Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Noreen, Jon, Lane, Pyrenees

University of Minnesota – Twin Cities

16 February 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Team 18	2/16	Complete SRS	1.0

1. Introduction

1.1 Purpose

Voting officials require a robust voting tally system to automate the counting process for a given election type. The election types relevant to the system include Instant Runoff Voting and Closed Party List Voting. The implementation of these include candidate elimination, allocation of seats, potential tiebreaks, and the generation of an audit file as a record of the tally process.

1.2 Document Conventions

Every requirement statement has its own priority.

1.3 Intended Audience and Reading Suggestions

This document is intended for voting officials, system testers, and the development team. The rest of this document contains information concerning the implementation of the voting system, interface and nonfunctional requirements, and system features. The document is intended to be read in sequential order. Developers and testers may wish to pay special attention to the system features section, while users may find the overall description of the system helpful to reference when interacting with the product.

1.4 Product Scope

The system will calculate the results of an election by using a file containing all the requisite information. A record of the election will be displayed in an audit file. Benefits of this includes:

- The voting itself is independent from the calculation of the winners (low coupling is always good.)
- Voting officials no longer need to tally votes by hand.
- The vote counting process becomes less prone to human error.
- The results of an election would be decided by unbiased software.
- Results of elections may be less prone to uncertainty.

1.5 References

Resources used to write this document includes:

- “Project1_Waterfall_VotingSRS_Spring2023.docx” found on the class Canvas page
- “CSCI5801_Spring2023_SRS_Rubric.pdf” also found on the Canvas page
- “srs_template-ieee.doc”
- “Example-of-Use-Cases.jpg” used as reference for the use cases

2. Overall Description

2.1 Product Perspective

Voting System v1 is a new, self-contained system designed from scratch to assist voting officials with the task of processing election data.

2.2 Product Functions

User Interactions:

- Input an election file
- Provide information when prompted

Voting System Capabilities:

- Generate an audit file
- Read in a file specified by the user
- Display the results of an election to the terminal
- Conduct Instant Runoff Voting
- Handle all edge cases related to IR Voting
- Conduct Closed Party Voting
- Handle all edge cases related to CPL Voting

2.3 User Classes and Characteristics

This system is designed to be used by voting officials and software testers. Voting officials will use this program for the purpose of processing voting data to get election results or produce an election audit file.

Software testers will use this program to find bugs within the program to improve the system and reduce risk of inadequate functionality.

2.4 Operating Environment

This software is designed to operate on University of Minnesota CSElabs machines – a Linux environment.

2.5 Design and Implementation Constraints

The program must be able to process 100,000 ballots in 4 minutes.

2.6 User Documentation

There are no user documentation components.

2.7 Assumptions and Dependencies

The Voting System is implemented with Java (JDK 17). The user must have Java installed on their system. The Voting System must run on CSELab machines. There are no software components that are intended for reuse.

3. External Interface Requirements

3.1 User Interfaces

The system will be operated via a terminal, which will prompt the user for any information that the voting file does not have. Lastly, the prompt will ask the user for the input file. Once given, the election calculations will run, and the information will be displayed in an audit file and on the screen.

3.2 Hardware Interfaces

The system runs on CSELab machines. A full list of CSELab machines with their respective hardware specifications is linked below.

https://cse.umn.edu/cseit/classrooms-labs

3.3 Software Interfaces

The software will be sure to work on a CSELab machine, which is of the Linux operating system (Ubuntu). This system will only need to interact with itself, using basic Java Input/Output to read from voting files and to write to audit files. Input shall be received via a .csv file. The voting data for the audit file shall be stored as a .txt file.

3.4 Communications Interfaces

Transferring of data from the data part of the system to the logic part will be done through basic Java.io.

4. System Features

4.1 Pass a File Into the Program

4.1.1 Description

A user can pass an election file (.csv format) into the program.

4.1.2 Stimulus/Response Sequences

1. The system prompts a user to input a file.
2. If the user chose to run the program with the filename through the command line instead, then the system accepts the file through this manner.
3. If there is an error in opening the file, the system notifies the user about the error with details. The system then re-prompts the user for the filename and returns the user to step 1, unless the user has already been prompted 3 times – in which case the program exits.

4.1.3 Functional Requirements

REQ-1: The user should be able to use either the command line to enter the name of the file when starting the program or enter the name of the file through a prompt from the system.

REQ-2: The system should validate whether the file can be accessed or not, such as whether the file has the appropriate permissions set.

REQ-3: In case of an error, the system should notify the user of the nature of the error and re-prompt the user for information. The system should exit if the user has been prompted 3 times already.

4.1.4. Associated Use Case

Name	Pass Election File Into the Program
ID	UC_001
Description	The user inputs the .csv file into the program to count votes for a given election.
Actors	Election official, testers
Triggers	The actor starts the program.
Preconditions	The file cannot have file errors. The file is within the working directory.
Postconditions	The file has been opened.

Main Course	1. The system prompts the user to enter the name of a file (see AC1). 2. The system takes the input and validates whether the file can be accessed (see EX1).
Alternate Courses	AC1: The user runs the program with the filename through the command line. 1. Return to Main Course Step 2.
Exceptions	EX1: Failure to open file. 1. The system notifies the user that an error has occurred with details about the error, including when an error occurs because the user entered the name of the file without the ".csv" file formatter at the end of the file name. 2. Reprompt user for the filename and return user to Main Course step 1. If the user has already been prompted 3 times, exit the program.

4.2 Determine Election Type

4.2.1 Description

The election type is specified via the first line of the input file. This line must be read and parsed to determine which data input process and election algorithm must be conducted.

4.2.2 Stimulus/Response Sequences

1. The user inputs the file into the program.
2. The system determines the election type by reading and processing the first line of the input file.

4.2.3 Functional Requirements

REQ-1: The input file has been opened by the program.
REQ-2: The input file has been processed by the program.

4.2.4 Associated Use Case

Name	Determine Election Type
ID	UC_002
Description	The system reads in the first line of the opened file and matches it to an election type.
Organizational Benefits	This allows the system to proceed with either IR or CPL voting in a streamlined process, making program flow easier to follow and allowing

	the program to be more efficient and logical.
Actors	Voting system, testers, election officials
Triggers	The file has been successfully opened.
Preconditions	The user has input a valid file.
Postconditions	The system executes election processing logic in accordance with specified election type.
Main Course	1. The system scans the first line of the open file. 2. The system places the first line into a String variable. 3. The system compares the String to the two election format specifiers. 4. The system proceeds with the rest of program logic in accordance with the string comparison.
Exceptions	None

4.3 Create an Audit File

4.3.1 Description

The system takes the input file and generates a file for auditing the vote tally with the title as the election type and date.

4.3.2 Stimulus/Response Sequences

1. The user inputs the file either through the system or passed into the command line when the program is run.
2. The system reads the type of election and generates an audit file to be written as the votes are tallied.

4.3.3 Functional Requirements

REQ-1: The input file must exist.

4.3.4 Associated Use Case

Name	Create an Audit File
ID	UC_003
Description	The system processes the header of the voting file to create an audit file named for the election type and date.

Organizational Benefits	The audit file can be used to ensure integrity of the election process, allowing for public trust in the Voting System.
Frequency of Use	Every election
Actors	Voting System, Election Official, Testers
Triggers	The system opens the user's file.
Preconditions	A user has input the file for processing.
Postconditions	An audit file is created to begin recording the vote tally process.
Main Course	1. The system reads the first line of the input file to determine the election type. 2. The system assigns the system date to date variable in program 3. The system creates an audit file named for the election type and the date of when the election occurred in the format, "MM_DD_YYYY_VOTINGTYPE.txt", where the VOTINGTYPE represents the type of voting for the election file processed and MM_DD_YYYY signifies the format of the election's date. This audit file is created to only be mutable during the writing process, and should be immutable with read-only permissions with the conclusion of the program.
Alternate Courses	None
Exceptions	None

4.4 Process File for CPL Election Data

4.4.1 Description

The system reads data from the CPL election file based on CPL file format specifications. The data is then stored efficiently, allowing for the CPL election algorithm to meet runtime requirements.

4.4.2 Stimulus/Response Sequences

1. The user inputs the CPL file.
2. The system processes the input file for data.

4.4.3 Functional Requirements

REQ-1: A valid .csv file is put into the system.
REQ-2: The first line of file is processed and matched with the CPL election specifier.

REQ-3: Handle errors in closing a file by displaying an informative message to the user and exiting out of the program.

4.4.4. Associated Use Case

Name	Process File for CPL Election Data
ID	UC_004
Description	The system reads in the number of parties, party identifiers, candidate identifiers, number of ballots, and all ballot information.
Organizational Benefit	The program stores the election data for easier processing.
Actors	Voting System, Testers
Triggers	The election type has been specified and matched with the CPL identifier.
Preconditions	The open file exists for reading.
Postconditions	All the data necessary to calculate CPL election results and create a valid audit file now exists in the program.
Main Course	1. The system reads in the number of contestants, contestant identifiers (such as contestant name and number of total ballots), and ballot information (such as the the vote(s) listed on each ballot based on specified format) and stores data in variables inside the program that can be referred to during the rest of the course of the program. 2. Once the end of the file has been reached, the system closes the file (see EX1).
Alternate Courses	None
Exceptions	EX1: Error occurs when closing file 1. The system prints out an informative error message and exits the program.

4.5 Conduct CPL Algorithm

4.5.1 Description

The system accesses vote counts for each party, determines a quota by dividing the total vote count by the number of seats to allocate, and allocates seats to parties based on their vote total divided by quota value.

If any seats remain, vote remainder is calculated for each party by subtracting the product of the total seats assigned and the quota from the total vote count. This remainder is then used to allocate remaining seats and assign them to parties.

If any party is unable to claim their full seat allocation, due to possessing fewer candidates than allocated seats, the seats that cannot be claimed shall be distributed via a lottery. See 4.6 for additional information.

4.5.2 Stimulus/Response Sequences

1. The user inputs the file.
2. The system determines the file is of CPL election type and runs the CPL algorithm.

4.5.3 Functional Requirements

REQ-1: The system must be able to process the input file.
REQ-2: The audit file must be named.
REQ-3: The election data must be present inside the system.

4.5.4 Associated Use Case

Name	Conduct CPL Algorithm
ID	UC_005
Description	The system allocates seats to parties based on the CPL election algorithm specifications.
Organizational Benefits	Adds CPL election functionality, allowing the system to meet CPL election handling requirements
Actors	Voting System, Election Official, Testers
Triggers	The input file is processed and the system determines the election type is CPL.
Preconditions	The audit file has been created, and all necessary data exists within the program.
Postconditions	All seats are properly allocated to their rightful parties.
Main Course	1. The total number of votes is divided by the number of seats to fill to obtain the quota. 2. The number of votes each party received is divided by the quota to determine the first allocation of seats (see AC1). 3. The remainder for each party is calculated, and the second round of seat allocation is determined by the number of seats left to fill and the parties which have the largest remainder of votes 4. The results are determined and the seats are properly allocated to the parties.
Alternate Courses	AC1: Two or more parties have the same number of allocated seats, and there are not enough seats to split equally.

	1. Break a tie between the involved parties until each party can get an appropriate number of seats according to the results of the tie break (see 4.9). AC2: A party is allocated more seats than they are able to accept. 1. Assign excess seats to remaining parties via lottery (see 4.6).
Exceptions	None

4.6 Assign Seats via Lottery

4.6.1 Description

In the event that a party wins more seats than candidates they have listed, all of the extraneous seats must be distributed via lottery. The lottery shall take in all parties except the party that originally won the seats and output a single winner. If multiple seats are to be distributed, multiple lotteries will take place.

4.6.2 Stimulus/Response Sequences

1. The user inputs a .csv file of CPL election type.
2. The system successfully performs CPL file data input and election algorithm.

4.6.3 Functional Requirements

REQ-1: The file has been successfully opened.
REQ-2: The file contains the CPL election specifier at the top of the file.
REQ-3: The seats have been allocated according to CPL election algorithm specifications.

4.6.4 Associated Use Case

Name	Assign Seats via Lottery
ID	UC_006
Description	The system assigns unclaimed CPL seats to remaining parties based on random lottery which is created via use of the tiebreaker function.
Actors	System, Voting official
Triggers	A party in a CPL election has fewer candidates listed than the number of seats that they win.
Preconditions	The main CPL election algorithm has been completed. The number of seats each party is entitled to and the number of candidates each party has is already known.

Postconditions	Any extra seats have been randomly assigned.
Main Course	1. The system calculates the number of excess seats. 2. All remaining parties are placed in the lottery. 3. The lottery operates until only one party is left. 4. The system checks if the party has enough candidates listed to accept the seat. 5. The system checks if any excess seats still need to be assigned – if yes the system repeats step 2.
Alternate Courses	AC1: The party that wins the lottery cannot accept the seat due to not having enough candidates. 1. The lottery is repeated but with all remaining parties. 2. In case of AC1 happening again, the system repeats the process until the seat is accepted with parties that cannot accept seats being eliminated from the lottery process with each iteration.
Exceptions	EX1: No party has enough candidates to fill the remaining seats. 1. The system returns an error message explaining the election could not be processed.

4.7 Process File for IR Election Data

4.7.1 Description

The system reads data from the IR election file based on IR file format specifications, and stores the data in the program. The data is stored in a way that allows the system to complete the IR election algorithm within runtime requirements.

4.7.2 Stimulus/Response Sequences

1. The user inputs the IR file.
2. The system processes the input file for data.

4.7.3 Functional Requirements

REQ-1: A valid .csv file has been opened.
REQ-2: The first line of file has been processed and matched with the IR election specifier.
REQ-3: Handle errors in closing a file by displaying an informative message to the user and exiting out of the program.

4.7.4. Associated Use Case

Name	Process file for IR Election Data
ID	UC_007

Description	The system reads in the number of candidates, candidate identifiers, number of ballots, and ballot information.
Organizational Benefit	The system stores the election data for easier processing.
Actors	Voting System, Testers
Triggers	The election type has been specified and matches the IR identifier.
Preconditions	The open file exists for reading.
Postconditions	All data necessary to calculate the election results and create a valid audit file now exist in the program.
Main Course	1. The system reads in the number of candidates, candidate identifiers (such as candidate name and number of total ballots), and ballot information (such as the the vote(s) listed on each ballot based on the specified format), and stores the data in variables inside the program that can be referred to during the rest of the program. 3. Once the end of the file has been reached, the system closes the file.
Alternate Courses	None
Exceptions	EX1: Error occurs when closing the file. 1. The system prints out an informative error message and exits the program.

4.8 Conduct IR Algorithm

4.8.1 Description

While the number of candidates is greater than two, and no candidate possesses a majority vote total, the algorithm follows a recursive process. The total votes are tallied, and the conditions for exiting the recursive process are checked. If they are not met, the candidate with the lowest total votes shall be eliminated. If there is a tie among candidates with the lowest number of votes, a tiebreaker shall determine who shall be eliminated. A similar tie-breaking process is initiated in determining the winner once there are two candidates remaining if there is a tie.

4.8.2 Stimulus/Response Sequences

1. The user inputs the file.
2. The system determines the file is of IR election type and runs the IR algorithm.

4.8.3 Functional Requirements

REQ-1: The system must be able to tally the number of first-choice votes for a candidate.

REQ-2: The system must be able to flip a coin in the event of a tie(s) in an unbiased way to determine which candidate gets eliminated or who wins the election in special circumstances.

4.8.4. Associated Use Case

Name	Conduct IR Algorithm
ID	UC_008
Description	The system processes the election data based on IRV specifications.
Organizational Benefits	This allows for voting officials to run an IR voting election and process votes. Users can see the winner of an IR voting election through this system without having to tabulate and calculate the results manually.
Frequency of Use	Every Instant Runoff election
Actors	Voting Official, Testers, Voting System
Triggers	The IRV specifier has been read in (see 4.2).
Preconditions	Election data regarding the number of contestants, contestant identifiers, number of ballots, and ballot information all exists inside the file. An audit file has been created.
Postconditions	The election winner has been chosen.
Main Course	1. The first-choice votes are tallied for each candidate (see AC1). 2. The system decides if a candidate has won a majority by using the result of this tallying process (see AC2). 3. The winner of the election is returned.
Alternate Courses	AC1 A candidate has already been eliminated when tallying votes. 1. Check all ballots where the eliminated candidate was top choice, and based on the next choice assign votes to a new candidate if the ballot had ranked a next choice candidate. 2. Return to Main Course Step 2. AC2 There is no candidate who has won a majority of the votes. 1. If there is a tie between the last two candidates, flip a coin and break the tie to determine the winner (see 4.9). Return to Main Course Step 3. 2. If there is no tie between the last two candidates, then the candidate who received a higher number of votes is the winner. Return to Main Course Step 3. 3. If there are more than 2 candidates remaining, take the

	candidate(s) with the least amount of votes and break ties until exactly one candidate is eliminated (see 4.9). Check all ballots where the eliminated candidate was top choice, and based on the next choice assign votes to a new candidate if the ballot had ranked a next choice candidate. Return to Main Course Step 1.
Exceptions	None

4.9 Break a Tie

4.9.1 Description

Break a tie between two or more candidates or parties.

4.9.2 Stimulus/Response Sequences

1. The user inputs the .csv file.
2. The system conducts the algorithm on its stored data, determines a tie, and breaks it.

4.9.3 Functional Requirements

REQ-1: The system must be able to simulate a completely random, unbiased, and fair coin toss.

4.9.4 Associated Use Case

Name	Break a Tie
ID	UC_009
Description	The system flips a coin to determine a winner when two parties have the same allocation of seats in CPL or when two candidates have the same amount of votes or the same rank in CPL and IRV respectively.
Organizational Benefits	This allows for completely unbiased determination of seats or a winner in an election, maintaining the election's credibility.
Actors	Voting System, Election Official, Testers
Triggers	There are four possible triggers for breaking a tie: 1. There is a tie between two or more candidates for elimination in IR voting. 2. There is a tie between the final two candidates in IR Voting. 3. There is a tie in CPL voting where two or more parties tie for the number of seats they may obtain, but there are not enough available seats to hand out equally. 4. In CPL voting, there is a situation where a party wins more seats than the number of candidates they have listed. In the event of

	this, unclaimed seats will be distributed to other parties via a random lottery.
Preconditions	There exists a tie between two or more candidates or parties.
Postconditions	The tie-breaking results are determined and returned.
Main Course	1. The system determines a tie. 2. The system flips a coin. 3. The system returns a winner.
Alternate Courses	AC1: There is a tie between three or more candidates. 1. Each candidate or party flips a coin against every other candidate or party in a round-robin fashion. 2. The worst performing candidate or party is eliminated. 3. Repeat until there is a two-way tie. 4. Return to Main Course step 2. AC2: In CPL, a party wins more seats than it has candidates. 1. See 4.6
Exceptions	None

4.10 Write to the Audit File

4.10.1 Description

To ensure the integrity and validity of the results, an audit file must be generated. This audit file shall contain all information present in the input file, information documenting the process and path of the election algorithm, and the final results of the election. This data will be written to the file throughout the election calculation process.

4.10.2 Stimulus/Response Sequences

1. The user inputs the election file.
2. As the system processes the election, the records are written to the audit file.

4.10.3 Functional Requirements

REQ-1: The input file must exist.
REQ-2: The input file must be valid.
REQ-3: The election algorithm must proceed without error.
REQ-4: The audit file must be successfully opened.
REQ-5: If an error occurs when attempting to write to the audit file, the system must be able to return an error and exit the program.

4.10.4 Associated Use Case

Name	Write to the Audit File
ID	UC_010
Description	The system will write information on the elections to an audit file.
Organizational benefits	This allows testers and election officials to see clearly what happened throughout the election process, allowing them to verify their results. Additionally, having a record of the election is useful.
Actors	Voting System, Election Official, Testers
Triggers	The program arrives at a point where it needs to record information, which can occur at multiple moments including: 1. Processing and storing the data within the file has begun. 1. One round of IR Voting has completed. 2. One seat allocation of CPL has been completed. 3. A tie is broken. 4. Processing of an election has concluded. 5. A single step in either election type has been completed, such as transferring ballots when a candidate in an IR Voting election has been eliminated.
Preconditions	The audit file has been successfully created. Reading of the voting file, processing of the voting file, and conduction of elections are fully functional such that the system goes through its entire process and records information.
Postconditions	The audit file is finalized with all necessary information and is viewable. The program has run to completion.
Main Course	1. The system reads election type from voting file (see 4.2) and records it in the audit file 2. The system records the information stored by 4.4 or 4.7 depending on the type of election, including items such as total number of votes. 3. The system records information produced by 4.5 or 4.8 accordingly. 4. Once the winner(s) have been decided depending on the election type, record results in the audit file.
Alternate Courses	AC1: CPL election encounters a tie. 1. After Main Course step 3, record information about the results of the tie break. 2. Proceed to Main Course step 4. AC2: IR election encounters no majority. 1. After Main Course step 3, record information on which candidate gets chosen as the winner. 2. Proceed to Main Course step 4.

	AC3: A candidate is eliminated in IR. 1. After Main Course step 3, record information on who got eliminated. 2. Proceed to Main Course step 4.
Exceptions	EX1: Failure to write to the audit file: 1. The program returns an error and terminates.

4.11 Display Results to Terminal

4.11.1 Description

After the votes are tallied, print the results to the user in the command line.

4.11.2 Stimulus/Response Sequences

1. The user inputs the file.
2. The system runs the algorithm and displays the results in the terminal.

4.11.3 Functional Requirements

REQ-1: The system must be able to complete the vote tally for both the CPL and IRV types of election.

REQ-2: The system must be able to determine a winner in the case of a tie for both types of elections.

4.11.4 Associated Use Case

Name	Display Results To Terminal
ID	UC_011
Description	Basic information about the election is output to the terminal. This information includes the total number of votes, the winner, number of votes received per candidate, and more.
Organizational Benefits	Displaying the output provides a concise summary of the most important information about an election. This allows for users to understand the results of an election without needing to study a more complex audit file.
Frequency of Use	Every election
Actors	Voting Official, Testers, Voting System
Triggers	The election has run to completion, i.e. the winner(s) have been decided.
Preconditions	Related information to be output, such as the number of votes per

	candidate/party, was properly tabulated and stored.
Postconditions	The information about the election is output to the terminal.
Main Course	1. The following information shall be retrieved: a. Number of ballots cast b. Number of votes received per candidate c. Election type d. Winner 2. The information shall be output to the terminal.
Alternate Courses	AC1: The election type is CPL. 1. The following information shall be retrieved: a. Number of votes received per party b. The winners c. The percentage of seats received per party d. The candidates associated with a party in proper order shall be collected in addition to the election type and number of ballots cast. 2. Return to Main Course Step 2.
Exceptions	EX1: The system has failed to obtain information. 1. Display as much of the information listed in Main Course Step 1 as possible. 2. Notify user that some information is missing.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system must be able to process 100,000 ballots in under 4 minutes on CSELab machines. This means that the system must be able to process a large number of votes quickly. The system should not expect only a small number of votes to be processed.

5.2 Safety Requirements

There are no requirements or policies related to safety.

5.3 Security Requirements

All election security is handled at individual voting centers. The input file should not be able to be edited by the system, and the audit file should not be able to be edited after the program runs to completion.

5.4 Software Quality Attributes

It must be possible to add new features into the voting system, such as potentially allowing write-in candidates or other pertinent information while maintaining correctness.

The voting system must be correct. For example, the results of both IR voting elections and closed party elections must be accurate and produce expected and unbiased results. In situations where ties are broken, the process must be fairly handled. For instance, if there were ever a tie or ties in an election, running an election 10,000 times should produce unbiased results that indicate that all candidates or parties involved in a tie have equal chances of advancing.

5.5 Business Rules

Voting officials can run the program, input a file, see output that summarizes key statistics in the election, and obtain an audit file. Similarly, testers may also perform the same functions. Use of the voting system is limited to voting officials and testers. Input files are in .csv format, output audit files are in .txt format. Elections are called by government officials, results shall be displayed on the terminal.

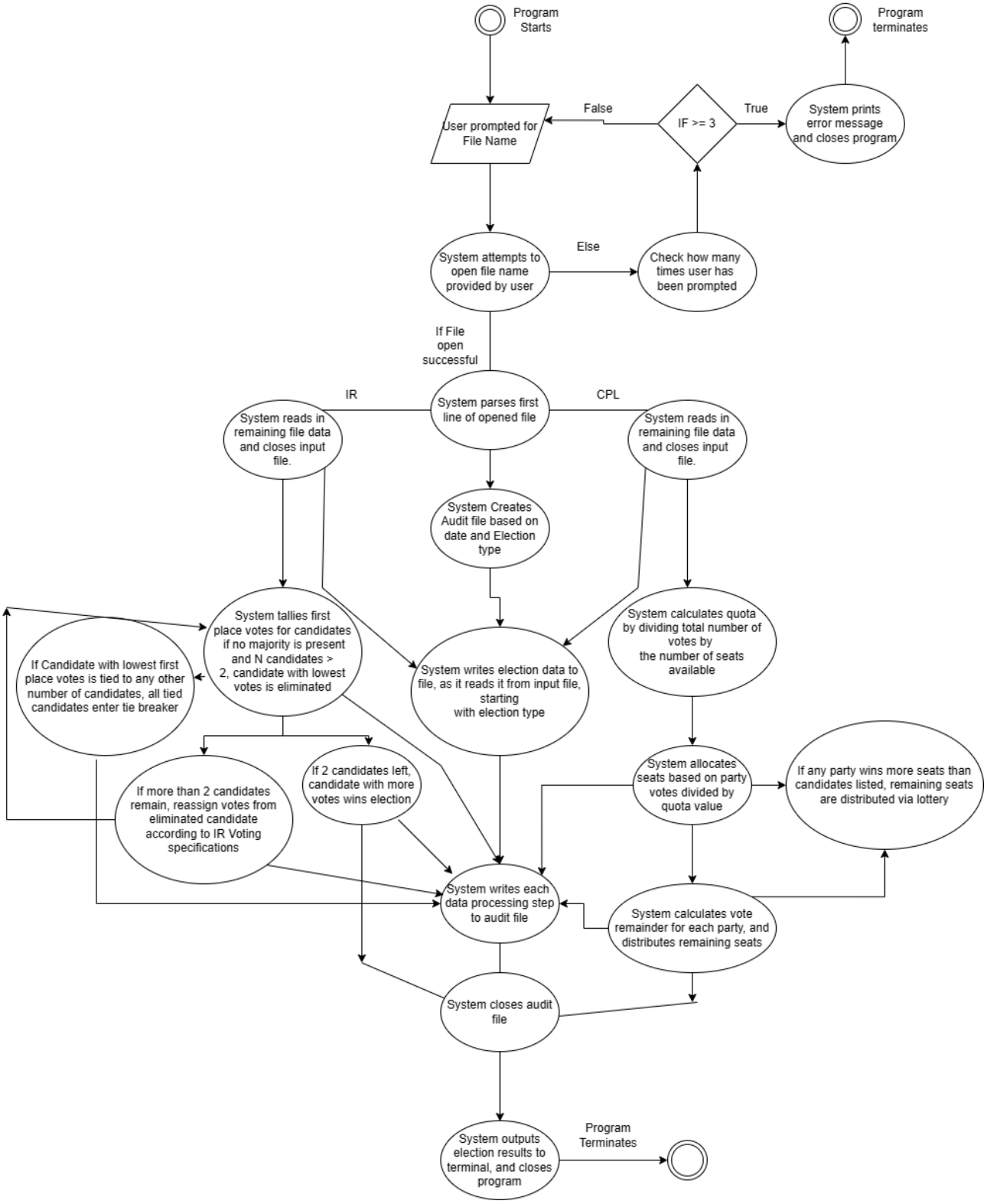
6. Other Requirements

There are currently no other requirements.

Appendix A: Glossary

Term	Definition
IR Voting/IR/IRV	A system of voting in which voters rank candidates on a ballot by preference. In this system, if a candidate has a majority of the votes, the candidate wins the election; if not, the candidate with the least number of votes is eliminated, and their votes are added to the next choice indicated by each voter. This repeats until there is a majority, or if all votes have been handed out and there's still a tie, the winner is declared through popularity.
Closed Party Voting/CP Voting/CPL	A system in which voters choose a party, and based on the total votes that each party receives, and the total seats available in a given election, the seats are allocated among the parties.
Audit File	A file containing all information necessary to verify the election process and results.
Voting Official	An official who oversees an election and is involved with calculating the results of an election.
Tester	Somebody who runs tests on software to detect the presence of bugs.
Lottery	Random process with at least 2 participants and a single winner.
CSE	College of Science and Engineers (referring to the engineering college at the University of Minnesota)
CSELab Computer	Refers to a specific Computer interface used for CSE labs, and runs on an Ubuntu Linux environment.

Appendix B: Analysis Models



Appendix C: To Be Determined List

N/A