimulus/Response Sequences

1. The system checks whether the header fits a IR or CPL data structure.
2. The system reads the header into the applicable election data structure.
3. The ballots are read into candidate data structures.
4. The input file is closed.

4.2.3 Functional Requirements

REQ-2.1: You will need to read in the file. How you choose to do this is up to you. Remember, it will be a comma separated values (CSV) file where each row is separated by a newline. The file will be exported from Excel into the CSV format. All preprocessing of the file will be done before you receive it. The first line of the file will list the type of voting (i.e. IR, CPL).

REQ-2.2: When the program is run, the user can be prompted for any needed information that cannot be extracted from the file. ~~It is not a requirement for the Waterfall to be able to get all needed information directly from the file (e.g. number of votes, number of candidates, etc).~~ You are encouraged to process the file for information, but file input can be very time consuming. For example, you will need to know how many parties there are and the candidates under these parties. (This initial requirement was changed during elicitation. See REQ-2.9)

REQ-2.3: There will be no write-in candidates for this system. It may happen in the future but not now.

REQ-2.4: You cannot change the file structure outside of the program since the election files will come in the predetermined format.

REQ-2.5: You can assume there are no errors in the ballots. Each ballot will have at least 1 ranking. The ranking numbers will not have issues (e.g.1,,3,4 : missing 2 will not happen).

REQ-2.6: For a closed party list ballot, voters express a preference for a particular party. Each ballot will only have one candidate indicated as a choice for the vote.

REQ-2.7: For an Instant RunOff ballot, each ballot must have at least one of the candidates ranked as their top choice. For this iteration, you may assume that if there is a ballot in the file, it will have at least one of the candidates ranked. As you will see, the more rankings you complete as the voter, the more likely your vote will actually be counted towards electing the candidate(s) that you have as your top rankings. The long-term goal is for this system to be part of an integrated online voting system and all preprocessing will be done via the voting system itself that you are developing. You do not need to do any preprocessing of the file. Instead of voting for one candidate, voters rank their candidates in order of preference. A candidate can be given only 1 ranking.

REQ-2.8: If you prompt a user, you can have just text or a wonderful GUI--your choice.

REQ-2.9: The ballot file header is guaranteed to have all the information needed to make calculations (Elicited Requirement)

REQ-2.10: See REQ-1.1

4.2.3 Use Case for Parsing the Input File

Name	Parse Input File
ID	UC_002
Description	The system will take the opened input file and parse it for election information it needs to make an election calculation.
Actors	System Engineer
Organizational Benefits	Receive election information from the input file and places it into memory for future election calculations
Frequency of Use	Once per election (normal and special elections included).
Triggers	An input file is accepted and opened by the system.
Preconditions	● System must have an input file opened.

Postconditions	● All information contained within the input file is copied into memory. ● The input file is closed.
Main Course	If line 1 is equal to the string "IR": 1. Instantiate IR data structure. 2. Create a number of candidate data structures that is equal to line 2 and place into IR data structure. 3. Split line 3 up and read into candidate data structures. 4. Read from line 5 to the line equal to the value contained in line 4, placing ranks into appropriate candidate structures. 5. The system closes the input file
Alternate Courses	If line 1 is equal to the string "CPL": 1. Instantiate CPL data structure. 2. Create a number of party data structures that is equal to line 2 and place them into the CPL data structure. 3. For lines 4 up to line (4 + the value of line 2), read into the respective party structures. 4. Read from line (4 + the value of line 2 + 1) to the line equal to the value contained in line (4 + the value of line 2 + 1), placing ranks into appropriate party structures. 5. The system closes the input file.
Exceptions	The header contains no information and is not spaced out by newline characters. (Current algorithm will always read into a CPL data structure. This may cause the system to read incorrect values or drop some votes)

4.3 Prompting User for Additional Election Information

4.3.1 Description and Priority

Users will be prompted for any needed information that cannot be extracted from the file. (e.g. number of candidates, number of seats, etc.)

4.3.2 Stimulus/Response Sequences

1. As soon as the election data file has been parsed, the system will check whether it has all the information needed to run an election calculation. If the system determines that it does not have the necessary information to run an election calculation, it will continue onto the next steps.
2. If the system could not determine the type of election from the input file, it will first prompt the user for the election type.
3. The system will use the election type to prompt the user for the additional information needed for whatever particular type of election that is going to be calculated.

4.3.3 Functional Requirement

REQ-3.1: See REQ-2.2, REQ-2.3, REQ-2.8

4.2.3 Use Case for Prompting a User for Additional Election Information

Name	Prompt User for Additional Election Information
ID	UC_003
Description	Users will be prompted for any needed information that cannot be extracted from the file. (e.g. number of candidates, number of seats, etc.)
Actors	Election Official, System Tester, System Engineers.
Organizational Benefits	Establish all information needed for election calculations
Frequency of Use	Zero to many times for an election.
Triggers	A reasonable value is not parsed from the input file.
Preconditions	File is accepted and is processed
Postconditions	All information needed for election calculations is determined. Calculations can proceed.
Main Course	1. Prompt the user for all NULL values within the Election Data Structure. 2. Checks if input is a reasonable value. 3. Stores value into variable for future calculations.
Alternate Courses	User inputs an unreasonable value: 1. System notifies the user of invalid input and prompts them to re-enter the value.
Exceptions	All information was extracted from the input file. Use case is skipped.

4.4 Validate Election Type

4.4.1 Description and Priority

A user should be able to verify the type of election that the system is going to compute prior to any actual computation. This will reduce the need for unnecessary computations.

4.4.2 Stimulus/Response Sequences

1. After the file is parsed and additional information is obtained from the user, the system will ask the user whether the election type in memory is the type of election the user wants the system to compute.
2. If the user agrees to the election type, that type of election can be computed. Else, if the user declines the election type, the system will prompt the user again for election information (see 4.3).

4.4.3 Functional Requirements

REQ-4.1: You need to write one program to handle both elections. The type of election will be stored on the first line of the file that contains the ballots.

REQ-4.2: This program will be run multiple times during the year at normal election times and special elections.

REQ-4.3: See REQ-2.2, REQ-2.8

4.4.4 Use Case for Verifying Voting Type

Name	Verify Voting Type
ID	UC_004
Description	A user should be able to verify the type of election that the system is going to compute prior to any actual computation.
Actors	Election official, System testers, System Engineer.
Organizational Benefits	Reduce erred and accidental computations.
Frequency of Use	One to many times in an election.
Triggers	The election data structure has all the information needed to run an election.
Preconditions	Input file is processed into the election data structure and all required data fields are filled with reasonable values.
Postconditions	Audit file is created. Program that aligns with the voting type is run.
Main Course	1. System ask whether the election type in memory is the type of election the user wants to calculate 2. If the user accepts the election type, that election type's calculation is run (see UC_007 or UC_008)
Alternate Courses	The user refuses the election type in memory. The system returns to UC-003

Exceptions	TBD

4.5 Creating Audit File

4.5.1 Description and Priority

An audit file must be created and named. This audit file will log the calculations of the election computation. The audit file will be written onto while the election calculations are being done. The process and winner will be logged on this file as well.

4.5.2 Stimulus/Response Sequences

1. The system will prompt the user for a name to associate with the file.
2. System creates a file with a combination of the name provided and unique identifiers. Unique identifiers will include date and election type.
3. The election type and details are written to the start of the file.

4.5.3 Functional Requirements

REQ-5.1: You will need to produce an audit file with the election information at the time (e.g. Type of Voting, Number of Candidates, Candidates, Number of Ballots, calculations, how many votes a candidate had, etc), you should list the winner(s), and you should show how the election progressed so that the audit could replicate the election itself. You should show who got what ballot and the ballots order of being received if applicable.

REQ-5.2: Your audit file is a strict requirement and an auditing should be able to follow the order of the ballots being assigned to the candidates. You should show the order of removal of candidates in IR and what ballots were redistributed. The file should show all of the steps.

REQ-5.3: Audit files should automatically compose a filename made up of type of election and date(e.g. IR02022023.txt) (Elicited Requirement)

REQ-5.4 The user should have the option of naming the audit file. If name is added, apply name to filename (e.g NAME_IR2022023.txt)

REQ-5.5: See REQ-2.2

4.5.4 Use Case for Creating Audit File

Name	Creating Audit File

ID	UC_005
Description	An audit file must be created and named. This audit file will log the calculations of the election computation.
Actors	Election Official, System Tester, System Engineers.
Organizational Benefits	The audit file will be written onto while the election calculations are being done. The process and winner will be logged on this file as well.
Frequency of Use	One time an election.
Triggers	The user chooses to compute an election.
Preconditions	Write permissions. User needs a name to identify the file with.
Postconditions	An open/working audit file that has a name and unique identifier associated with it.
Main Course	1. Prompt user for a name to associate with the file. 2. System creates a file with a combination of the name provided and unique identifiers. Unique identifiers will include date and election type. 3. The election type and details are written to the start of the file. (triggers UC-006)
Alternate Courses	User does not give a name to associate with audit file: 1. System creates a file with a combination of unique identifiers. Unique identifiers will include date and election type.
Exceptions	The program does not have write permissions: The user will have to adjust operating system settings to allow proper function of program.

4.6 Writing to Audit File

4.1.1 Description and Priority

The system will write to the audit file to log election type, election process and election winner. The information about the election calculations will be saved for future reference and/or verification.

4.1.2 Stimulus/Response Sequences

1. The calculation that triggers the write sequence is specified on the audit file, followed by the calculations.
2. New line operator is added.

4.1.3 Functional Requirements

REQ-6.1: See REQ-5.1, REQ-5.2

4.1.4 Use Case for Writing to Audit File

Name	Writing to Audit File
ID	UC_006
Description	The system will write to the audit file to log election type, election process and election winner.
Actors	System Engineer.
Organizational Benefits	The information about the election calculations will be saved for future reference and/or verification.
Frequency of Use	Many times in an election.
Triggers	● Audit file is created ● After election computations.
Preconditions	Write permissions. Audit file must be created and opened.
Postconditions	File contents saved to audit file. File remains open.
Main Course	1. The calculation that triggers the write sequence is specified on the audit file, followed by the calculations. 2. New line operation.
Alternate Courses	None
Exceptions	Unable to write. Need to inform the user of the problem. Might need to cancel calculations.

4.7 Run an Instant Runoff Election Calculation

4.7.1 Description and Priority

If a user chooses to run an Instant Runoff (IR) election calculation, the system must be able to use the Instant Runoff process to determine a clear majority. See Appendix B.2 for a flow chart of this process.

4.7.2 Stimulus/Response Sequences

1. User chooses to run an IR election.
2. System checks for majority (>50%). If a majority is determined, skip to step 4.

3. The candidate with the lowest number of votes is eliminated and their votes are distributed among the remaining candidates and steps 2-3 are repeated until a clear majority is determined. Popular vote commences if no clear majority.
4. Winner is logged.

4.7.3 Functional Requirements

REQ-7.1: If there is not a clear majority in IR, then popularity wins after all votes have been handed out.

REQ-7.2: If there is ever a tie, flip a coin via the computer. You must randomly select the winner in a fair coin toss.

REQ-7.3: For Instant Runoff Voting, the ballot will have at least 1 person ranked. You can rank from 1 to the number of candidates. Remember, we do <u>not allow</u> write in candidates.

REQ-7.4: See REQ-2.7

4.7.4 Use Case for Instant Runoff Election Calculation

Name	Running Instant Runoff Election Calculation
ID	UC_007
Description	Instant runoff voting process will be used to establish a clear majority of the Instant Runoff Election and will log the calculation and results.
Actors	Election Officials, System Testers, System Engineers
Organizational Benefits	This function will determine if a clear majority exists in an IR election, or it will determine whether a popular vote is needed to determine a winner of the IR election.
Frequency of Use	At most 1 time for an election.
Triggers	User chooses to run an Instant Runoff election calculation.
Preconditions	Audit file has been opened and the user has chosen to run an Instant Runoff election calculation.
Postconditions	A winner has been determined for an Instant Runoff Election and calculations of the election have been logged.
Main Course	1) If a candidate receives over 50% of the first choice votes, he or she is declared elected (triggers UC_006) 2) If no candidate receives a majority, the candidate with the fewest

	votes is eliminated. (triggers UC_006) 3) Votes for the eliminated candidate are transferred to the remaining candidates. (triggers UC_006) 4) Repeat steps 1-3 until a clear majority is determined.
Alternate Courses	● 2 candidates ties for the fewest votes received during an iteration: A tie between 2 candidates must be leveled by a Fair Coin Toss TieBreaker (see UC_011) to determine the candidate to be eliminated. ● More than one candidate ties for the fewest votes received during an iteration: A tie between 3 or more candidates must be leveled by a Lottery TieBreaker (see UC_012) to determine the candidate to be eliminated.
Exceptions	No clear majority is found; popularity vote commences.

4.8 Run a Closed Party List Election

4.8.1 Description and Priority

If a user chooses to run a Closed Party List (CPL) election calculation, the system must be able to use the Instant Runoff process to determine a clear majority. This feature will automate the process of determining the quota, allocating seats compared to that quota, and when to call upon the highest remainder approach functionality. See Appendix B.3 for a flow chart of this process.

4.8.2 Stimulus/Response Sequences

1. Determine a quota by taking the votes received for a party divided by total number of seats available..
2. The quota is then divided into the votes that each party receives and the party wins one seat for each whole number produced.
3. If any remainder votes remain, they are distributed through the Highest Remainder Approach.

4.8.3 Functional Requirements

REQ-8.1: For closed party listing, all independent groups will have a single person in the group. You could have different named independent candidates (each in their own group.) The name of the group will have to be different (e.g. Independent1, Independent2.)

REQ-8.2: We are doing closed party listing and not open. The order of the candidates have a particular order. You will calculate the winner(s) based on the number of seats and the order of the candidates.

REQ-8.3: See REQ-2.6

4.8.4 Use Case for Closed Party List Election Calculation

Name	Running a Closed Party List Election Calculation
ID	UC_008
Description	The Closed Party List (CPL) process will be used to calculate how many seats are allocated to each party.
Actors	Election Officials, System Testers, System Engineers.
Organizational Benefits	This feature will automate the process of determining the quota, allocating seats compared to that quota, and when to call upon the highest remainder approach functionality (see UC_009).
Frequency of Use	At most 1 time per election.
Triggers	User chooses to run a CPL election calculation.
Preconditions	● Audit file has been opened and the user has chosen to run a CPL election calculation. ● Highest Remainder Approach functionality must be implemented.
Postconditions	Seats are allocated, logged and display to terminal
Main Course	1. Determine quota by taking the total number of valid votes in the district and dividing this by the number of seats. (triggers UC_006) 2. The quota is then divided into the vote that each party receives and the party wins one seat for each whole number produced. (triggers UC_006) 3. If any remainder votes remain, they are distributed through the Highest Remainder Approach (triggers UC-006 and UC_009)
Alternate Courses	There are more seats allocated than there are candidates: The seats will be allocated one by one in a lottery fashion. (see 4.12)
Exceptions	TBD

4.9 Determining seat allocation for Closed Party List Voting

4.9.1 Description and Priority

Closed Party List (CPL) voting will be used to allocate seats appropriately. This will automate the the process of the number of seats allocated to each party

4.9.2 Stimulus/Response Sequences

1. The party with the highest remainder is allocated a seat.
2. The next party with the second highest remainder is allocated the second seat.
3. Repeat until all seats are allocated.

4.9.3 Functional Requirements

REQ-9.1: In CPL, allocate leftover seats of the winning party to other parties by lottery. (Elicited requirement)

REQ-9.2: See REQ-8.2

4.9.4 Use Case for Determining Seat Allocation for Closed Party List Voting

Name	Determining seat allocation for Closed Party List Voting
ID	UC_009
Description	Closed Party List (CPL) voting will be used to allocate seats appropriately.
Actors	System Engineer.
Organizational Benefits	Automate the process of the number of seats allocated to each party.
Frequency of Use	Once per CPL noted election.
Triggers	Called by CPL calculations function
Preconditions	CPL calculations are running.
Postconditions	All seats have been allocated appropriately to a party.
Main Course	1. The party with the highest remainder is allocated a seat. (triggers UC_006) 2. The next party with the second highest remainder is allocated the second seat. (triggers UC_006) 3. Repeat until all seats are allocated.
Alternate Courses	● Two parties have the same highest remainder number: Level the tie by using a fair coin toss (triggers UC_011) ● 3 or more parties have the same highest remainder values: Level the tie by lottery (see UC_012)
Exceptions	TBD

4.10 Running Popularity Election Calculation

4.10.1 Description and Priority

A popular vote is used to determine who has the most votes between two people. Used to determine a winner when Instant Runoff (IR) voting calculation cannot determine a majority

4.10.2 Stimulus/Response Sequences

The candidate with the most votes is determined to be the winner.

4.10.3 Functional Requirements

REQ-1: See REQ-7.1, REQ-7.2

4.10.4 Use Case for Running Popularity Election Calculation

Name	Running Popularity Election Calculation
ID	UC_010
Description	A popular vote is used to determine who has the most votes between two people.
Actors	System Engineer
Organizational Benefits	Automation. Used to determine a winner when Instant Runoff (IR) voting calculation cannot determine a majority.
Frequency of Use	<= ½ times an election
Triggers	There are only 2 candidates left in an Instant Runoff calculation and a clear majority has not been determined.
Preconditions	● An IR calculation must be initiated. ● Both candidates must be fully ranked. ● No clear majority is determined in IR
Postconditions	A clear winner between the 2 candidates.
Main Course	The candidate with the most votes is determined to be the winner. (triggers UC_006)
Alternate Courses	Both candidates have the same number of votes: Level the tie with a fair coin toss. (see UC_011)
Exceptions	TBD

4.11 Leveling a Tie by Fair Coin Toss

4.11.1 Description and Priority

If a tie between candidates occurs, randomly select the winner in a fair coin toss via the computer. This will determine a tie in a more efficient and unbiased way than if done by an individual.

4.11.2 Stimulus/Response Sequences

1. System seeds the random generator.
2. System randomly flips a coin 100 times
3. Winner of coin toss after 100 times selected

4.11.3 Functional Requirements

REQ-11.1: Runtime constraints: An election should be able to run 100,000 ballots in under 4 minutes.

REQ-11.2: See REQ-7.2

4.1.4 Use Case for Leveling a Tie by Fair Coin Toss

Name	Leveling a Tie by Fair Coin Toss
ID	UC_011
Description	If a tie between candidates occurs, randomly select the winner in a fair coin toss via the computer.
Actors	System Engineer.
Organizational Benefits	Determine a tie in a more efficient and unbiased way than if done by an individual.
Frequency of Use	Zero to many times during an election.
Triggers	● In CPL, more seats given than candidates, how to assign extra seats remaining ● In CPL, if there's a tie in allocating a seat ● In IR not all candidates have been ranked in ballot, and no clear majority ● In IR the lowest number of votes redistributed.
Preconditions	An election computation must be in process and there must be a clear (discrete) tie between 2, and only 2, candidates.

Name	Leveling a Tie by Fair Coin Toss
ID	UC_011
Description	If a tie between candidates occurs, randomly select the winner in a fair coin toss via the computer.
Postconditions	A clear winner is determined between two candidates.
Main Course	1. System seeds the random generator. 2. System randomly flips a coin 100 times 3. Winner of coin toss after 100 times selected (triggers UC_006)
Alternate Courses	None
Exceptions	The random process does not have enough randomness after 100 coin tosses: Increase the amount of flips until we eliminate bias. We must be sure not to affect time restraints.

4.12 Leveling a Tie by Lottery

4.12.1 Description and Priority

If a tie between 3 or more candidates occurs, randomly select the winner in a lottery process via the computer. By using a lottery, you eliminate bias produced by a fair coin toss between 3 or more candidates. (see https://statisticsbyjim.com/fun/monty-hall-problem/ for a summary of this probability issue)

4.12.2 Stimulus/Response Sequences

1. System seeds the random generator.
2. System picks a random number from 1 to n number of candidates 100 times.
3. The candidate who is picked after 100 picks is determined the winner of the lottery.

4.12.3 Functional Requirements

REQ-11.1: See REQ-9.1, REQ-11.1

4.1.4 Use Case for Leveling a Tie by Lottery

Name	Leveling a Tie With Lottery
ID	UC_012
Description	If a tie between 3 or more candidates occurs, randomly select the winner in a

	lottery process via the computer.
Actors	System Engineers.
Organizational Benefits	Determine a winner between 3 or more candidates. By using a lottery, you eliminate bias produced by a fair coin toss between 3 or more candidates.
Frequency of Use	Zero to many times during an election.
Triggers	● In CPL, more seats given than candidates, how to assign extra seats remaining ● In CPL, if there's a tie in allocating a seat ● In IR not all candidates have been ranked in ballot, and no clear majority ● In IR the lowest number of votes redistributed.
Preconditions	An election computation must be in process and there must be a clear (discrete) tie between 3 or more candidates.
Postconditions	A clear winner is determined between 3 or more candidates.
Main Course	1. System seeds the random generator. 2. System picks a random number from 1 to n number of candidates 100 times. 3. The candidate who is picked after 100 picks is determined the winner of the lottery. (triggers UC_006)
Alternate Courses	None
Exceptions	The random process does not have enough randomness after 100 coin tosses: Increase the amount of flips until we eliminate bias. We must be sure not to affect time restraints.

4.13 Finalizing Audit File

4.13.1 Description and Priority

After election data has been processed and a winner has been determined, an audit file outlining the process of how the winner was determined as well as the final result will be finalized and converted to a read only file .txt file.

4.13.2 Stimulus/Response Sequences

1. Election calculations are written to the file. (triggers UC_006)
2. The audit file is changed to read-only.
3. The audit file is closed.
4. File name/path is displayed to the user

4.13.3 Functional Requirements

REQ-13.1: The finalized audit file should be read-only .txt file (Elicited Requirement)

REQ-13.2: You will have programmers, testers, and election officials running and using this program. The results of the election could be shared with media personnel.

REQ-13.2: See REQ-5.1, REQ-5.2, REQ-5.3, REQ- 5.4

4.13.4 Use Case for Finalizing Audit File

Name	Finalize Audit File
ID	UC_013
Description	After election data has been processed and a winner has been determined, an audit file outlining the process of how the winner was determined as well as the final result will be finalized and converted to a read only file .txt file.
Actors	System Engineer.
Organizational Benefits	Produce a finalized read-only file that describes the process and calculations of the election that are computed by the system.
Frequency of Use	Once per election
Triggers	Election has finished being processed and a winner has been determined.
Preconditions	Audit file has been created. It is updated with the result and all the information of how the election progressed.
Postconditions	Election officials are able to audit the election process via the finalize read-only audit file.
Main Course	1. Election calculations are written to the file. (triggers UC_006) 2. The audit file is changed to read-only. 3. The audit file is closed. 4. File name/path is displayed to the user
Alternate Courses	None.
Exceptions	Program is unable to change permissions: The user will have to adjust operating settings to ensure proper function of the system.

4.14 Display Winner and Percentage

4.14.1 Description and Priority

Display to the screen the winner(s) and information about the election (type of election, number of seats, number of ballots cast, winners, number of votes received, percentage of votes received). This also serves the users to verify that the calculations were successfully completed.

4.14.2 Stimulus/Response Sequences

1. The system prints out the information about the election.
2. The system notifies the user that the program is done running and allows the user to exit the program.

4.1.3 Functional Requirements

REQ-14.1: You should display to the screen the winner(s) and information about the election (e.g. type of election, number of seats). You do not need to show an audit of the votes but showing the number of ballots cast, the winners and the stats for everyone such as number of votes received, the percentage of votes received, the winner(s), etc.

REQ-14.2: See REQ-2.8

4.14.4 Use Case for Displaying Winner and Percentage

Name	Display Winner and Percentage
ID	UC_014
Description	Display to the screen the winner(s) and information about the election (type of election, number of seats, number of ballots cast, winners, number of votes received, percentage of votes received).
Actors	Election officials, System testers, System Engineers
Organizational Benefits	Clear display of winner of election as well as statistics of election. This also serves the users to verify that the calculations were successfully completed.
Frequency of Use	Once per election
Triggers	Instant Runoff (IR) or Closed Party LIst (CPL) elections are finished with computations.
Preconditions	Program has been run, and the winner of the election has been determined, and the audit file has been finalized.
Postconditions	Visible display of the winner, type of election, and election results. The program is finished and can be exited.
Main Course	Type of Election is IR: Print to terminal: 1. Type of Election (IR)

	2. Number of ballots cast 3. Candidates 4. Number of votes and percentage for each candidate. 5. Winner
Alternate Courses	Type of Election is CPL: Print to terminal: 1. Type of Election CPL. 2. Number of seats. 3. Number of ballots cast. 4. Parties and their candidates. 5. Number of votes per party. 6. Number of seats allocated to each party. 7. Elected candidates of each party.
Exceptions	Something happens to the finalized read-only audit file prior to this use case. We may consider doing a portion of this in parallel to reduce the chance of this happening.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The program will be able to run 100,000 election ballots in under four minutes. In the case of a tie the program will flip a coin as many times as necessary in order to determine the winner in a fair manner. The runtime of the coin toss is included in the total runtime of the program. In the case of a tie between three or more candidates, the winner will be randomly selected in a lottery process via the computer. Leveling a tie with a lottery system is included in the total runtime of the program.

See UC_011, UC_012

5.2 Safety Requirements

The program does not have any safety requirements. Any safety concerns are the responsibility of the election officials.

5.3 Security Requirements

Security such as ensuring one vote for one person is handled at the voting centers. There are no other security requirements and thus any user can use it without any additional privileges. Any security concerns are the responsibility of the election officials. The input file will only be read by the program, there will be no changes to the election results.

5.4 Software Quality Attributes

There is one program designed to run both closed party listing and instant runoff elections. The election files given to the program will come in a predetermined format and the structure cannot be changed outside of the program. In addition to this the election files provided do not allow write in candidates and so the program does not account for it. The program will produce an audit file which follows the order of the ballots being assigned to the candidates. The audit file will show the order of removal of candidates in instant runoff elections and what ballots were redistributed. The program can be reused for every closed party listing and instant runoff election.

5.5 Business Rules

The program will be run and used by programmers, testers, and election officials. The results of the elections will be shared with the public including but not limited to media personnel, candidates, and the general public.

6. Other Requirements

There will be future iterations on this system. Readable, commented and refactorable code will help everybody in the long run.

Appendix A: Glossary

CSV File: A comma-separated value file, or CSV file, is a text file that follows a specified format where commas separate each value. Often these files also separate each data record onto a new row using a newline character.

CPL: Closed party listing voting, or CPL, is a proportional voting type. For this type of voting each party provides a list of candidates to fill the seats they are given after the election. Voters then vote for the party instead of the individual candidate. Once votes are cast the seats are proportionally split between each party based on the percentage of votes that they received. Candidates from each party are then appointed based on their position on the provided candidate list.

IEEE: Institute of Electrical and Electronics Engineers, or IEEE, is a professional group known for being a leader in development of industry standards for many engineering and technology related areas.

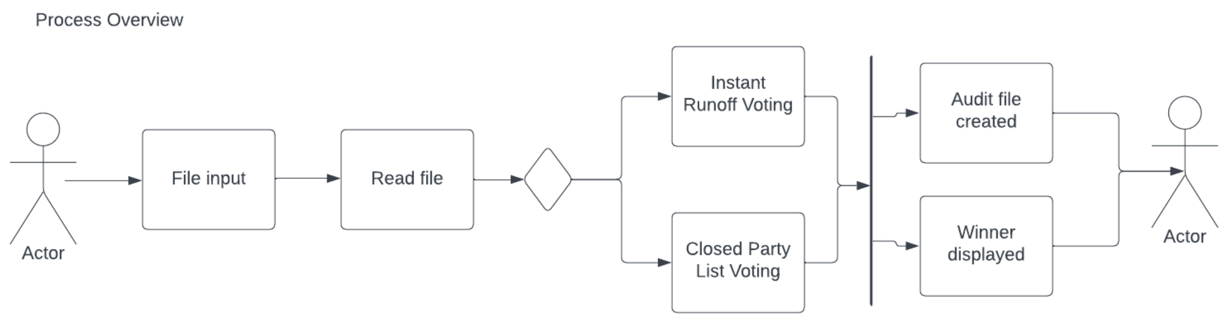
IR: Instant runoff voting, IR or IRV, is a plurality or majority type voting system. For this type of voting all candidates appear on the ballot. Voters then indicate their order of preference for each candidate, but are not required to rank all candidates. The winner is then calculated by determining if any candidate has a majority once votes are counted. If there is no majority the candidate with the least votes is eliminated, and any votes for that candidate move to the next ranked candidate indicated on the ballot. This process continues until there is a majority, and a winner is declared.

Majority: A majority in IR voting is considered to be over 50% of the total number of votes.

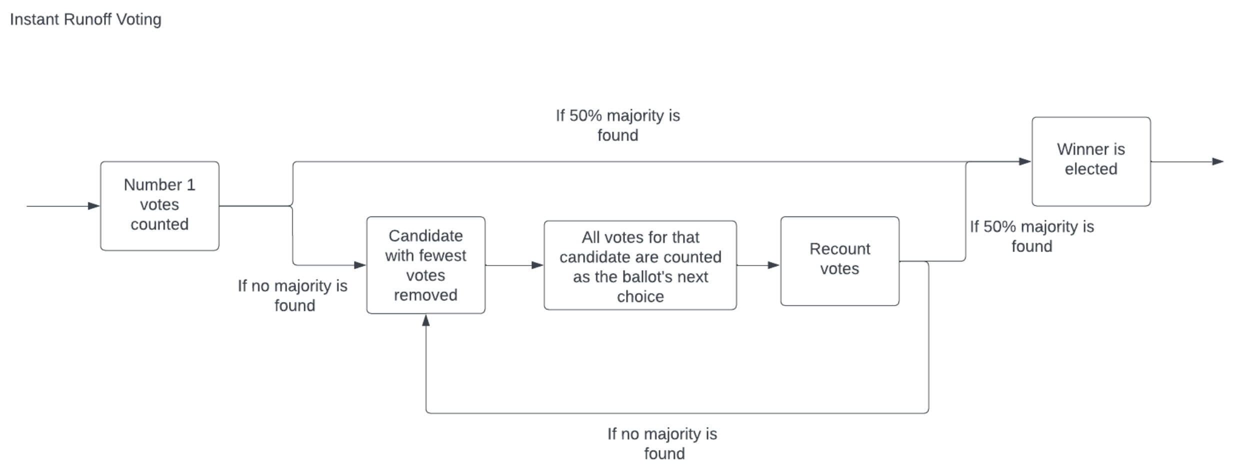
TBD: TBD, meaning To Be Determined, means that the specified issue has yet to be resolved.

Appendix B: Analysis Models

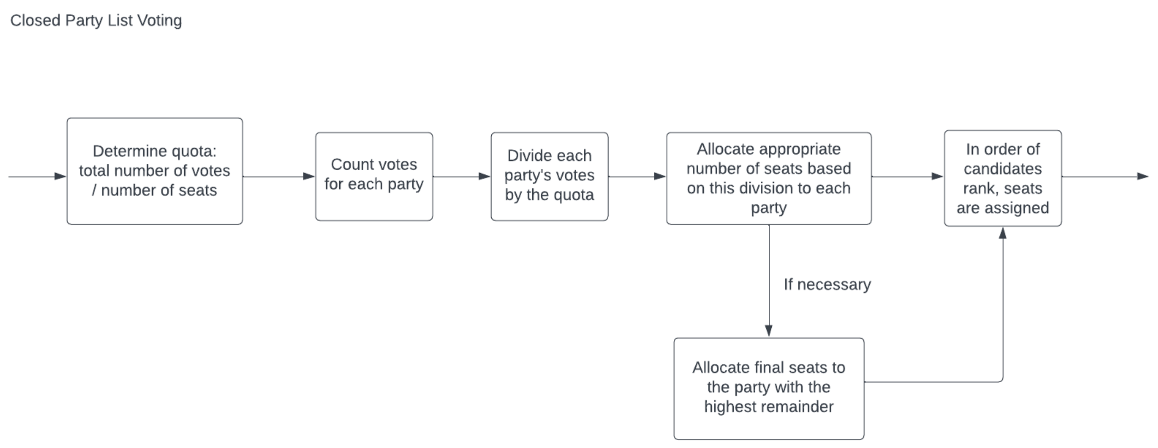
Appendix B.1



Appendix B.2



Appendix B.3


Closed Party List Voting
Determine quota: total number of votes / number of seats
Count votes for each party
Divide each party's votes by the quota
Allocate appropriate number of seats based on this division to each party
In order of candidates rank, seats are assigned
If necessary
Allocate final seats to the party with the highest remainder

Appendix C: To Be Determined List

The requirements to this system have been established to the best of our ability and knowledge. The only processes that are to yet be determined are processes or issues we have yet to foresee. The Exceptions field of use cases UC_004, UC_008, UC_009, UC_010 have been deemed TBD only because we have not been able to conceive reasonable exceptions for these cases yet. We do not imply that there are no exceptions to our system, only that we have not yet experienced them.