Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Conor Brown (brow4339, Team 21), Sean Carter (carte899, Team 21), Joe Cassidy (cassi083, Team 21), and Jack Soderwall (soder365, Team 21)

University of Minnesota

2/18/21

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Jack Soderwall	2-15-21	Section 1: Introduction	1.01
Jack Soderwall	2-16-21	Work on Sections 2 and 3	1.02
Jack Soderwall	2-17-21	Work on Sections 2, 3, 5	1.03
Jack Soderwall, Sean Carter, Conor Brown	2-17-21	Addition of use case diagrams, near-finishing of Sections 2, 3, and 5	1.04
Jo Cassidy, Jack Soderwall	2-18-21	Finishing of Section 2, Finishing of use cases, Finishing table of contents	1.05

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed overview of our Voting System software, including the features of the software, the constraints under which the software will operate, and any other relevant information regarding the operation of this software, including operating environment or hardware requirements.

1.2 Document Conventions

This document was created based on the IEEE standard for System Requirements Specification Documents.

1.3 Intended Audience and Reading Suggestions

This document is intended for programmers, testers, and election officials who will be closely involved with the operation of this software. Programmers and testers may be most interested in use case documentation, where fundamental user procedures will be outlined, as well as things such as the handling of exceptions. Election officials may be most interested in reading Section 4.1 regarding System Features, Section 2 regarding the overall product description, and Section 3 regarding any External Interface requirements. Seeing as how election officials will be the primary users of this software, it is recommended that they pay special attention to the details laid forth by this document.

1.4 Product Scope

Our Voting System software is intended to provide organized election results for two specific election types, Instant Runoff Voting and Open Party List voting. Audit files will be generated that provide a detailed overview of the election, such that the audit files could replicate the election themselves. Our system will not handle the actual voting itself, but rather the votes will be cast online, and then be delivered to our system in the form of a CSV file, which will then be parsed to generate audit files. Text files outlining the election results may also be generated for media use by our software.

1.5 References

Detailed information regarding the voting systems that our software will handle can be found at FairVote.

2. Overall Description

2.1 Product Perspective

This software was developed in an effort to create one robust system that is capable of handling different election types using a single program. This software is entirely independent, as it is a new creation. It has not been built off of any existing election software, and it was specifically designed with the cases of Open Party List and Instant Runoff voting in mind. This software was also designed with the intention of being able to trace through an election via the data provided its audit files.

Development of this software was performed using Java, so it has been designed to work with any operating system that contains the required Java version. This is thoroughly outlined in Section 2.4: Operating Environment.

2.2 Product Functions

1. Read in a file
    - The software will take in the name of the election file at the command line at program start.
    - The file will always be in CSV format.
    - The user must input the file name for the desired election.
2. Determine the election type
    - The software must determine the election type from the file.
    - The first line of the file will determine the election type.
3. Perform counting of votes and generate election statistics
    - Counts each ballot and determines stats such as victors and total votes for each candidate.
4. Produce audit files based on election results
    - Audit files will contain the results of each ballot in order (if applicable).
    - Audit files will contain relevant election information at the time, including total ballots cast, candidates, candidate names, and any relevant calculations made.
    - Individual logging of votes
5. Produce summary files for media members upon request
    - Media members may request to view election results, so general summary statistics will be delivered to them in the form of a text file.
6. Display general election results and stats to the screen
    - After the software is finished counting ballots, the total number of ballots counted, election winners, and statistics such as total votes for each candidate will be displayed to the screen.
7. Handle ties via flipping of a fair coin
    - In the event of a tie, a simulation of a fair coin is used to determine the winner of the election. The coin will be simulated more than once in order to eliminate any possible biases within the coin.

2.3 User Classes and Characteristics

Election officials will be the primary users of this software. They will be the ones who actually operate the system at election time. The other two fundamental user classes will be programmers and testers, who will have technical knowledge of the software that the election officials will not possess. Programmers and testers will also know how to run the program and validate its results in order to confirm its reliability.

2.4 Operating Environment

The software is written in Java which results in it being easily accessible to any operating system so long as they are running on the same JVM and Java version. It is recommended to have at minimum Java 7 installed as earlier versions may work but are not supported. Any versions after Java 7 will work accordingly.

This software will generally be run using one of the following system specifications:

Operating System	RAM	Processor
Windows 10	32 GB	Xeon @ 3.0 GHz
Ubuntu	8 GB	2 x Xeon @ 2.0 GHz
Ubuntu 20.04	32 GB RAM	Intel Core i7-4970 @ 3.6 GHz
MAC OS X	16 GB RAM	Intel Core i5 @ 3.2 GHz

It is recommended that at minimum the software be run on a system with 8 GB of RAM and a 2.0 GHz Quad-Core processor, in addition to the Java version requirements specified above. Since there are no graphics involved with the program, a graphics card is not required for use of this software.

2.5 Design and Implementation Constraints

The main obstacle in the design process will be the timing and memory requirements of the software. An election must be able to run 100,000 ballots in eight minutes, and elections must be able to be held whenever they are called, whether that be during regularly scheduled elections or special elections that are called. Files containing ballots will also always be in CSV format, so the software must account for this when reading the file in to generate the audit file and election results.

2.6 User Documentation

The software will come packaged with a manual in pdf format, complete with startup, use and debug instructions.

2.7 Assumptions and Dependencies

All users are assumed to employ one of the operating environments described in 2.4, allow the software access to the election file, and express the knowledge necessary to run this software from the command line. Additionally, the input election file is assumed to be a csv formatted as follows:

1) The election type is indicated on line 1 (IR for Instant Runoff, OPL for Open Party Listing)
2) Line 2 denotes the number of candidates.
3) Line 3 names the candidates, separated by commas.
4) Line 4 denotes the number of ballots.
5) Each remaining line represents a unique vote such that:
    a) If the election is Instant Runoff, the candidates are ranked 1 through <number of candidates> via comma-delimited values, where the nth value corresponds to the nth candidate listed on line 2. Blank rankings are allowed, so long as no number is skipped among those appearing (i.e. rankings 1, 2, and 3 of five candidates are allowed, but rankings 1, 3, 5 of those same candidates are not)
    b) If the election is Open Party Listing, the chosen candidate is ranked 1 by the rules discussed in (a), and all other candidate rankings are left blank.

3. External Interface Requirements

3.1 User Interfaces

A UNIX operating system and terminal are required. No further software is necessary. A prompt to enter the name of the ballots file will appear when run in the terminal. A general prompt will appear to confirm that counting has begun once the file is entered. Specific error messages will be displayed on the terminal if needed. The system can be stopped or interrupted by typing Control + ‘C’ in the terminal running the program. There will be no GUI components provided, as all information of interest to the user will be output to the audit files.

3.2 Hardware Interfaces

In terms of actual hardware, all that is required for use of the program is a keyboard, mouse, and screen, as the program will be run from the command line. The hardware requirements for running the program are outlined in Section 2.4: Operating Environment, although the more powerful the system, the faster the program will likely be to execute.

3.3 Software Interfaces

Our software will be written in Java, so at a minimum Java 7 should be installed on the system. This was further outlined in Section 2.4 of this document. The software will work on Ubuntu, Ubuntu 20.04, MacOS X, and Windows 10, and the primary communication between this software and the operating system will be via the operating system's file system, as this is where files will be brought in to be processed, audit files will be stored when produced, and media files will be stored when produced.

3.4 Communications Interfaces

Ballots will be fed through a terminal by an election official to the system. The provided ballots file will be in a comma delimited text file. An audit file will be produced at completion with a unique file name for the given election. There are no internet connectivity requirements, as the files fed to this software will already contain the required election results. The file will be in the same directory as the program, so there will be no need for any outside communication mechanisms. While the system itself requires no internet connectivity, there may be internet requirements to send election results to the media should they request it.

4. System Features

4.1 Obtaining File

4.1.1 Description and Priority

The file containing all of the election ballot information is created following the formatting constraints and is placed within the same directory as the program. The priority is high because without the file the program cannot operate.

4.1.2 Stimulus/Response Sequences

1. The file contains all of the ballots information required.
2. The file is verified to ensure that it satisfies the formatting constraints
3. The file is placed within the same directory as the program
4. Upon running the program, the correct file name is passed in as a command line argument

4.1.3 Functional Requirements

Name	Obtaining Filename
ID	VS_01
Description	Supplying file for program use

Actors	Election official, testers
Organizational Benefits	Contains all necessary election information
Frequency of Use	Used once per program call
Triggers	When polls have closed, a ballots file will be made. This file must then be placed in the right directory for program usage
Preconditions	An election is called and the file exists with proper formatting and information
Postconditions	The file is in the proper directory and the program was able to correctly read through the file
Main Course	1. Election is called 2. Ballots are cast 3. Ballots are compiled into a comma delimited text file 4. The file is sent to the correct directory for use 5. The file name is passed as a command line argument upon program start
Alternate Courses	N/A
Exceptions	1. File is empty 2. File does not exist 3. File does not follow proper formatting 4. File is in wrong directory

4.2 Display Election Results

4.2.1 Description and Priority

This function will display a summary of the counted election to the screen once the program is completed. This function is of high priority in order to receive the results in a timely fashion.

4.2.2 Stimulus/Response Sequences

1. The program is run at the terminal with the proper filename entered as a command line argument.
2. The ballots in the file are properly counted based on the election type specified within the file with no errors.
3. The results and summary of the election is displayed in the terminal.

4.2.3 Functional Requirements

Name	Display Election Results
ID	VS_02
Description	At completion of the program display the type of election, the number of seats, election winners, the number of ballots cast and the number of votes each candidate received.
Actors	Election officials, testers
Organizational Benefits	Compiles a concise summary of the election that can be reported quickly.
Frequency of Use	Results will be displayed every time the program completes without error.
Triggers	The program is finished and a winner has been determined.
Preconditions	A winner for the election has been selected after the program finished counting all of the ballots.
Postconditions	A summary of the election will be displayed to the screen
Main Course	1. An election is run in the terminal with the proper file name entered as a command line argument. 2. Determine the election type from the ballots file 3. Count the ballots 4. Display election summary
Alternate Courses	N/A
Exceptions	1. No election type is specified or incorrect election type 2. There is an error with the ballots file while counting

4.3 Creating Audit File (Instant Runoff)

4.3.1 Description and Priority

Creates an audit file containing data and timestamps of the program's progression. Used by the election officials and testers to prove that the election was not fraudulent

as well as get a report over the program's exact procedure. Priority is medium as it is not necessary in calculating the victor, but is important for validity of the election.

4.3.2 Stimulus/Response Sequences

1. Program is run
2. A unique audit file is created based on the time of execution
3. Data is procedurally written to the file
4. Upon completion of reading ballots, results and general data are written to the file
5. File is closed and program finishes

4.3.3 Functional Requirements

Name	Audit File (IR)
ID	VS_03
Description	Creates an audit file for the instant runoff election type containing general election statistics and a summary on the progression of the election
Actors	Election officials, testers
Organizational Benefits	Provides a record of the election information and statistics. Useful for following the algorithms process and verifying the election was not fraudulent
Frequency of Use	For each time the program is run, an audit file is created. Upon completion, the audit file can be viewed whenever necessary
Triggers	The program is run and the type is instant runoff
Preconditions	● An election was called ● The election type is instant runoff ● The program successfully retrieved the ballot file
Postconditions	● File was successfully created ● Data on the audit file is valid ● Data is formatted correctly ● At least one candidate is ranked
Main Course	● Program is run ● A unique audit file is created based on the date and time of program execution ● For each ballot counted, a new line is written to the ballot file displaying the

	ballot information and the time in which is was read ● If a candidate is eliminated from contention, it is written to the file and the new totals for the other candidates are posted ● Upon finishing the counting, the results are displayed and the candidates are ranked (highest to lowest)
Alternate Courses	Ctrl + C is pressed while system is running 1. Program immediately stops the counting of ballots 2. A message stating that the program was stopped early is written to the audit file 3. Statistics of the counted ballots are cast, but no ranking occurs 4. Program exits
Exceptions	● Program terminates before file can be created

4.4 Creating Audit File (Open Party)

4.4.1 Description and Priority

Creates an audit file containing data and timestamps of the program's progression. Used by the election officials and testers to prove that the election was not fraudulent as well as get a report over the program's exact procedure. Priority is medium as it is not necessary in calculating the victor, but is important for validity of the election.

4.4.2 Stimulus/Response Sequences

1. Program is run
2. A unique audit file is created based on the time of execution
3. Data is procedurally written to the file
4. Upon completion of reading ballots, results and general data are written to the file
5. File is closed and program finishes

4.4.3 Functional Requirements

Name	Audit File (OP)
ID	VS_04
Description	Creates an audit file for the open party election type

	containing general election statistics and a summary on the progression of the election
Actors	Election officials, testers
Organizational Benefits	Provides a record of the election information and statistics. Useful for following the algorithms process and verifying the election was not fraudulent
Frequency of Use	For each time the program is run, an audit file is created. Upon completion, the audit file can be viewed whenever necessary
Triggers	The program is run and the type is open party
Preconditions	● An election was called ● The election type is instant runoff ● The program successfully retrieved the ballot file
Postconditions	● File was successfully created ● Data on the audit file is valid ● Data is formatted correctly
Main Course	1. Program is run 2. A unique audit file is created based on the date and time of program execution 3. For each ballot counted, a new line is written to the ballot file displaying the ballot information and the time in which is was read 4. Upon finishing the counting, the results are displayed and the candidates are ranked (highest to lowest)
Alternate Courses	Ctl + C is pressed while system is running 1. Program immediately stops the counting of ballots 2. A message stating that the program was stopped early is written to the audit file 3. Statistics of the counted ballots are cast, but no ranking occurs 4. Program exits
Exceptions	● Program terminates before file can be created

4.5 File Parsing (Instant Runoff)

4.5.1 Description and Priority

Walk through the provided ballots file from the command line and keep track of votes for each candidate. This function is high priority because the program must ensure a fair and accurate election.

4.5.2 Stimulus/Response Sequences

1. An election is run through the terminal by an election official with the correct file name entered as a command line argument.
2. The type of election is gathered from the file
3. Get the number of candidates from the file
4. Get the names and party of each candidate
5. Get the number of ballots in the file
6. Read each lines rankings until you've reached the number of ballots specified and keep track of each

4.5.3 Functional Requirements

Name	File Parsing
ID	VS_05
Description	Parse the file provided from the command line and update votes.
Actors	Programmers, Officials composing the file
Organizational Benefits	Have a computer read the the ballots rather than humans to receive the election results as fast as possible
Frequency of Use	Everytime a file is successfully entered into the program through the command line
Triggers	When a file is specified at the command line by an election official
Preconditions	The file must be in the same directory as the program and be entered correctly at the command line. The file also must have the correct formatting to be properly read. The type of election has already been read in by the determining Election Type function and it is Instant Runoff.
Postconditions	All ballots contained in the file will be accounted

	for and given to the correct candidate.
Main Course	1. Read how many candidate are on the ballot 2. Read the possible candidates names and party 3. Read the number of ballots within the file 4. Read and store the ranks of each ballot for each candidate as they are read
Alternate Courses	N/A
Exceptions	1. There is an error with the file formatting 2. The program is terminated prior to completion 3. Ctrl + C or Ctrl + Z is entered while running

4.6 File Parsing (Open Party Listing)

4.6.1 Description and Priority

Walk through the ballots file provided from the command line and keep track of votes for each candidate. This function is high priority because the program must ensure a fair and accurate election.

4.6.2 Stimulus/Response Sequences

1. The type of the election has been read by the Determining Election Type function.
2. Get the number of candidates from the file
3. Get the names and party of each candidate
4. Get the number of seats for this election
5. Get the number of ballots in the file

4.6.3 Functional Requirements

Name	File Parsing
ID	VS_06
Description	Parse the file provided from the command line and update votes
Actors	Programmers, Officials composing the file
Organizational Benefits	Have a computer read the the ballots rather than humans to receive the election results as fast as

	possible
Frequency of Use	Everytime a file is successfully entered into the program through the command line
Triggers	When a file is specified at the command line by an election official
Preconditions	The file must be in the same directory as the program and be entered correctly at the command line. The file also must have the correct formatting to be properly read. The type of election has already been read in by the determining Election Type function and it is Open Party Listing.
Postconditions	All ballots contained in the file will be accounted for and given to the correct candidate.
Main Course	1. The type of election is acquired 2. Read how many candidate are on the ballot 3. Read the possible candidates names and party 4. Read the ballots in the file, counting votes for each candidate as they are read
Alternate Courses	N/A
Exceptions	1. There is an error with the file formatting 2. The program is terminated prior to completion 3. Ctrl + C or Ctrl + Z is entered while running

4.7 Determining Election Type

4.7.1 Description and Priority

Read the type of election from the first line in the file specified by the election official at the command line. This is of high priority so that the correct calculation can be done to determine the winners.

4.7.2 Stimulus/Response Sequences

1. Open the file specified at the command line
2. Read the first line in the file

4.7.3 Functional Requirements

Name	Determining Election Type
ID	VS_07
Description	Read the first line in the ballots file to determine the election type
Actors	Programmers
Organizational Benefits	No need to enter the election type at the command line. As well as have a computer know which election to do so that it is faster than humans.
Frequency of Use	Every time a file is entered correctly in the command line and the file has proper formatting with the election type, either IR or OPL, on its own line at the very top of the file.
Triggers	When an election official successfully entered the filename through the command line and the file is opened for reading
Preconditions	The file must be in the same directory as the program and have correct formatting
Postconditions	The type of election will be successfully read and stored for file parsing and for the summary after the program is completed.
Main Course	1. Open the file specified at the command line 2. Read the first line of the file to store the election type
Alternate Courses	N/A
Exceptions	1. The file does not have correct formatting

4.8 Grouping Independents (Open Party Listing)

4.8.1 Description and Priority

Upon reading the ballot file, all independents are gathered and grouped together to form their own party. The priority is high, since failure to properly group the independents can result in independents having fewer or no seats gained in the election than they otherwise would have.

4.8.2 Stimulus/Response Sequences

1. Open ballot file
2. For each candidate listed as an independent, add them to the list of other independents

4.8.3 Functional Requirements

Name	Grouping Independents (Open Party)
ID	VS_08
Description	When reading from the ballot file, if a candidate is an independent, list them amongst the other independents.
Actors	Candidates participating in the election, voters who voted for independent candidates
Organizational Benefits	Grouping the independents together allows for the independent party to receive its proportional amount of seats should they receive any
Frequency of Use	Occurs anytime the program is run and it is determined that the election is an Open Party Listing election
Triggers	A candidate registered as an independent runs in an election and receives votes
Preconditions	● Ballot file exists and formatted correctly ● An election has been called ● Ballot file contains independent candidates
Postconditions	● All independent candidates are listed together as if they were their own party ● Only independents occur in the list
Main Course	1. Open ballot file 2. Read through candidate list 3. If the candidate is an independent, add them to a list of other independents 4. Continue until there are no more candidates in ballot file
Alternate Courses	No independents in election 1. Proceed to main program
Exceptions	● File does not exist ● File not properly formatted

4.9 Determining Winner (Instant Runoff)

4.9.1 Description and Priority

Upon completion of reading the election file, the program must determine whether a candidate has received more than 50% of the prioritized (highest-ranking) vote. If not, it must move the votes from the last place candidate to each of their voters' next highest-ranked candidate, and recheck for a 50% or more vote-getter, repeated until a single winning candidate remains. This is a high priority requirement, as determining the winner of an election is the point of an election.

4.9.2 Stimulus/Response Sequences

1. Check voting percentages for candidates
2. If a candidate is over 50%, declare the winner
3. If the leading candidate is under 50%, reallocate votes and check again

4.9.3 Functional Requirements

Name	Determining Winner (Instant Runoff)
ID	VS_09
Description	Upon the completion of parsing the election file given to the system, it must be determined if one candidate reached over 50% of the vote. If not, the election proceeds with votes being transferred from the last place candidate to their voters second choice candidate.
Actors	Election officials, testers
Organizational Benefits	Allows for the election results for an Instant Runoff election to be determined without having to hand count votes, or hand calculate runoffs
Frequency of Use	Anytime an election is called and it is determined that it is an Instant Runoff election
Triggers	When the file given to the program is determined to be an Instant Runoff election file
Preconditions	● An election has been called ● Voting has taken place ● A valid election file has been created and

	given to the system ● The file was determined to be of the Instant Runoff type
Postconditions	An election winner will have been determined
Main Course	1. Check if any candidate has received over 50% of the votes 2. If one has, declare them the winner 3. If not, proceed to alternate course
Alternate Courses	No candidate has received more than 50% of the vote: 1. Give votes from the ballots of the last place candidate to their second choice candidate 2. Back to Main Course
Exceptions	1. The election file does not exist or is in the wrong directory

4.10 Determining Winner (Open Party)

4.10.1 Description and Priority

Determines the winners of seats in an Open Party Voting election. The leading vote getters from the party that wins seats are determined and the remainder from each party is calculated to distribute the remaining seats.

4.10.2 Stimulus/Response Sequences

1. Check voting for candidates
2. Distribute seats based on Open Party Listing quota
3. Use remainders to determine any leftover seats if necessary

4.10.3 Functional Requirements

Name	Determining Winner (Open Party)
ID	VS_10
Description	A quota is set based upon the number of votes cast and seats to be filled. Voting statistics for each candidate are used to determine the number of seats each party gains, with top vote-earners filling the

	seats first. Remaining votes are compared across parties, with the party or parties with the most remaining votes winning those seats.
Actors	Election officials or testers
Organizational Benefits	Determines the actual outcome of the election, which is the purpose of this software.
Frequency of Use	Anytime an election of the Open Party Listing type is called.
Triggers	An election file of type Open Party Listing is given to the software.
Preconditions	1. The file has been parsed for the proper data 2. The data has been adequately and accurately collected and stored 3. The file has been formatted correctly
Postconditions	1. The seats up for election will have been allocated
Main Course	1. Election is called 2. Ballots are cast and the election file is given to the system 3. System calculates vote quota for each seat and determines votes for each candidate 4. Candidate data is used to determine who is given which seat
Alternate Courses	N/A
Exceptions	1. The election file is missing, in the wrong directory, or formatted incorrectly

4.11 Breaking tie votes

4.11.1 Description and Priority

If there is a tie in votes when determining a winner a random number generator will pick the winner. This is a medium priority function, as a tie is highly unlikely but still possible.

4.11.2 Stimulus/Response Sequences

1. Select the two candidates who have the same number of votes
2. Call a function that randomly selects one of the candidates

3. Eliminate the loser and move forward with the winner

4.11.3 Functional Requirements

Name	Breaking Tie Votes
ID	VS_11
Description	If there is a tie a winner will be selected by a random number generator.
Actors	Programmers
Organizational Benefits	Solve the issue of a tie with a random number generator
Frequency of Use	Whenever the program counts all votes correctly and there is a tie
Triggers	When there is a tie for the winner
Preconditions	All votes have been correctly counted
Postconditions	A winner is determined by the random number generator
Main Course	1. Randomly select a winner if the winning candidates have the exact same number of votes
Alternate Courses	If there are n number of candidates ties then the random number generator must pick from n numbers
Exceptions	1. The program is terminated for the random number generator has selected a winner

4.12 Creating Media File

4.21.1 Description and Priority

The media may request the results of an election, which will be given to them in the form of a text file.

4.12.2 Stimulus/Response Sequences

1. Media requests election results
2. System generates media file
3. Media file delivered to media

4.12.3 Functional Requirements

Name	Creating Media File
ID	VS_12
Description	The media can request the results of an election and general statistics in the form of a text file
Actors	Election officials
Organizational Benefits	Provides a way to display the outcome of the election to the public via media coverage
Frequency of Use	Anytime the media requests the statistics from a completed election
Triggers	The media contacts election officials and asks for the election results
Preconditions	The election must be completed and general statistics, such as the winner and their percentage of votes, as well as votes for other candidates, must be ready. These would be found in an audit file.
Postconditions	The media will have a text file containing general election information that is separate from the audit or election files. Those will remain unchanged.
Main Course	1. Media requests file 2. File is created 3. File is delivered to the media
Alternate Courses	N/A
Exceptions	1. Election is not finished, in which case they cannot request results 2. Failure to create the text file for the media

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system must be able to count 100,000 ballots in under 8 minutes. This is to ensure ballots are counted in a timely fashion and to get results to the public as fast as possible.

5.2 Safety Requirements

Due to the nature of the program, there are no safety requirements that need to be satisfied.

5.3 Security Requirements

There are no security requirements. All security will be handled at the time of vote collection.

5.4 Software Quality Attributes

This system was designed to last many years and be used in many different elections. Keeping this in mind, the primary goal was to ensure that the codebase was maintainable, portable, and precise.

Using object-oriented design, the code will be written in a general manner such that code maintenance is relatively easy. This will be accomplished through clear and concise classes, polymorphism, and proper member methods.

As for the portability of the codebase, since it is expected for this system to be used for elections of a wide variety while having no guarantee that the election officials will have access to a Linux/Unix machine; as such, Java was selected as the language of choice. Java's ability to be compiled and run on any different operating system with the same results was a large factor in its choosing and will make the software much more accessible.

Finally, in order for the program to have any credibility whatsoever, the system must correctly count the ballots in order to prevent election fraud. As such, careful and methodical planning of the counting algorithm will go into the code design such that no fraudulent activity is permitted.

5.5 Business Rules

The system will follow any laws and regulations the location of the election has.

6. Other Requirements

Appendix A: Glossary

JVM: The Java Virtual Machine. The JVM is installed alongside Java and converts Java programs into Java bytecode that can be run regardless of operating system.

RAM: Random access memory. Space in which short-term memory is stored by programs available for usage, but deleted when no longer necessary. Allows for quicker access time than retrieval from the central drive.

IR: Instant Runoff voting, as described in Section 4.9.1.

OPL: Open Party List voting, as described in Section 4.10.1.

Appendix B: Analysis Models

At this time, there are no models thought to be pertinent.

Appendix C: To Be Determined List

At this time, there is nothing yet to be determined.