Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Jacob McIsaac, Dan Kong, Caleb Otto, and Ruolei Zeng

CSCI 5801 Team 10

February 2023

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Jacob McIsaac	1/28/23	Inserting Template	0.7
Jacob McIsaac	1/30/23	Identified TBD items/areas of concern	0.71
Jacob McIsaac	2/9/23	Added sections 2.6, 5.1, and 5.2.	0.75
Dan Kong	2/10/23	Added sections 2.4, 2.5, 2.7, 3.1, and 3.4.	0.8
Caleb Otto	2/11/23	Added sections 1.1, 1.2, 1.3, 1.4, 1.5, 2.1, 2.2, and 2.3.	0.9
Jacob McIsaac	2/11/23	Added section 5.3.	0.91
Ruolei Zeng	2/12/23	Added sections 3.2, 3.3, 5.4, and 5.5.	0.95
All members	2/15/23	Added section 4, finalized format.	0.98
Dan Kong	2/16/23	Touched up introduction	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description and explanation of the voting system Version 1.0. It will define the functionality and purpose of the system as a whole, how the user is able to interact and use it, and what it actually does including the limitations that it has. This write-up is meant for users of the system as well as anyone seeking to improve on or add to its functionality.

1.2 Document Conventions

This document was created using the IEEE template for System Requirement Specification Documents. This will be used in conjunction with the Waterfall method of development for the overall success of the system. There are no special fonts or other typographical conventions with special significance used in this document.

1.3 Intended Audience and Reading Suggestions

This SRS document is broken up into the following sections: Introduction(1), which gives a basic understanding of the system. Overall Description(2), which provides information on how the product works and functions. External Interface Requirements(3), which defines how users, hardware, and software interact and interface. System Features(4), which defines what features and functions the system has. Other Nonfunctional Requirements(5) describes the requirements of the system that aren't necessary for its overall functionality, such as how fast it must perform as well as safety, security, and rules. Other Requirements(6) defines anything not covered earlier. Appendices are for reference on new terminology in the documentation, or anything that needs more clarification. Below are the intended audiences of the SRS:

- Users of the software such as election officials, who intend to use it for counting up ballots and votes in order to determine the winner in the two types of elections.
    - Should begin with reading through the overall description(2) to ensure it is understood how it works.
    - Continue with the External Interface Requirements(3) to develop knowledge of how to actually use the system.
    - This covers the most important features and then other parts of the document that should be read in order to ensure full understanding.
- Media, in order to examine the results and voting process from the elections.
    - Begin with the Overall Description(2) to familiarize with how the system works and how results are determined.
    - Continue with External Interface Requirements(3) to see how to obtain information about the election.

    - The rest of the sections can be read in any order to ensure that the reporting is based on full understanding.
- Auditors of the election, in order to examine the voting and election process for validity.
    - Start with the Overall Description(2) to ensure that the system is calculating the results correctly and will process an election in a fair manner.
    - Continue with Nonfunctional Requirements(5) to ensure the security and activity of the system is designed well enough to handle elections.
    - The rest of the sections can be read in any order to check for any other flaws in the system as a whole.
- Engineers who seek to expand or improve upon the functionality of the system as a whole.
    - Start with the Overall Description(2) to ensure you understand how the system works as a whole.
    - Then work through System Features(4) to determine what features the system has and how it actually all works together.
    - The rest of the sections can be read in any order to complete your understanding of the project prior to adding functionality or improvements.

1.4 Product Scope

This voting system is a tool that will be used by election officials to determine winners in two types of elections, Instant Runoff Voting and Closed Party List Voting. It will take in a file with information on the election as well as ballots and determine the winner from these in both election types, in order to avoid the difficulties and tendency for error or cheating that can come with manual counting. The user will be able to input information that isn't included in the file. The software will also produce an audit file for use in determining the validity and accuracy of the election if needed which defines the election information and history. The election results may also be shared with media personnel. In the case of any tie, it will also be able to randomly and fairly determine a winner. This software is meant to optimize the election process in an efficient and secure manner.

1.5 References

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

2. Overall Description

2.1 Product Perspective

The Voting System is designed to be used in easily determining the winner of elections in both of the types it is meant to handle (IRV and Closed Party List). It should be able to take in a CSV file full of ballot information and determine the winner of the election in either case. It is also meant to

ensure the security, accuracy, and integrity of the election by producing an audit file that can be reviewed if needed. There are voting systems already on the market as a whole but this will add to them and ensure a lesser need for by-hand counting for elections.

2.2 Product Functions

System:

- Take in a CSV file for the election.
- Determine the type of election and other election metadata from the file or the user.
- Count the ballots and analyze them based on the election type.
- Produce the winner and, if necessary, perform a tiebreaker.
- Keep a record of the election history and produce an audit file with all pertinent information.
- Provide an option for the information to be shared with any media

User:

- Can input the CSV file.
- Can manually input election metadata (such as the total number of ballots, candidates, etc).
- Receives an audit file.
- Can see the results of the election.
- Can choose to send results to the media.

2.3 User Classes and Characteristics

- Election officials (users) that will use the voting system to determine the results of an election of either of the two types defined above.
- Auditors and election reviewers will use the audit file to determine the validity and accuracy of the election.
- Media officials who will be sent the results of the election.
- Future programmers who will add to the system so it is able to handle other types of elections or optimize it further.
- System testers will need to use the system often to verify its functionality.

2.4 Operating Environment

The program will operate in an Ubuntu 20.04.5 LTS environment and will require a functional computer to run. The system should have an Intel Core i7 (2.5 GHz) processor and 32GB of RAM. As long as the program is installed on the computer, and the necessary files are provided (see 3.3), no other requirements will be necessary for it to function.

2.5 Design and Implementation Constraints

The program must take in a .csv file that is comma-delimited. The program must provide an audit log at the end of processing the .csv file. For Instant Runoff Voting, the ballot will have at least 1

person ranked. There will never be more than one file given per election. For closed party listing, all independent groups will have a single person in the group. There will be no write-in candidates for this system. The program cannot change the file structure outside of the program since the election files will come in the predetermined format. If there is ever a tie, flip a coin via the computer. The program must randomly select the winner in a fair coin toss. If there is not a clear majority in IR, then popularity wins after all votes have been handed out. The program will calculate the winner(s) based on the number of seats and the order of the candidates. An election should be able to run 100,000 ballots in under 4 minutes. The election file will be located in the same directory as the program. The audit file will show all metadata for an election which is provided in the file (see section 2.7). Auditing should be able to follow the order of the ballots being assigned to the candidates. The audit should show the order of removal of candidates in IRV and what ballots were redistributed. For CPL voting, the audit file will also show the order of ballots being assigned to parties as well as the process of assigning seats to each party and the associated calculations. The audit file should show all of the steps of the election, and it should be able to replicate the election itself.

2.6 User Documentation

There will be no formal delivery of user documentation or tutorials with this system. The system is designed to require minimal interaction from the user, which means that in-depth documentation is unnecessary. Election officials who use this system will need to be given brief training on how to use the system and what the end result of running the program will be. This will be as simple as coaching the election officials on how to run the program, how to enter the correct file name, and how to locate the audit file.

2.7 Assumptions and Dependencies

The following is assumed: the user has a functional computer with the correct operating environment installed (see 2.4), the user has access to or has downloaded a valid .csv file, the user has access to the directory in which the program is installed, and the user has read permissions on the .csv file. There are no errors in the file or the ballots. For an Instant Runoff ballot, if there is a ballot in the file, it will have at least one of the candidates ranked. There are no errors in the ballots. The first line of the .csv will list the type of voting (i.e. IRV, CPL). There will be no more than one file per election. We are assuming that the .csv election files will adhere strictly to the following two formats, depending on the election type. Note that each row in the table is a line in the file.

IR
<<Number of candidates>>
<<Comma-separated list of candidate names and parties i.e. Johnson (D), etc>>
<<Number of ballots>>

<<Ballot entry 1 i.e. 1,4,2,3>>
<<Ballot entry 2 i.e. ,,1,2>>
<<More ballot entries until end of file>>

The above table is for an IRV file. Notice how each ballot is assumed to have at least one candidate ranked (but they do not have to rank them all). Additionally, any rankings that are made will be in order. For example, someone couldn't rank a first and third choice without ranking a second choice. Here is the assumed and strict format for CPL files:

CPL
<<Number of parties>>
<<Comma-separated list of party names>>
<<Comma-separated list of candidates for first party>>
<<Comma-separated list of candidates for second party, etc>>
<<Number of available seats>>
<<Number of ballots>>
<<Ballot entry 1 i.e. ,,1,>>
<<Ballot entry 2 i.e. 1,,,>>
<<More ballot entries until end of file>>

Note that the ballots in CPL will only have one entry marked. This "1" will represent the party that that person is voting for.

3. External Interface Requirements

3.1 User Interfaces

The user will run the program in the console and errors and prompts will show up in the console as text. When the program is finished, the program will notify the user by text via the console, informing them of the winner, the type of election, the number of ballots cast, and stats for every candidate.

3.2 Hardware Interfaces

This voting system has to support desktop computers (the standard ones) having a RAM of 32GB and above. The processors of such computers should be Intel Core i7 @ 2.5 GHz. The overall system needs to have printers attached to it for the printing out of results as well as audit files. This voting system likewise needs to have more secure methods for the storage of ballot information. Such a storage method might include an encrypted USB drive or hard drive.

3.3 Software Interfaces

The voting system needs to be fully compatible with a specific LTS operating system, which is Ubuntu 20.04.5. Additionally, the machines must have the gcc compiler version 9.4.0. Standard C++ libraries should be installed on the machine for compilation purposes. The voting system needs to have the ability of reading and processing data and information derived from CSV files. This system needs to have the capacity of generating audit files in readable formats such as TXT. This system needs to also have the ability to communicate with printers, which will be printing out audit files and results.

3.4 Communications Interfaces

There are no necessary communication interfaces for the program.

4. System Features

While reading this section, please note that all use cases are considered to be a high priority unless otherwise specified.

4.1 Prompting User for Election File Name

4.1.1 Description and Priority

The user will be prompted to enter the name of the file which contains the election information as well as the data for all ballots in the election.

4.1.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_GET_FILENAME in the use cases document.

4.1.3 Functional Requirements

REQ-4.1.1: The system must recognize when no command line arguments are given. In response, it will prompt the user to enter a .csv file name.
REQ-4.1.2: The system must be able to save the input given by the user.
REQ-4.1.3: The system must parse the inputted filename to ensure it is of type .csv.

REQ-4.1.4: If the file is of incorrect type, the program must print an error message explaining this. It will then reprompt the user for a filename.

4.2 Getting Election Filename on Command Line

4.2.1 Description and Priority

The program receives and internalizes the name of the election ballot file from a command line argument given by the user.

4.2.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_CMD_LINE_FILENAME in the use cases document.

4.2.3 Functional Requirements

REQ-4.2.1: The system must recognize when it has one or more command line arguments.
REQ-4.2.2: The program must verify that the first command line argument given is a valid .csv file name.
REQ-4.2.3: The system must print an error message when an invalid .csv file name is given, and will then prompt the user to input the file name manually.

4.3 Prompt User to Input Election Metadata

4.3.1 Description and Priority

After the file is successfully opened, the user will be given the option to manually input the election metadata (type of election, number of candidates, number of ballots, etc). This use case can be considered medium to low priority since it is mainly being included to prepare for potential multifile elections in the future. However, this system is not required to deal with several files per election for this specification.

4.3.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_ASK_META in the use cases document.

4.3.3 Functional Requirements

REQ-4.3.1: The system must prompt the user to answer “y” or “n” in response to asking them if they would like to manually enter the file metadata.
REQ-4.3.2: The program must be able to recognize when the user enters “y” or “n”.
REQ-4.3.3: The program must ask for the election type (“IRV"' or “CPL”) and store this information. If neither of these are entered, it should reprompt the user.
REQ-4.3.4: If IRV was entered, the system should then ask for the number of candidates and this should be saved. It will then ask for a comma-separated list of the candidate names followed by the first letter of their party in

parentheses. This will be saved. It will then ask for the number of ballots to be entered and will save this value.

REQ-4.3.5: If CPL was entered, the system will prompt for the number of parties and will save this. The system will then prompt for a comma separated list of parties and will save this. It will then prompt once for each party and ask for a comma-separated list of the candidates in that party. These will be saved in the program as well. The system prompts for the number of candidates and saves this. It also prompts for the number of ballots and saves this.

REQ-4.3.6: In response to initially receiving "n", the system will move on to manually extracting file metadata.

REQ-4.3.7: The program should reprompt anytime the input given does not match the data type requested.

REQ-4.3.8: The system should print an error message and start over if the number of candidates or parties given does not match the previously input number.

4.4 Opening the Election File

4.4.1 Description and Priority

The program opens up the .csv ballot file that was given by the user and prepares to read data from it.

4.4.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_OPEN_BALLOT_FILE in the use cases document.

4.4.3 Functional Requirements

REQ-4.4.1: The program must open the .csv file that it is given for reading only.

REQ-4.4.2: If the file is not present in the current directory, an error message should be printed and the user should be reprompted for a filename.

4.5 Parse File for Election Type

4.5.1 Description and Priority

The system feature here revolves around parsing files in determining the election type being conducted. The entire process here is very important (high priority) because the election type determines the specific procedures and rules that must be followed.

4.5.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_PARSE_ET in the use cases document.

4.5.3 Functional Requirements

REQ-4.5.1: The system needs to parse file formats to correctly establish the election types.

REQ-4.5.2: In situations where the file formats are not correct, the voting system will be providing error messages while prompting users to upload correct format files.

REQ-4.5.3: The system will be providing appropriate or suitable feedback directly to its users on the specific election types that have been identified.

4.6 Parse File for IRV Information

4.6.1 Description and Priority

The system feature here revolves around parsing files for data and information associated with IRV (Instant-Runoff Voting).

4.6.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_PARSE_IRV in the use cases document.

4.6.3 Functional Requirements

REQ-4.6.1: The system needs to parse file formats for IRV-related data and information, like the total number or amount of seats being filled and candidates.

REQ-4.6.2: In situations where the files do not have IRV-related data, the voting system works to skip the step and immediately proceed with the remaining processing steps.

REQ-4.6.3: The system will be providing appropriate or suitable feedback directly to its users on all identified IRV-related data and information.

4.7 Parse File for CPL Information

4.7.1 Description and Priority

It involves extracting the CPL (closed party list) information from input files which is used to conduct Closed Party List elections.

4.7.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_PARSE_CPL in the use cases document.

4.7.3 Functional Requirements

REQ-4.7.1: The system needs to parse file formats for CPL-related data and information, like the total number or amount of seats being filled and amount of votes a voter can be permitted to cast.

REQ-4.7.2: In situations where the files do not have CPL-related data, the voting system works to skip the step and immediately proceed with the remaining processing steps.

REQ-4.7.3: The system will be providing appropriate or suitable feedback directly to its users on all identified CPL-related data and information.

4.8 Read in Ballot Data

4.8.1 Description and Priority

The program will read in the information for each ballot in the election file and store it internally for use in the election algorithms.

4.8.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_READ_BALLOTS in the use cases document.

4.8.3 Functional Requirements

REQ-4.8.1: The system should parse through each ballot line in the file after it has read the header. The ballot information should be stored in an object, and those objects should go into a data structure for processing.

REQ-4.8.2: The system should recognize when the end of the file has been reached and stop parsing the file. The file should then be closed.

REQ-4.8.3: The program should recognize when it runs out of memory (in the case where there are too many ballots) and abort.

4.9 Determine Winner in IRV Election

4.9.1 Description and Priority

Determine the winner of an Instant Runoff Voting election.

4.9.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_IRV_WINNER in the use cases document.

4.9.3 Functional Requirements

REQ-4.9.1: The IRV election data will have been read in already to each candidate.

REQ-4.9.2: The second and third place votes from voters are able to be accessed if the need arise for redistribution of votes.

REQ-4.9.3: The system needs to be able to go by popularity if majority is not present(see 4.10).

REQ-4.9.4: The system needs to be able to run however many rounds necessary in order to determine the overall winner as well as the placement of other candidates.

4.10 Determine Winner When No Candidate Has Majority at End of IRV

4.10.1 Description and Priority

Determine the winner in IRV via popularity vote if there is no clear majority.

4.10.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_IRV_POPULARITY_VOTE in the use cases document.

4.10.3 Functional Requirements

REQ-4.10.1: There must not be a clear majority at the end of a round in IRV.
REQ-4.10.2: There must be a winner determined via popularity or using a tie breaker.
REQ-4.10.3: The data must be able to be accessed for the vote and majority counts.

4.11 Determine Winner(s) in CPL Election

4.11.1 Description and Priority

Determine the winner for a Closed Party List election from the data in the CSV file.

4.11.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_CPL_WINNER in the use cases document.

4.11.3 Functional Requirements

REQ-4.11.1: The largest remainder formula must work successfully to allocate seats to the winning parties.
REQ-4.11.2: Parties must assign their earned seats in the order of their candidates.
REQ-4.11.3: Must be able to log the vote counts of different parties and their seat allocations where it can be accessed later.
REQ-4.11.4: Must be able to handle seat issues such as when a party wins more seats than they have candidates for(see 4.12) or a tiebreaker for the last seat(see 4.13)

4.12 Distribute Seats When Party Wins More Seats than They Have Candidates

4.12.1 Description and Priority

Determine how to lottery off excess seats if a party earns more seats than candidates in CPL.

4.12.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_SEAT_LOTTERY in the use cases document.

4.12.3 Functional Requirements

REQ-4.12.1: A party earns more seats than they have candidates for.

REQ-4.12.2: There must be enough candidates who didn't earn seats to fill the remainders.

REQ-4.12.3: The system uses tiebreaker(see UC_TIE) to lottery off excess seats to other parties.

4.13 Break a Tie that has Occurred

4.13.1 Description and Priority

In the case of a tie, a random coin flip will be used to determine the winner.

4.13.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_TIE in the use cases document.

4.13.3 Functional Requirements

REQ-4.13.1: There must be a tie for a seat in CPL, a need for seat lottery in CPL, a tie for lowest votes in IRV, or a tie for winner in IRV(see UC_TIE).

REQ-4.13.2: Ties must be broken with a random coin flip, there should be no biased option.

REQ-4.13.3: The ties must be logged in the history of the election for further reference.

4.14 Create/Open Audit File for Writing

4.14.1 Description and Priority

To verify the election, an official needs an audit file to confirm the process of the election and the results of the software.

4.14.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_CREATE_AUDIT in the use cases document.

4.14.3 Functional Requirements

REQ-4.14.1: The user must have write permissions on the .txt file that is created.

REQ 4.14.2: The system must create the file with the election type, year, month, and day in the file name.

4.15 Populate Audit File

4.15.1 Description and Priority

Populate the audit file with relevant election information so that the election process can be verified and is clearer.

4.15.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_POPULATE_AUDIT in the use cases document.

4.15.3 Functional Requirements

REQ-4.15.1: The user must have write permissions.
REQ-4.15.2: The election results must be finished processing.
REQ-4.15.3: The audit file must be created..
REQ-4.15.4: The system must write all election metadata, ballot information, and election result calculation information to the audit file.
REQ-4.15.5: The system must close the file when finished with writing to it.

4.16 Print IRV Results to Terminal

4.16.1 Description and Priority

Once the voting file has been processed and the IRV voting algorithms have been run on the ballot data, the program will output important information regarding the results of the IRV election.

4.16.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_IRV results in the use cases document.

4.16.3 Functional Requirements

REQ-4.16.1:After the IRV election has been fully processed, the system should print "Instant Runoff Voting" to the terminal.
REQ-4.16.2: A table will then be printed. The table will show the candidate names and their party in the left column, with the winner having an asterisk before their name. There will then be a column for the number of votes received and a column for percentage of total votes acquired.
REQ-4.16.3: In the case where one vote transfer was needed, the following table is printed: The first column shows the candidates and their parties, and the winner has an asterisk before their name. The second column will show the number of original first choice votes and the percentages of the total first choice votes as well. The third column will show the number of votes gained by the three highest vote getters from the last place candidate. The new totals of votes are displayed in the fourth column, as well as the percentage of votes after adjustment.
REQ-4.16.4: In the event where two transfers are required to find the winner, the table should be: The first four columns are the same as AC1 step 2, but two additional columns are added here. The fifth column will show the number of votes transferred from the third place candidate to the two top candidates. The fifth column will show the new totals and the percentage of votes for the final two candidates.

REQ-4.16.5: When more than two transfers are required, the table will be the same as described above; additional rows will be added to display additional vote transfers. These vote transfers will be displayed the same way that they are described in the above requirements.

REQ-4.16.6: The program should terminate after the table is printed.

4.17 Print CPL Results to Terminal

4.17.1 Description and Priority

Once the voting file has been processed and the CPL voting algorithms have been run on the ballot data, the program will output important information regarding the results of the CPL election.

4.17.2 Stimulus/Response Sequences

Refer to the triggers, courses, and exceptions of use case UC_CPL_RESULTS in the use cases document.

4.17.3 Functional Requirements

REQ-4.17.1: The program will print "Closed Party List Voting" to the terminal after the election has been fully processed.

REQ-4.17.2: The program should then print a table. The table will contain 7 columns. The first column will display the names of all parties in the election. The second column will show the number of votes that each party received. The third column will show the number of seats that each candidate has earned after the initial allocation of seats. The fourth column will show the number of remaining votes for each party after this initial seat allocation. The fifth column will then show how many seats are allocated to each party in the second allocation of seats. The sixth column shows the final number of seats for each party. The seventh column will display the percentage of the vote for each party and also the percentage of the total number of seats that that party has been allocated.

REQ-4.17.3: The program should terminate when the table has been fully printed.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system should be able to run to completion on a ballots file that is 100,000 ballots large in under four minutes. This means that, when the ballots file contains 100,000 ballots, the system should be able to process the ballot data, produce the audit file, and report the winner or winners in less than four minutes. This rate of performance equates to under one minute for a ballots file with 25,000 ballot entries. The speed at which a certain ballot file should be processed can be calculated

from this given information and the number of ballots in the file. Following this performance requirement is crucial as election results must be reported in a timely manner in order to ensure a speedy result.

5.2 Safety Requirements

There are no special safety requirements that need to be documented here due to the nature of the system as a data processing program. Due to the fact that the program is operating on sensitive election data, safeguards will need to be in place to ensure this data is not tampered with, which is specified in the section below (Section 5.3: Security Requirements).

5.3 Security Requirements

While ballot information is sensitive data, there will be no special security requirements of the system when it comes to ensuring that the ballot data in the file is accurate and not altered. Security such as ensuring that only one vote is cast per person and other general election security practices are handled at the voting centers and is not a responsibility of this system. We are assuming that the ballot file that is given to the system to analyze has not been tampered with and adheres strictly to the file format specified in section 2.7. When it comes to producing the audit file, which is described in detail in section 2.5 (as well as 4.15 and 4.16), special precautions must be taken to ensure that this file is not allowed to be tampered with and is also not replaced. The file will have the naming convention “et_day_month_year” to ensure that each election file has a unique identifier. Note that “et” refers to the election type (“CPL” or “IRV”) and the day, month, and year will be the date that the audit file was produced. Additionally, the audit file will only have reading permissions; this will ensure that the file cannot be tampered with.

5.4 Software Quality Attributes

Robustness

The voting system needs to have the capability of handling unexpected inputs while still providing accurate results.

User-friendliness

The voting system needs to have one user-friendly interface that makes it easy for officials to use when inputting or processing data.

Reliability

The voting system needs to have the capability of handling vast data amounts while operating unswervingly without errors.

Accuracy

The voting system needs to be accurately determining the election winners based upon the data fed into it (input data alone)

Security

The voting system needs to have more secure methods when it comes to the processing and storage of ballot information so as to prevent any sort of hacking or tampering.

Integrity

The voting system needs to be able to generate audit files. These audit files will get reviewed so as to ensure some validity and reliability of the election.

5.5 Business Rules

One notable business rule here revolves around the idea that just the authorized officials of the election can access this voting system. These authorized officials will be the only ones granted permission to input and process data. All audit files need to be made freely available to auditors and the media for review after the overall election process. All results that this voting system generates must be considered final so that they do not get subjected to any form of manipulation.

6. Other Requirements

This system is expected to be used several times each year. More specifically, it will be used during each election (normal or special). It therefore needs to be reusable at these times and ultimately reliable enough to allow accurate, fair, and speedy election results.

Appendix A: Glossary

- IRV: Instant runoff voting. This style of election involves people ranking their preferences for candidates. Each candidate's first place votes are compared. If one candidate has a majority, they are the winner. If no candidate has a majority, the candidate who has the fewest votes has their votes distributed to the remaining candidates; this means that those who voted for the losing candidate will give their second choice a vote. This process of elimination and redistribution continues until one candidate has more than 50% of the votes.
- CPL: Closed party list voting. This type of voting is where people vote for a particular party. Using a formula called the largest remainder formula, each party wins an initial number of seats that are allocated to that party based on the ordering of the candidate list. Candidates with the largest number of remaining votes after this initial seat allocation get the remaining seats.
- CSV: Stands for "comma-separated value". Describes a file with extension .csv. These files contain values on each line, and these values are separated by commas.

- TXT: Short for text. Specifies a file type that has the extension .txt. These files contain ASCII characters.

Appendix B: Analysis Models

No diagrams are necessary at this time.

Appendix C: To Be Determined List

No TBDs at this time.