Software Requirements Specification

for

The Voting System

Version 8.0 approved

Prepared by Sidney Huong Nguyen, Chiemeka Nwakama,

Ted Casey, Phuong Lieu

x500's: nguy4257, nwaka013, casey526, lieu0009

Team #3

CSCI 5801 Software Engineer I, University of Minnesota

February, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Group	2/1	Intro. 1.1 and 1.2 and 2.7	1
Chiemeka	2/2	1.4, 2.3	2
Phuong	2/2	1.3, 2.2, 2.6	2
Sidney	2/3	2.1, 2.7	2
Ted	2/3	1.5, 2.4	2
Group	2/3	2.5	2
Phuong	2/7	3.4	3
Chiemeka	2/8	3.3	3
Sidney	2/8	3.2	3
Ted	2/8	3.1	3
Group	2/8	Use Cases	3
Chiemeka	2/9	5.5 + Use Case 12: Party List Voting	4
Sidney	2/9	Appendix A: Glossary	4
Phuong	2/10	5.1 + Use Case 1, and 2	4
Ted	2/10	5.4	4
Phuong	2/12	4.5 + Use Case 6, 12	5
Ted	2/12	4.2, UC_007 to UC_010	5
Chiemeka	2/13	4.3 Closed Party Voting feature,	5
Sidney	2/13	4.6 Display Results + Use Case 13 & 14	5
Group	2/13	Verify and revise use cases	6
Phuong	2/13	Use case 12, 13, 14	6
Chiemeka	2/13	Use cases, 3, 4, 11, 8, 9	6
Sidney	2/13	Use case 16, adding to Appendix A	6
Ted	2/14	4.1, 4.4, document formatting, use case revisions	6
Group	2/15	Edits, Finalized Use Cases	7
Chiemeka	2/15	Added CPL processing Use Case as well as audit file naming scheme use case	7
Phuong	2/15	Use case: update audit file, modify process IR and CPL audit file	7
Ted	2/15	UC_004, UC_006 rework	7
Group	2/16	Review	8

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of an Election Voting System that supports both the instant runoff and closed party list voting systems, and produces the current election information. It will describe the purpose of the voting system, the various interfaces, what the system will do, the constraints by which it must operate and how the system will respond to external stimuli. This document is intended for election officials, the system programmers and testers, and any other individuals who may have permission to view the results of the system as well as the progression that lead up to that result.

1.2 Document Conventions

This document was based on the Institute of Electrical and Electronics Engineers (IEEE) template for System Requirement Specification(SRS) Documents.

1.3 Intended Audience and Reading Suggestions

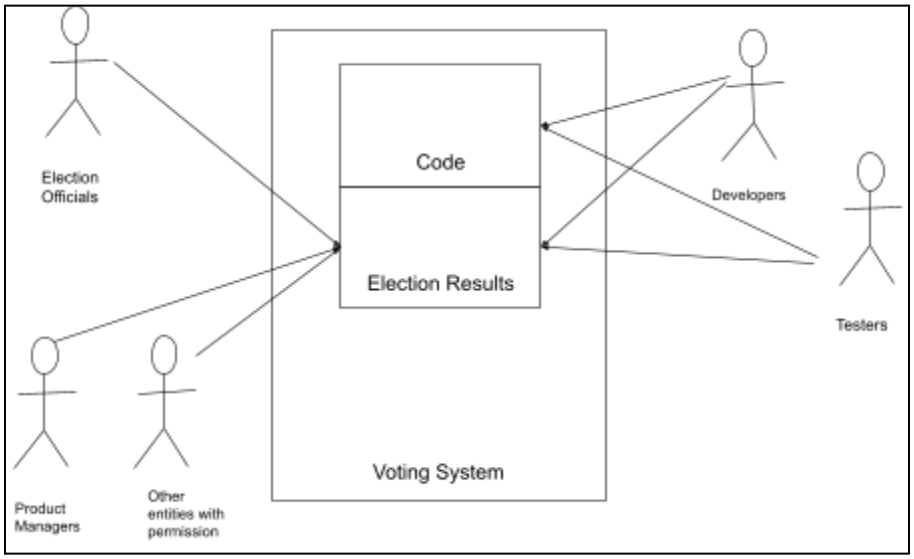

Figure 1 - System Environment

In this SRS documentation contains all of the information that is about the Voting System software that will be used during elections. The Voting System has 3 users that will be running and using the system. The programmers and testers will have access to all of the Voting System including the election results and the running software code. As they are designing and implementing the system, they will need to read all of the SRS reports and need to focus on the Overall Description, External Interface Requirements, and System Features. Election officials and other future individuals that have permission to view the result must read the External

Interface Requirements of 3.1, 3.3, and 3.4, all of section 4 System Features. and Other Nonfunctional Requirements of 5.2, and 5.3.

1.4 Product Scope

This Voting System is a software that election officials will be able to use to take in a file containing all ballots, and return the calculated results of either an instant runoff election or a closed party list election.

This software will benefit elections and election officials since it will allow a way to quickly produce current and updated results. It currently supports two different voting systems, with the capability to seamlessly introduce more in the future. The results of the system can be used mediawise for the general public. The participating parties and candidates can use election results generated by the system for future elections.

1.5 References

Additional Information on Instant Runoff Voting or Ranked Choice Voting
https://fairvote.org/our-reforms/ranked-choice-voting/

Additional Information on Party Listing and Closed Party Listing
https://www.electoral-reform.org.uk/voting-systems/types-of-voting-system/party-list-pr/

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

Java Download
https://www.java.com/en/download/

2. Overall Description

2.1 Product Perspective

During elections, there are different types of voting systems and algorithms to compute the current results. The Voting System was developed as a tool to be used for two different types of voting systems and is capable of calculating the results of the election. While there are many types of voting algorithms, this particular system is a new self contained product. However, it has the potential to be a replacement for existing similar systems. The Voting System is independent of the actual voting ballots, and it is a part of a larger system in that it will be used by the election officials to keep track of the current state of the election. It is a component in the election process, but can function independently. It was developed for systems that support Java SE 19.02 and higher.

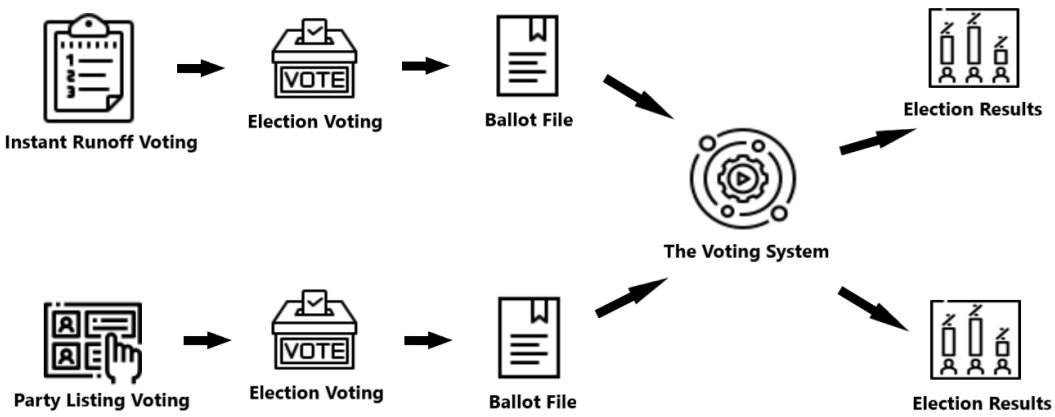

Figure 2 - Product Process Diagram

2.2 Product Functions

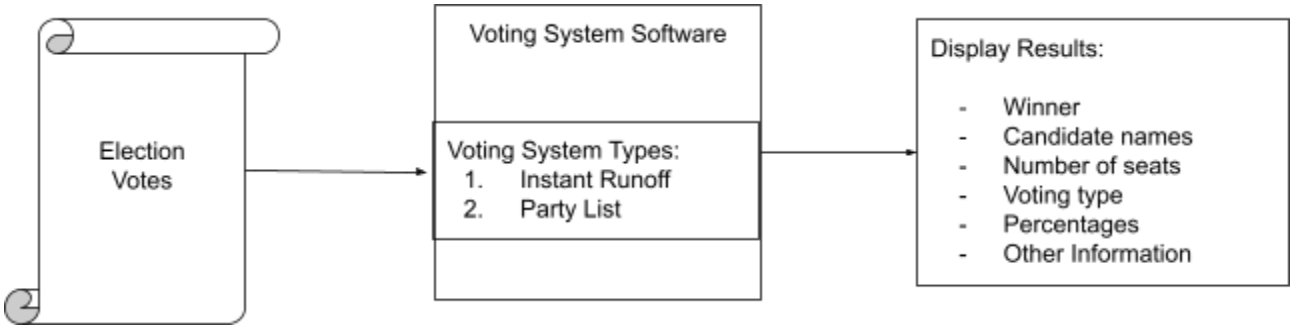

Figure 3 - Voting System Process

The Voting System will take in a provided valid election votes file, where the user will choose which type of voting system they want to perform. Then, the software will run and analyze the voting, stored information into the system, and calculate the election result. When finished, the result of the election will be displayed on to the screen where the user can see all of the information. The diagram is for preference of how the system will be running.

2.3 User Classes and Characteristics

Typical users such as Election officials who will be mainly interacting with this voting system on a basic level. Most the processes will be automated for them with the ones that are not, easily manageable in a user friendly interface. They will bring in the file and then collect the results.

Testers are the programmers interested in making sure that the voting systems used are fair and accurate. Testers are mainly concerned with being able to make sure that the system does not have any bugs via many tests and simulations.

Programmers are the individuals both designing, testing, and implementing the system

2.4 Operating Environment

The primary operating system for the Voting System is Linux. The codebase will be published to Github.

2.5 Design and Implementation Constraints

The Voting System is designed with the Waterfall process model and is developed in Java. The program's performance requirements are as follows. Its performance and functionality are required to operate on the University of MInnesota CSE lab machines. The program must be able to run 100,000 ballots in under 4 minutes. Currently, this specification document is written only in English with no language substitutions.

2.6 User Documentation

All users are expected to be Internet literate and be able to do simple tasks on a computer. Such as using a mouse to open an app, download software, pull up/down menus, and similar tools.

For election officials and other individuals with permission, please visit section 1.5 for information about the regulation in voting systems associated with this software.

Testers and Programmers please look through sections 3, 4, and 5 for clarification on requirements. Note that it may be helpful to understand information about current voting systems, which can be found in section 1.5.

Additional questions about interface, please look at section 3.1, 3.3, and 3.4.

For other questions, please look at content 5, 6, and the Appendix.

2.7 Assumptions and Dependencies

The Voting System was developed in Java, and therefore requires that Java be installed on your machine. The Voting System requires Java SE 19.02 or later and this applies to Windows, Linux, and Mac systems. This system assumes the actual voting has already occurred and that the input to the system is a compiled file of the ballots. There will only be 1 file provided for each election. This file is to be provided by an election official and its structure cannot be changed. In addition, the system assumes this file contains no errors to the file itself and the ballots. It is assumed there are no changes to the regulation of the Instant Runoff system and the Closed Party

listing system. It is assumed that the input ballot file contains at least 1 ranked official and independent groups in the closed party listing voting are named differently. There is currently no write in for candidates for this system. If there are errors in the input ballot file, it will affect the system functionality and affect the computed election results.

3. External Interface Requirements

3.1 User Interfaces

The voting system will have a simple text interface. There will be a terminal, when the program is started, to input a properly formatted CSV file. The UI will also display the election results when the program has finished.

3.2 Hardware Interfaces

The minimum hardware requirements for the Voting System software are 1190 Megahertz CPU and 128 megabytes of RAM. Though the software was developed for the Linux operating system, it will run for Windows 64-bit, Linux 64 bit, and Mac 64 bit. Since the software must be downloaded from the internet webpage, all hardware required to connect to the internet will be necessary.

3.3 Software Interfaces

The Voting System requires Java, preferably the latest version of Java SE 19.02. For more information, reference section 2.7 of this document.

3.4 Communications Interfaces

Users need to have the latest version of Java installed on their laptop or computer. See section 1.5 References for download. The app will be installed by the Programmers, and the result will be displayed to screen right after the program is running, so no other requirements are needed. Contact the software process manager or team if the runtime is longer than 4 minutes.

4. System Features

4.1 Interacting With the System

4.1.1 Description and Priority

The user will interact with a simple text interface to use the program. They will upload a voting file to the program, which will be checked for formatting errors and then input into its respective voting algorithm.

4.1.2 Stimulus/Response Sequences

After launching the program, the user will be prompted to input a voting file.

4.1.3 Functional Requirements

REQ-1: There will be a place for the user to upload the vote file
REQ-2: The file will be checked for errors, i.e. file type, formatting, etc.
REQ-3: The file will then be input into either the IRV or CPL voting algorithm depending on which vote type it is.

4.1.4 Use Case: Start program

Name	Start program
ID	UC_001
Description	User opening the Voting System software app to run.
Actors	Election Judges, Testers, Programmers, and Organizations with permission.
Triggers	Opening the software app.
Organizational Benefits	Election Judges: be able to get the result of the election fast. Organizations with permission: use the results for future usage. Testers and Programmers: make sure the program is running correctly/no errors.
Frequency of Use	Whenever the user uses the software.
Preconditions	Assume the software app already installed into the operating system (Windows, Linux, or Mac).
Postconditions	The program is able to open without errors.
Main Courses	Users use the software app to run election results.
Alternate Courses	1. Actors may choose to use another voting system instead of the one provided to them. 2. The program cannot open due to the operating system not up to date.

Exception	Actors can choose to close the software app at any time.

4.1.5 Use Case: Prompt user for information

Name	Prompt user for information
ID	UC_002
Description	The name of the ballot file must be given either by user prompt or by command line.
Actors	Election officials, Testers, and Programmers,
Triggers	The program is run either with the command line argument or without.
Organizational Benefits	Election officials: They may not know how to use the command line, thus the user prompt is convenient for them.. Testers and Programmers: They are able to test that the election officials can provide the file name to the program in one way.
Frequency of Use	Election officials: once per election. Testers and Programmers: Every time when running the program for testing and implementation
Preconditions	1. A file is provided by an election official in the directory 2. The user has read this document and understands the file name is given by prompt or command line 3. The user knows how to operate the system
Postconditions	File is identified
Main Courses	If user enters file name by prompt: 1. User starts program 2. User is given prompt to enter file name 3. User enters file name 4. See use case 4.1.6 If user uses command line: 1. User starts program with file name in command line 2. See use case 4.1.6
Alternate Courses	1. Users enter the wrong command. 2. Terminal ended with the wrong command given.

Exception	Users decide to close the terminal, which program will not get to run.

4.1.6 Use Case: File identification - file exist

Name	File identification - file exist
ID	UC_003
Description	The file will be attempted to be brought in and identifying the election file. If this is unable to be done, the system will show an error to the user where the user will be prompted again for a file.
Actors	Election officials, Testers, Programmers
Triggers	Attempt to bring in the election file to be identified
Organizational Benefits	The organization benefit to this is to allow election to be properly primed for ran on either the CPL or IR algorithms
Frequency of Use	Every election and during testing as well
Preconditions	Election file is searched for identification
Postconditions	The election file is able to be found and gets moved on for processing and formatting (UC_005/UC_004)
Main Courses	1. At the end of voting, the file is brought into the programs directory 2. The Election official or tester is prompted to put in the file name 3. Identification of the file is attempted 4. file is properly identified and moved on for reading and formatting (UC_005/UC_004)
Alternate Courses	1. At the end of voting, the file is brought into the programs directory 2. The Election official or tester is prompted to put in the file name 3. Identification of the file is attempted 4. Identification fails
Exception	Testers may be able to bypass this process, running simulated elections without needing to input a file If file identification fails, an error will display allow for a different

	file name to be entered *The election file will be of the CSV format. Files of any other type will raise an error.

4.1.7 Use Case: Processing Instant Runoff Voting (IRV) Vote File

Name	Processing IRV Vote File
ID	UC_004
Description	Determine the voting file is IRV and prepare it to be analyzed by the IRV algorithm,
Actors	Election Judges, Testers, Programmers, Organizations with permission.
Triggers	User inputs the file either through the command line or the interface.
Organizational Benefits	Election Judges: be able to get the result of the election fast. Testers and Programmers: testing the input file is correct.
Frequency of Use	Whenever a voting file is input
Preconditions	Assume the file is valid and contains enough information to run the program.
Postconditions	The file was read in correctly, and all of the contents are extracted for the program to use.
Main Courses	1. Open the file. 2. Check if the file is open successfully. 3. Start to read the file. 4. Read the first 4 lines of the file to determine a. vote type is “IRV” b. number of candidates c. Candidate party affiliations d. Number of ballots 5. Set a breakpoint when reading to the end of the file. 6. Close the file. 7. Input into IRV algorithm UC_006
Alternate Courses	AC1. If vote type is not IRV, instead process as CPL (UC_005)

Exception	EX1. If the first 4 lines do not provide voting information, return an error. EX2. If the vote file is the wrong file type, return an error

4.1.8 Use Case: Processing CPL vote file

Name	CPL Processing of the CSV file
ID	UC_005
Description	Once the first line is read, the system will know that the csv file is CPL and read in the data in a way to conforms to the formatting that has been indicated by the customer in the write up
Actors	Election Judges, Testers, Programmers, Organizations with permission.
Triggers	Once a file is identify and the first line reads “CPL”
Organizational Benefits	Election Judges: Process is automated allowing for human error to be eliminated from the equation. Allows votes to be counted much faster though the system then hand counting would allow. Testers and Programmers: testing the input file is correct as well as the method of entering the data from csv into the system is correct
Frequency of Use	Whenever a CPL election is ran
Preconditions	File is properly identified Assume the file is valid and contains enough information to run the program (No errors)
Postconditions	The file’s contents were read into the system correctly, and all of the contents are extracted for the program to use in the CPL algorithm (see UC_008)
Main Courses	1. Open the CSV file. 2. Read the first line of file to see is it is “CPL” or “IR” (already done) 3. Switch to CPL formatting knowing that what the next lines will be 4. Read in the second line which is the number of parties and store it 5. Read in the third line which has what the parties are separated by commas. Split by commas and individually store each party in variables (or an array)

	6. Read in the all of the following candidate lines and store them 7. Read in the line after all of the candidates which is the number of total seats and store it. 8. Read in the line after which tell us the number of votes/ballots we have and store it. 9. Based on the number of ballots and seats there are, number will be calculated to be used later in the party list algorithm that will determined the distribution of seats via division by that number (see UC_008) 10. Based on the number of ballots, the ballots will be read in up until that number taking note of the position of the 1's which will directly correspond to a correlated parted based on the order in which the parties were read in on back during step 5. 11. Once all lines/contents of the csv election file are read, the file will be closed and the system will move on to using this new info to run the election using the CPL algorithm (see UC_008)
Alternate Courses	1. Open the CSV file. 2. Read the first line of file to see is it is "CPL" or "IR" (already done) 3. Switch to IR formatting knowing that what the next lines will be 4. Starts reading in info in the IR format (see UC_004)
Exception	If file is unable to be open for any reason, an error will be thrown

4.2 Instant Runoff Voting

4.2.1 Description and Priority

This feature will run when the election is Instant Runoff Voting, and it will produce a winner.

It is of high priority, as it must run accurately every time.

4.2.2 Stimulus/Response Sequences

Run when a vote file is input with "IRV" at the top.

4.2.3 Functional Requirements

REQ-1: Automatically run when a file is recognized to be IRV.

REQ-2: Immediately declare a winner if a candidate receives over 50% of the votes

REQ-3: Eliminate a candidate and redistribute their votes if there is no clear winner or a tie.

REQ-4: If a candidate is eliminated, the feature should then repeat, checking for a winner or eliminating another candidate.

REQ-5: When voting, voters must rank at least one candidate and we do not allow write-in candidates.

4.2.4 Use Case: Instant Runoff Voting (IRV) Algorithm

Name	Instant Runoff Voting Algorithm
ID	UC_006
Description	The IRV algorithm will decide the winner of the election from an input voting file. If a candidate receives over 50% of the vote they are automatically declared the winner, otherwise the candidate with the fewest votes is eliminated and their votes redistributed. This process will repeat until a winner is declared.
Actors	Programmers, Testers, and Election Officials
Triggers	The input vote file is an IRV vote and is input into this algorithm by the system.
Organizational Benefits	It will quickly provide the winner of the election without the need for a second runoff vote.
Frequency of Use	Whenever an IRV election is conducted.
Preconditions	The election file is properly submitted and the vote type is IRV.
Postconditions	If there is a clear winner: The results should be visible to the user. An audit file is produced containing the details of the algorithm's process. If there is no clear winner: repeat this process after eliminating a candidate and redistributing their votes or move to either UC_008 or UC_009 if there is a tie.
Main Courses	1. If the vote is Instant Runoff, this algorithm will run and parse the votes. 2. The algorithm will determine which candidate received the fewest votes. 3. It will then redistribute that candidate's votes to each voter's next choice and eliminate them from the election.

	4. Steps 2-4 will repeat until a candidate receives over 50% of the vote (AC1).
Alternate Courses	AC1. If a candidate receives over 50% of the votes they automatically win, and move to UC_012. AC2. If there is a tie between two candidates, move to UC_009 to resolve the tie. AC3. If there is a three or more way tie between candidates, move to UC_010.
Exception	EX1. If no clearly majority due ballots being thrown out, candidate will be elected by popular vote (See UC_007) EX2. The votes are improperly entered A. The Program will display an error to the user. B. The algorithm will exit and return to UC_002

4.2.5 Instant Runoff No Clear Winner

Name	Instant Runoff No Clear Winner
ID	UC_007
Description	Due to how IRV voting is conducted, voters may choose only one candidate. If that candidate is eliminated, due to said voters having not indicated another candidate to redistribute their votes to, their votes are eliminated along with the candidate. This may result in no candidate receiving over 50% of the vote, causing the algorithm to pick the candidate with the highest percent of votes to win.
Actors	Programmers, Testers, and Election Officials
Triggers	All except 2 candidates are eliminated from the voting process, but neither candidate has over 50% of the votes.
Organizational Benefits	It will quickly provide the winner of the election without the need for a second runoff vote.
Frequency of Use	Whenever an IRV election is conducted, no candidate receives over 50% of the votes.
Preconditions	1. An IRV election is conducted 2. No candidate receives over 50% of the votes
Postconditions	There will be a winner declared

Main Courses	1. The algorithm will compare both candidates' final vote tallies. 2. The candidate receiving the most votes will be declared the winner
Alternate Courses	AC1. If there is a tie between the two candidates, move to UC_009 to resolve the tie.
Exception	EX1. The votes are improperly entered A. The Program will display an error to the user. B. The algorithm will exit and return to UC_002

4.3 Closed Party List Voting

4.3.1 Description and Priority

Users can access this system of voting by uploading a ballot file that has the “CPL” marker on the top of it. This feature is a high priority feature since it is one of the two major voting systems that our voting system supports. It along with Instant Runoff (see above in 4.1) must be functional in order for our system to complete the requirements as detailed in this document.

4.3.2 Stimulus/Response Sequences

The main stimulus for triggering Close Party List Voting is by the mode being identified during the identification use case by reading in the “CPL” marker on the top of the ballot file. Once this occurs, the system will start performing the Closed Party List Voting algorithm, save steps/results to an audit file, and allow for results to be displayed at the end.

4.3.3 Functional Requirements

REQ-1: Should automatically run when a file is recognized to be CPL

REQ-2: Should distribute won seats in a hierarchy of the listed order of candidates in each party. This order will be determined by each party privately. Candidates will be listed in order of priority on the official ballot.

REQ-3: Candidate seat allocation should be determined by how many total votes a party receives divided by a determined quota quantity. The number that is not a remainder shall be granted to parties that earned the votes

REQ-4: The reminder after this “largest remainder formula” is implemented should be determined based on the parties with the biggest remainders and how many seats are left. A seat should be awarded to party based on biggest remaining votes in order of amount

REQ-5: Once this process is completed, a seat should be awarded to candidates based on order selected by candidates on how many seats were given out to a party based on the previously explained method (see REQ-3 and REQ-4)

REQ-6: If there are more seats awarded to a party than candidates, then extra seats shall be awarded to candidates based on a randomly selected pool for as many extra seats that there are.

4.3.4 Use Case: Closed Party List Algorithm

Name	Closed Party List Algorithm
ID	UC_008
Description	The CPL algorithm is one of the two voting systems supported by the voting system. In this algorithm, ballots consist of parties that an individual can vote for. Once votes are tallied, depending on the total number of seats that are available, votes for each party will be divided by a particular value (to be determined). From this division, the whole number of seats will be granted to each party. Afterwards, the remaining seats will be granted, in order, to parties with the highest remainders. After votes are given to parties, candidates will be given seats based on their hierarchical listing on the ballot.
Triggers	Once the proper voting system is identified in the file "CPL" (see UC_003)
Actors	Election Judges, Testers
Organizational Benefits	Will allow for the ability to for the customer election judges and polling stations to perform automated partylist voting using our voting system that works properly without bias or other errors This will guarantee the integrity of the election.
Preconditions	The election file is properly submitted and the vote type is CPL.
Frequency of Use	Since there is the option of having an election that is instant run-off voting as opposed to partylist, the partylist algorithm will only be used 50% of the time. Testers will, however, use this algorithm much more often than election judges to make sure it works properly. Testing the partylist algorithm will also occur around 50% of the time.
Postconditions	The CPL algorithm is completed
Main Courses	1. Votes are read in for each party (UC_005) 2. House seats are awarded to the parties based on division of total by a certain factor (to be determined) 3. Any remaining seats will be given to parties based on the greatest number of remaining votes after division

	4. These awarded seats will be awarded to candidates based on party determined order 5. Once this process is finished an audit file will be created with steps and results of the election
Alternate Courses	1. Parties have same amount of votes, depending on how many parties are involved, if only 2, a fair coin toss will resolve this tie (see UC_009) If more than 2 parties tie, parties will be pool selected to determine potential seating distribution disputes between parties (see UC_010) 2. More seats allocated to a party than candidates will have additional seats distributed via random pool selection between party member candidates (see UC_010)
Exception	Two parties receive an equal amount of votes - A fair count toss will be performed to distribute contested seats between the parties (See UC_009) More than two parties receive an equal amount of votes - Tied parties will be randomly selected in by pool to receive a contested seat (See UC_010) A party is given more seats than it has candidates on the ballot - Additional seats will be handed out to candidates by a randomly selected pool No Audit file is created 1. System notifies user of error 2. Return to Main Course step 1. 3. User reaches out to support if no information is displayed

4.4 Tie Breaker

4.4.1 Description and Priority

When a tie occurs in either the IRV or CPL algorithms, a tie breaker will be executed. If a two-way tie occurs, a coin flip will be used to determine the results of the tie. If a three or more way tie occurs, a pool selection will be used to determine the results of the tie. This feature is of medium priority, because it will be rarely used but when used must give all candidates equal chance of being selected.

4.4.2 Stimulus/Response Sequences

This feature will be used when a tie occurs in either the IRV or CPL voting process. It will then execute a coin flip or a pool selection to determine the results of the tie.

4.4.3 Functional Requirements

REQ-1: Should automatically run when a tie occurs

REQ-2: Should be able to execute either a coin flip for a two-way tie or a pool selection for a three or more way tie.

REQ-3: Should clearly detail the decision process in the audit file.

4.4.4 Use Case: Coin Toss

Name	Coin Toss
ID	UC_009
Description	Breaks a two way tie in either IR or CPL at any part of the process whether it be a tie between two parties in CPL when distributing seats to parties, or if candidates receive equal votes in IR. The candidate who received the least votes will be eliminated, and their votes will be redistributed. It will then either declare a winner or repeat this process.
Actors	Programmers, Testers, and Election Officials
Triggers	In IR: - If two potential eliminee candidates tie in votes In CPL: - If two parties tie in votes and there is dispute over a odd number of seats to be distributed
Organizational Benefits	Will guarantee the integrity of either election type is not impeded on
Frequency of Use	During testing will be used as well as potentially in any election ran in the system
Preconditions	The election file is properly submitted and there is some sort of two way tie (see triggers above)
Postconditions	Two way tie is resolved allowing for the continuation of either algorithm eventually resulting in a winner and seats allotment to be determined.
Main Courses	In CPL: 1. Two parties receive the same amount of votes and there is a

	disputed seat 2. A fair coin flip is performed to determine who gets the seat 3. the winner of the coin toss is granted the disputed seat 4. CPL algorithm continues like normal (see UC_008) In IR:
Alternate Courses	In CPL: 1. More than three parties receive the same amount of votes and there is a disputed seat 2. Tied parties are pool selected instead of a coin toss (see UC_010) In IR: 1. More than three candidates tie in votes in where one must be removed/kicked out is determined by pool selection instead of a coin toss (see UC_010)
Exception	In CPL: - If two parties tie, but there are no seats to be disputed (there is an even amount of seats in total to be given to the parties, allowing for no need for a coin flip) In IR: - If two candidates tie in votes but are not in danger of being eliminated in the current run-off round, there is no need for a coinflip as tie candidates have no influence on the candidate to be eliminated

4.4.5 Use Case: Pool Selection

Name	Pool Selection
ID	UC_010
Description	Breaks a tie of more than two entities in either IR or CPL at any part of the process whether it be a tie between parties in CPL when distributing seats to parties, or if candidates receive equal votes in IR. The candidate who received the least votes will be eliminated, and their votes will be redistributed. It will then either declare a winner or repeat this process.
Actors	Programmers, Testers, and Election Officials
Triggers	In IR:

	- If three or more potential eliminee candidates tie in votes In CPL: - If more than three parties tie in votes and there is dispute over a odd number of seats to be distributed - If there are more seats awarded to a party than candidates on the ballot
Organizational Benefits	Will guarantee the integrity of either election type is not impeded on
Frequency of Use	During testing will be used as well as potentially in any election ran in the system
Preconditions	The election file is properly submitted and there is some sort of three or greater way tie (see triggers above)
Postconditions	Three way tie is resolved allowing for the continuation of either algorithm eventually resulting in a winner and seats allotment to be determined.
Main Courses	In CPL: 1. Three or more parties receive the same amount of votes and there is a disputed seat 2. All tied parties are put into a pool and randomly selected to determine who gets the seat 3. the winner of the coin toss is granted the disputed seat 4. CPL algorithm continues like normal (see UC_008) In IR: 1. Three or more candidate tie in a run-off round and all candidates are in threat of getting out in that round 2. All tied candidate are put into a pool and randomly selected to determine who gets kicked and who gets to stay 3. The candidates closest to the desire number get to stay 4. IR algorithm continues like normal (see UC_006)
Alternate Courses	In CPL: 1. Two parties receive the same amount of votes and there is a disputed seat will be resolved with a coin flip 2. Tied parties are determining via coin flip selected instead of a pool (see UC_008) In IR: 1. Two Candidates tie in votes in where one must be removed/kicked out will be resolved with a coin flip

Exception	In CPL: - If three are more parties tie, but there are no seats to be disputed (there is an even amount of seats in total to be given to the parties, allowing for no need for a coin flip) In IR: - If more than three candidates tie in votes but are not in danger of being eliminated in the current run-off round, there is no need for a coinflip as tie candidates have no influence on the candidate to be eliminated

4.5 Auditing of Results

4.5.1 Description and Priority

This is an automated feature that will produce a text file (possible .txt file) for the result of the election. After the program is finished, then beside a display result, there will also be a text file result. The file will contain all the information that the user saw on the display. Addition to all the steps result in the choosing voting system. For example, the removal of candidates in IR and what ballots were redistributed. This feature will ensure that the user will be able to copy/extract the file from the program for other uses. Users can access this result file through the same directory as the program.

4.5.2 Stimulus/Response Sequences

1. The user starts the program with a provided file and chosen voting system
2. The program computes and writes the results of the election to an audit file

4.5.3 Functional Requirements

REQ-1: Appropriate voting system is given
REQ-2: Correct ballot file is given
REQ-3: The audit file is created with the election results inside.
REQ-4: The audit file will be accessed in the same directory as the program

4.5.4 Use Case: Create Audit File - IR

Name	Create Audit File - IR
ID	UC_011
Description	An audit file is created containing the generated results of the

	election Type of Voting, Number of Candidates, Candidates, Number of Ballots, calculations, how many votes a candidate had, etc),
Actors	Developer, election official
Triggers	The Voting System program is run
Organizational Benefits	A single file is produced containing the current election results
Frequency of Use	Every time the program is run, the voting system is IR.
Preconditions	The type of voting system must be IR when running the program and the ballot file must be processed.
Postconditions	An audit file for IR is produced
Main Courses	1. User runs the program 2. The program computes the election results which is stored in an audit file with IR style 3. Audit file is produced contains: a. Type of voting system: IR b. The number of candidates c. Candidates name, initial of the party d. Number of ballots in the file e. Candidates ranking f. The removal of candidates g. Ballots redistributions h. The winner of the election i. Any calculations or additional steps along the way
Alternate Courses	The user may choose a different voting system 1. Return to Main Course step 1. The user identified the incorrect ballot file 1. Return to Main Course step 2. 2. Choose the correct ballot file
Exception	The audit file was not produced 1. System notifies user of error 2. Return to Main Course step 1. 3. User reaches out to support if audit file is still not produced

4.5.5 Use Case: Create Audit File - CPL

Name	Create Audit File - CPL
ID	UC_012
Description	An audit file is created containing the generated results of the election
Actors	Developer, election official
Triggers	The Voting System program is run
Organizational Benefits	A single file is produced containing the current election results of CPL
Frequency of Use	Every time the program is run, and the voting system is CPL
Preconditions	The type of voting system must be specified as CPL when running the program and the ballot file must be processed.
Postconditions	An audit file for CPL is produced
Main Courses	1. User runs the program 2. The program computes the election results which is stored in an audit file with CPL style 3. Audit file is produced contains: a. Type of voting system: CPL b. The number of parties c. The parties separated by commas d. Candidates name in ranked order e. Number of seats f. Number of ballots in the file g. Only a single 1 will be placed in the position of the selected party h. Seats redistributions i. The party name with the amount of seats j. Immediately followed by candidates name k. Any calculations or steps made during the CPL election process
Alternate Courses	The user may choose a different voting system 1. Return to Main Course step 1. The user identified the incorrect ballot file 1. Return to Main Course step 2. 2. Choose the correct ballot file

Exception	The audit file was not produced 1. System notifies user of error 2. Return to Main Course step 1. 3. User reaches out to support if audit file is still not produced

4.5.6 Use Case: Update audit file

Name	Update Audit File
ID	UC_013
Description	An audit file is created containing the generated results of the election
Actors	Developer, election official
Triggers	The Voting System program is run
Organizational Benefits	A single file is produced containing the current election results
Frequency of Use	Every time the program is run, the result of the election is produced.
Preconditions	The type of voting system must be chosen when running the program and the ballot file must be processed.
Postconditions	An audit file for the choosing voting system is produced
Main Courses	1. User identifies the ballot file 2. User runs the program after an audit file created 3. New audit file will overwrite the old audit file 4. New informations of the choosing voting system will be written into the file 5. See UC_011 and UC_012 for each voting system audit file generated information.
Alternate Courses	The user may choose a different voting system 1. Return to Main Course step 1. The user identified the incorrect ballot file 1. Return to Main Course step 2. 2. Choose the correct ballot file
Exception	The audit file was not produced 1. System notifies user of error

	2. Return to Main Course step 1. 3. User reaches out to support if audit file is still not produced

4.5.7 Use Case: Naming/storage scheme of Audit file

Name	Naming/storage scheme of Audit file
ID	UC_014
Description	User did not specify how the audit file should be named or stored, therefore unless the user indicates otherwise, the audit file will have to be given a default name that properly indicates that it is an election audit file as well as a default place to store it. *The formatting of the audit file is to be determined; however, most likely the file will be a text or pdf as these are easily readable formats for if this file ever needs to be looked at for any reason.
Actors	Developer, Tester, Election official
Triggers	After the Voting system has be completed and the audit file is complete, the audit file will be saved using this naming scheme
Organizational Benefits	During testing, it will be clear where the audit file is saved and what it will be named, making it easy to see if the audit file is being produced correctly During elections, the audit file will have a predictable location and naming scheme, allowing for an easy way to know if an election has been successfully completed or not by mere presence of the file.
Frequency of Use	Every time the program is run, an election either IR or CPL is successfully completed.
Preconditions	An audit file is being produced
Postconditions	An audit file is successful saved using this naming scheme and specified location
Main Courses	1. An audit file is done being produced 2. The audit file is named the election data followed by the election type and ended with the word “election audit” by default 3. Audit file is saved to the same directory as the program by default

Alternate Courses	1. An audit file is done being produced 2. There may be possibly be a way to name and store the audit file elsewhere using user prompts
Exception	If an audit file fails to be produced, an error will arise before reaching use case *If the user provides a naming scheme and specified location for the audit file in the future, this use case may be modified

4.6 Display Results

4.6.1 Description and Priority

This is an automated feature of the Voting System. Once the program finishes computing the results and generating an audit file, it will display to the interface or screen, the election winner and results. Note that it will not display the audit of the votes, and there may be information present in the audit file that will not be displayed. This feature is of medium priority, as the results are also contained in the audit file. The displayed results are a more convenient way for the user to analyze the results, rather than opening another file.

4.6.2 Stimulus/Response Sequences

3. The user starts the program with a provided file and chosen voting system
4. The program computes and writes the results of the election to an audit file
5. The program displays the results of the election to the screen

4.6.3 Functional Requirements

REQ-1: Appropriate voting system is given
REQ-2: Correct ballot file is given
REQ-3: The election results are correctly computed

4.6.4 Use Case: Display Election Results

Name	Display Election Results
ID	UC_015
Description	The election results have been generated and written to an audit file, the election winner and results are displayed to the interface.
Actors	Developer, election official
Triggers	The Voting System program has run, and an audit file has been produced containing the results
Organizational Benefits	The election results are displayed on the interface for easy readability and access
Frequency of Use	Every time the program is run, therefore, everytime the election official sees fit to update the election results
Preconditions	An audit file is produced with the results
Postconditions	The interface or screen displays the election winner and results
Main Courses	4. The program is run and generates the results 5. The results are written to an audit file 6. The program displays the winner and results to the interface
Alternate Courses	The user reads directly from the audit file 3. Return to Main Course step 1 and step 2 4. When audit file is produce, open file and view contents
Exception	Incorrect information is displayed 1. Return to Main Course step 1. 2. User reaches out to support if incorrect information still displays No results are displayed 4. System notifies user of error 5. Return to Main Course step 1. 6. User reaches out to support if no information is displayed

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Voting system requires a system with at least 1190 Megahertz CPU and 128 megabytes of RAM. Since the system can process up to 100,000 ballots in 4 minutes, it would require the operating system to be up to date. This includes Windows 64-bit, Linux-64 bits, and Mac-64 bits. In addition with high speed Internet capability.

5.2 Safety Requirements

N/A

5.3 Security Requirements

N/A

5.4 Software Quality Attributes

The software must evaluate 100,000 ballots in under four minutes and be reusable multiple times throughout a year. It must run accurately every time and its process will be detailed in an audit file to certify this. It should be created so that there is space to add features in the future and should be scalable to handle a large election. When the software is reused, the old audit file will be overwritten with the new election results.

5.5 Business Rules

Election officials must be trained to use the voting system. Voting system must be able to be run in multiple, separate independent elections. Each time there is a new election, the system must be reset as to convolute previous election votes with current ones.

Election officials are not permitted to open up the election file since this could lead to wrongful editing and tampering of votes.

This voting system will only be able to accept and process CSV files (see glossary for definition). Any other file type will not be accepted by the system and raise an error.

6. Other Requirements

N/A

Appendix A: Glossary

Term	Definition
Audit file	In this system, an audit file is a text based file containing the generated results of the election
Ballot	A device used to cast votes in an election
Ballot File, Election File	The CSV file that contains the details of the election and the votes. Note “ballot file” and “election file” refer to the same file.
Closed Party Listing Voting (CPLV or CPL)	A type of voting method where voters vote for one political party as a whole, rather than individual candidates. The voters do not have influence on which candidates are elected from the winning party.
CSV	Comma-Separated Values file
Election	A formal and organized choice by vote of a person for a political office or other position
Git	A distributed version control system that tracks changes in any set of computer files.
Github	An internet hosting service for software development and version control for Git
IEEE	An acronym that stands for Institute of Electrical and Electronics Engineer. A professional association for electronic engineering and electrical engineering with the mission of advancing technology for the benefit of humanity.
Instant Runoff Voting (IRV or IR)	A type of ranked preferential voting method

	that uses the majority voting rule, which is the principle that the group with the most supporters is chosen. Voters rank the election candidates in order of preference.
Java	A high level, class based, object oriented programming language that is designed to have as few implementation dependencies as possible. Its latest release is Java SE 19.02
Software Requirements Specification (SRS)	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. This document is a software requirements specification document.
User(s)	Election official, Developer, Tester
Version control	The practice of tracking and managing changes to software code. It is systems of software tools that help software teams manage changes to software code over time.
Waterfall Model	A sequential model that divides software development into pre-defined phases where each phase must be completed before the next phase can begin with no overlap.

Appendix B: Analysis Models

N/A

Appendix C: To Be Determined List

N/A