Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Maria Zavala, Aaron Raines,

Camden Fessler, Ethan Johnson

University of Minnesota

February 3, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Aaron Raines	2/3/2023	Starting Introduction Section	1.0
Maria Zavala	2/3/2023	Starting Introduction Section	1.0
Camden Fessler	2/3/2023	Starting Introduction Section	1.0
Ethan Johnson	2/3/2023	Starting Introduction Section	1.0
Camden Fessler	2/8/2023	Working on Listing all Use Cases	1.01
Ethan Johnson	2/8/2023	Working on Listing all Use Cases	1.01
Maria Zavala	2/8/2023	Working on Use Cases	1.01
Aaron Raines	2/8/2023	Working on Use Cases	1.01
Aaron Raines	2/9/2023	Worked on Section 2	1.02
Maria Zavala	2/10/2023	Worked on Section 5 and Glossary	1.02
Ethan Johnson	2/12/2023	Working on Section 3	1.02
Camden Fessler	2/13/2023	Wrote Rough Draft for Section 4	1.03
Camden Fessler	2/14/2023	Added Details for Section 4	1.04
Maria Zavala	2/14/2023	Added Details for Section 4	1.04
Camden Fessler	2/15/2023	Finishing up SRS	1.05
Aaron Raines	2/15/2023	Finishing up SRS	1.05

Ethan Johnson	2/15/2023	Finishing SRS	1.05
Aaron Raines	2/16/2023	Checked and corrected some final details	1.06

1. Introduction

1.1 Purpose

The purpose of this document is to describe a system that is capable of processing voting of the following two types of voting: Instant Runoff Voting (plurality/majority type) and Party List Voting using Closed Party List (proportional voting type). It explains the purpose, system features, user interface, and constraints of the system. This document is intended for voting officials and developers of the system.

1.2 Document Conventions

This document will follow the IEEE template for System Requirement Specification Documents.

1.3 Intended Audience and Reading Suggestions

Typical Users: Voting officials, who will use the system for determining elections, Suggested Sequence for reading:

1 Introduction
2.1 Product Perspective
2.2 Product functions
2.6 User Documentation
3.1 User Interfaces
4 System Features
5.5 Buisness Rules

Advanced/Professional Users: Testers, who would act as voting officials and test the validity of the system, Suggested Sequence for reading:

1 Introduction
2.1 Product Perspective
2.2 Product functions
2.6 User Documentation
4 System Features

Programmers: Software engineers who are initially creating the voting system and further improving it, Suggested Sequence for reading:

2.1 Product Perspective
2.2 Product functions
2.4 Operating Environment
2.5 Design and implementation Constraints
3 External Interface Requirements
4 System Features
5.1 Performance requirements
5.4 Software quality attributes

1.4 Product Scope

This system software will be used by election officials to determine the winner of an election. This system is capable of processing Instant Runoff (IR) voting and Closed Party List (CPL) voting given an input CSV file. This system assumes all CSV files are cleaned and formatted to a specified preference.

In the case of IR voting if a candidate received over 50% of the first choice he or she is declared elected. If no one receives 50% or more of the votes the candidate with the fewest votes is eliminated. The ballots of the defeated candidate then go to the candidate that was ranked next. This process is repeated until one of the candidates obtains 50% or more of the votes.

In the case of Closed Party List voting legislators are elected in large, multi-member districts. In this system, parties put up a list of candidates and voters indicate their preference for a party. There are two types of party list systems: closed list, where the party determines the order of the candidates, and open list, where voters have some degree of preference for candidates on the list.

The objective of this system will be to take a user-input election file and read the contents to establish an election winner. The benefit of this system is its capability of handling two types of voting (IR and CPL). The goals of this system are to give a consistent, accurate, and fast result of any given election file

1.5 References

- *IEEE Template for System Requirement Specification Documents:*
  - *https://goo.gl/nsUFwy*
- Description of voting algorithms
  - https://fairvote.org/
- Information about CSE Lab Machines at the University of Minnesota
  - https://cse.umn.edu/cseit/classrooms-labs
- Definitions for Glossary Terms
  - https://languages.oup.com/google-dictionary-en/

- https://www.google.com/webhp?hl=en&sa=X&ved=0ahUKEwirpOPIm5j9AhXllYkEHeegCuUQPAgJ
- https://en.wikipedia.org/wiki/Main_Page

- Use Cases located in separate document
  - https://github.umn.edu/umn-csci-5801-02-S23/repo-Team5/blob/main/SRS/UseCases_Team5.pdf

2. Overall Description

2.1 Product Perspective

This voting system is a follow-on member of a product family designed to automate the process of counting votes in an election. It is part of a larger voting system, the larger system requires the votes that have been received to be counted. The vote-counting system takes the input file generated from the voting machines using an interface designed for the voting officials and performs the calculation of the number of votes cast for each candidate. The system then outputs the results of the election via the user interface.

Vote Input	→	Vote Counting	→	Results Display

2.2 Product Functions

The major functions that the vote counting system must perform are

1. Data processing: Analyzing and processing data to determine the number of votes cast for each candidate.
2. Data reporting: Generating reports on the results of the election, including the number of votes cast for each candidate, the winner of the election, and the distribution of votes across different demographic groups.
3. Audit trail: Maintaining an audit trail of all activities and changes made to the data.
4. User interface: Providing a user-friendly text interface for managing and accessing the data.

2.3 User Classes and Characteristics

- Election Officials: They are responsible for overseeing the election process and ensuring that the voting is conducted fairly. They should have access to all the features of the system, including the ability to configure the voting parameters, manage voter lists, and tally the results.
- Auditors: They are responsible for auditing the election results to ensure that the results are accurate. They should have access to the election results and be able to perform audits to verify the accuracy of the results.
- Developers and programmers: The software engineer will need access to all parts of the system, including the source code, database, and administrative tools, to create/develop and maintain the software with further modification and improvements.
- Testers: They will be responsible for validating the functionality and performance of the system. They will need access to the testing environment and tools, including test scripts, test data, and bug-tracking systems, to perform their testing tasks. They may need to simulate different user scenarios and test different parts of the system, including user interfaces, database interactions, and system performance.

2.4 Operating Environment

The voting system is expected to operate in a controlled environment, with the necessary hardware, operating system, and software components required for its smooth functioning. The specific hardware platform will be the most up-to-date CSE lab machine and the operating system shown below. This system will be developed using the programming language Java. (Use the link provided in the Reference section 1.5 under "Information about CSE Lab Machines at the University of Minnesota" for more details.)

CSE lab machines and the operating systems:

Operating Systems: Windows 10:

- Computer Specifications:
    - Dell Precision T3420
    - Intel® Core™ i5 @ 3.4GHz (x4)
    - 64 GB RAM
- Computer Specifications:

    - Dell OptiPlex 9020
    - Intel® Core™ i7 @ 4.2GHz (x4)
    - 32 GB RAM
- Computer Specifications:
    - Lenovo Workstation TS P620
    - Threadripper Pro 5945WX 12 Core Processor
    - 32 GB RAM

Operating Systems: Ubuntu 20.04:

- Computer Specifications:
    - Dell Precision T3620
    - Intel® Core™ i5 @ 800Hz (x3)
    - 32 GB RAM
- Computer Specifications:
    - Dell Precision 3650 Tower
    - Intel Core i7 @ 2.5 GHz (x8)
    - 32 GB RAM

2.5 Design and Implementation Constraints

- Regulatory policies: There may be regulatory policies in place for the use of voting systems, however, none of these have been described to us therefore none will be considered beyond the requirements received.
- Interfaces to other applications: There should be no need to interact with other interface applications. The voting files containing all information necessary should be provided.
- Communications protocols: The system will need to communicate with users such as voting officials, testers, developers, and programmers in order to ensure that the system will be able to run flawlessly.
- Security considerations: According to the instructions received there are no special safety or security requirements.

- Design conventions or programming standards: The system will be programmed using Java and will communicate via just text with the user.

2.6 User Documentation

A PDF manual will be included with the system which would include:

- A brief overview of the voting system and its purpose.
- A detailed description of the user interface, including navigation, buttons, and menus.
- A step-by-step guide on how to input and manage data, as well as setting up voting sessions.
- A description of the steps involved in running a voting session, including the file input, determining what kind of voting is taking place, and displaying the results.
- A description of the reports that can be generated and how to access them.

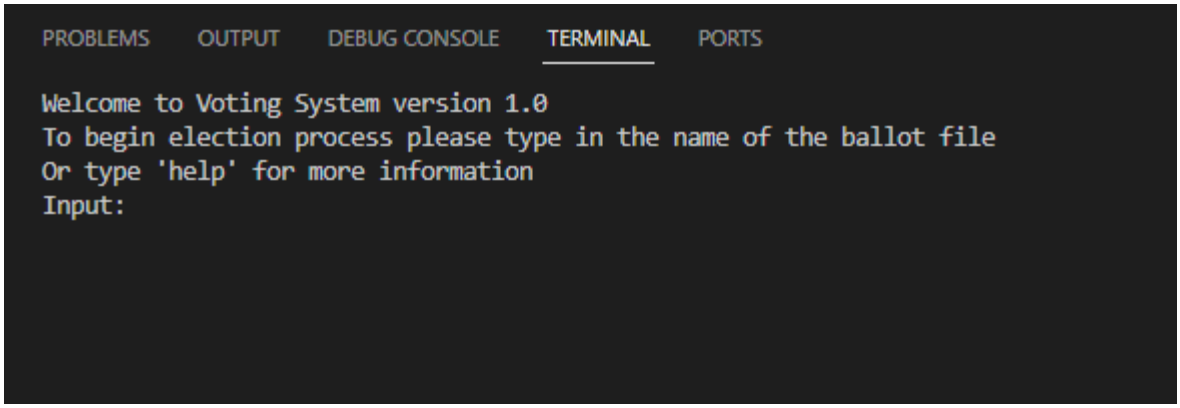
2.7 Assumptions and Dependencies

At this time there are no assumptions as to what could affect the requirements stated in the SRS. Unless the requirements for the voting system change, the project should not face any negative impacts.

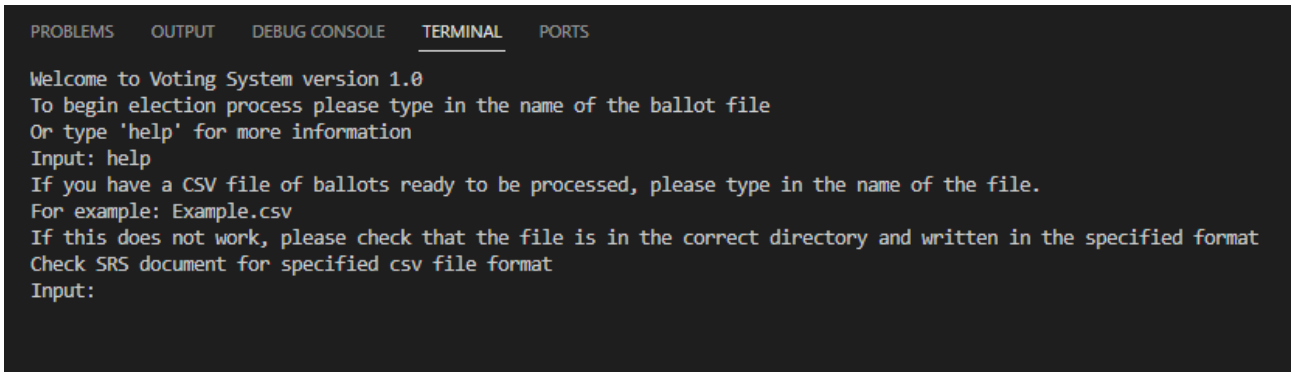
3. External Interface Requirements

3.1 User Interfaces

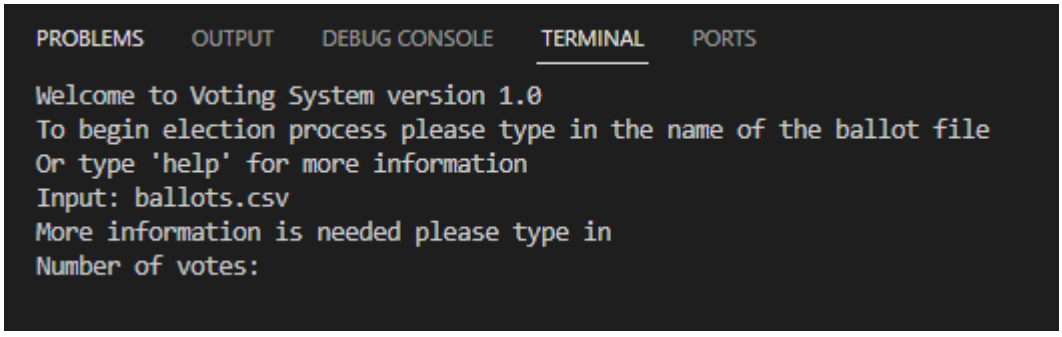
Once the system has been started the user interface will open through the terminal. It begins by welcoming the user and then prompting the user:
The user will first be prompted to either enter in a CSV ballot file or a help option

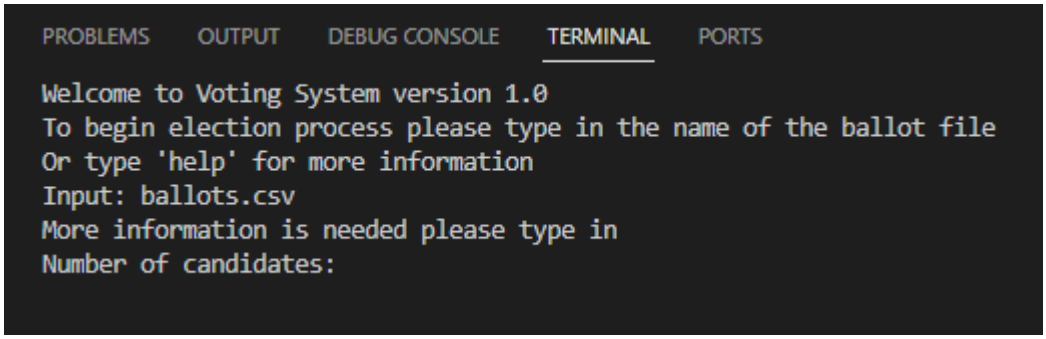


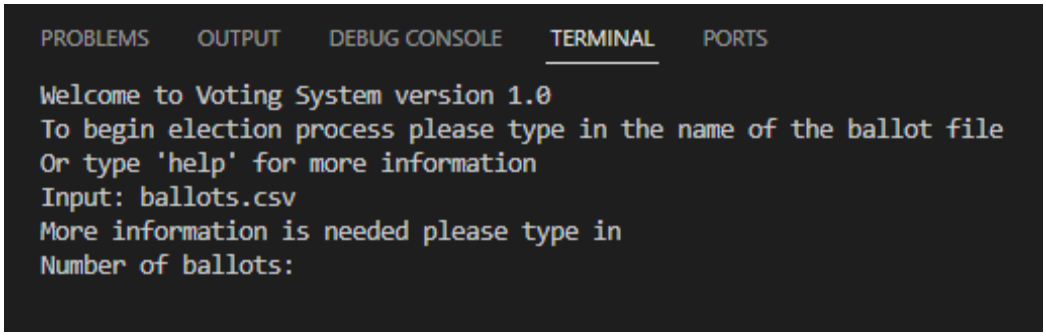
Inputting “Help” will print all possible input commands to the screen

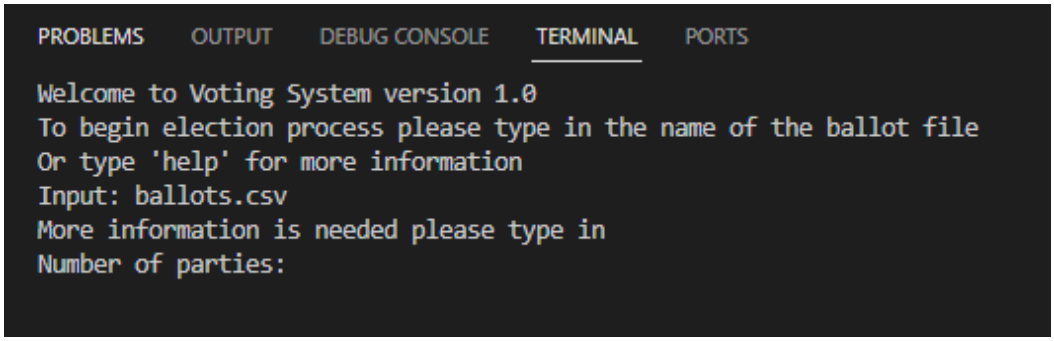


If the user inputs a CSV file, the file will be read and will prompt the user for any more needed information such as: Number of votes, candidates, ballots or parties.









- Once the system gets all necessary information it will run and print the winner to the screen
- It will then ask for more input for a new election until the user closes the system

The user interface is constrained to the terminal and there is no GUI. If the system is given false input the help screen will be displayed and if any errors occur during run time an error message will be printed to the screen with a detailed explanation of what went wrong.

3.2 Hardware Interfaces

The hardware used to create this system will be the CSE labs machine computers as follows:

- Computer Specifications:
    - Dell Precision T3620
    - Intel® Core™ i5 @ 800Hz (x3)
    - 32 GB RAM
    - Dell Precision 3650 Tower
    - Intel Core i7 @ 2.5 GHz (x8)
    - 32 GB RAM

- Dell Precision T3420
- Intel® Core™ i5 @ 3.4GHz (x4)
- 64 GB RAM
- Dell OptiPlex 9020
- Intel® Core™ i7 @ 4.2GHz (x4)
- 32 GB RAM

Refer to Section 1.5 for "Information about CSE Lab Machines at the University of Minnesota" for detailed hardware/software specifications.

3.3 Software Interfaces

Voting System 1.0 needs Java to be installed on the system, specifically Java version 7 or 8. Therefore this system requires an operating system that supports Java: Windows, Mac OS, and the various versions of UNIX/Linux like HP-Unix, Sun Solaris, Redhat Linux, Ubuntu, CentOS, etc. Refer to Section 1.5 for "Java System Requirements" for detailed hardware/software specifications.

3.4 Communications Interfaces

This Voting System will be shared among programmers on Github and will possibly be in an open source repository in the future.

4. System Features

This section references the use cases documented in the file named UseCases_Team5.

4.1 Prompting a User for a File

ID: UC_01

4.1.1 Description and Priority

Users will be prompted by the system to enter the name of the election file to be processed. There will be a message displayed to the user stating “Enter the name of the file containing the ballots: " and the user will be allowed to type the name of a file in that same directory they want processed. There will never be more than one file given per election, and this file’s structure cannot be changed during the course of the program’s runtime. This file was exported from Excel and all preprocessing of the file will have been done beforehand. This is a High Priority feature because this is the main way for election files to be read in by the program.

4.1.2 Stimulus/Response Sequences

Main Course:

1. The voting system is run (started).
2. The system checks if it has command line arguments for the file name. (AC1)
3. If not, the system then displays the prompt:

   Enter the name of the file containing the ballots:
4. The user enters a valid file name. (AC2)
5. The name gets saved in the program for later use.

Alternate Course 1 (AC1):

1. If there is already a command line argument for the file name, do not display the prompt.
2. The program then saves the name from the command line argument.

Alternate Course 2 (AC2):

1. If the file name given by the user is not a valid file name, output an error message.
2. Prompt the user once again to enter a correct file name.

4.1.3 Functional Requirements

REQ-1: The program must have been started by the user manually.

REQ-2: There must be some way to display text to the user.

REQ-3: There must be some way for the user to type in the name of a file.

REQ-4: If there is an error entering the file name or if the input is invalid, the program must display a message to the user explaining what went wrong.

REQ-5: There must be a way to save the name of the file inputted by the user for future use by the program.

4.2 Checking if the File Exists

ID: UC_02

4.2.1 Description and Priority

After obtaining a valid file name from the user, the program will check to see if the file exists in the same directory as the program. If the file does exist, the program will also check if the file is the correct type, that being a comma-separated value file. This is a High Priority feature because the program cannot do what was intended if the file does not exist or if it is not in the correct format.

4.2.2 Stimulus/Response Sequences

Main Course:

1. File name from the user is searched for in the working directory.
2. Check if the file is NULL. (AC1)
3. If not, then check if the file is a comma separated value file. (AC2)

Alternate Course 1 (AC1):

1. If the file is NULL (does not exist) then print an error message to screen telling the user the file does not exist, then prompt the user to enter a filename again.

Alternate Course 2 (AC2):

1. If the file is not a CSV file, print an error message to the screen telling the user the file is not the correct format, then prompt the user to enter a filename again

Exception 1 (EX1):

1. There is no need to run a file to be processed

4.2.3 Functional Requirements

REQ-1: The program must be currently running.

REQ-2: The file name inputted by the user must be accessible.

REQ-3: The program must be able to check the working directory to see if the file exists.

REQ-4: The program must be able to check if the file is in the correct format.

REQ-5: The program must be able to prompt the user to input another file name if there is an error.

4.3 Determining the Correct Election Type

ID: UC_03

4.3.1 Description and Priority

Once the file inputted by the user is confirmed to exist and be of the correct type, the file is then opened for reading. The program then determines whether the file corresponds with an Instant Runoff (IR) election or a Closed Party List (CPL) election. This information is then stored for future use in the audit file. This is a High Priority feature because if the election type cannot be determined, the program will not be able to determine the winner of the election.

4.3.2 Stimulus/Response Sequences

Main Course:

1. Check if there is an error opening the file. (EX1)
2. Open the file for reading.
3. Read the first line of the CSV file.
4. Check if the first line is either ‘IR’ or ‘CPL’. (AC1)
5. Store the type of election for later use.

Alternate Course 1 (AC1):

1. Close the file and stop all processes in the program. Print out an error message to the user and prompt them to enter a filename again.

Exception 1 (EX1):

1. Stop all processes in the program. Print out an error message to the user and prompt them to enter a filename again.

4.3.3 Functional Requirements

REQ-1: The file must exist and be the correct format.

REQ-2: There must be no errors when opening the file for reading.

REQ-3: There must be a way to read lines of a CSV file.

REQ-4: There must be a way to end the process and start again if an error occurs.

REQ-5: There must be a way to store the election type for future use.

4.4 Saving Instant Runoff Election Information

ID: UC_04

4.4.1 Description and Priority

Once the type of election has been determined to be IR, the program will then read the file for more information, including the number of candidates, candidates' names, and the number of total ballots. This information is then stored for future use in the election audit file. If certain information is not found by scanning the header of the file, the user will be prompted to enter in the information manually. This is a High Priority feature because this information needs to be in the audit file produced at the end of processing the election file.

4.4.2 Stimulus/Response Sequences

Main Course:

1. The program saves the number of candidates
2. The program saves the names of the candidates
3. The program saves the number of ballots

Alternate Course 1 (AC1):

1. If any information is not found by scanning the header, the user will be prompted to enter the information into the program manually.

Exception 1 (EX1):

1. Stop all processes in the program. Print out an error message to the user and prompt them to enter a filename again.

4.4.3 Functional Requirements

REQ-1: The type of election must be known to be IR by the program.

REQ-2: The file must be readable by the program.

REQ-3: The information obtained from reading the file must be stored for future use.

REQ-4: The program must be able to stop all processes and produce an error message if an error occurs.

4.5 Saving Instant Runoff Ballot Information

ID: UC_05

4.5.1 Description and Priority

After processing the election information from the file, the ballots are read through in the file and saved. These ballots were casted online by voters and are saved in a CSV file for the program. The ballot information will be used in the future when counting the ballots and determining the winner of the election. All of the ballots will have at least one person ranked on them with no write in candidates allowed. This is a High Priority feature because the ballots need to be saved in a way that the program can read and count them in the future.

4.5.2 Stimulus/Response Sequences

Main Course:

1. Create a data structure to store ballot information
2. Read every ballot information in the opened CSV file starting from the line where ballots are located.
3. Save information to program

Alternate Course 1 (AC1):

1. If the system fails to save information, Indicate an error occurred and attempt to run again.

Exception 1 (EX1):

1. Ballot is incomplete/missing information

4.5.3 Functional Requirements

REQ-1: A data structure must be able to store all of the ballot information.

REQ-2: The program must be able to stop and produce an error message if an error occurs when reading or saving the information.

REQ-3: The file must be known to be for an IR election.

4.6 Run IR Algorithm

ID: UC_06

4.6.1 Description and Priority

After the ballot information has been saved, the program will read through the information and count how many votes each candidate received. This information will be used to determine the winner of the election in the future, given there is not a tie. This is a High Priority Feature because this is vital for determining the result of a particular election.

4.6.2 Stimulus/Response Sequences

Main Course:

Part 1 of algorithm: Processing Ballots

1. Look at each individual ballot
2. Process the ranking of each ballot.
3. Once all ballots have been processed the current state is saved

Part 2 of algorithm: Ranking candidates

4. Count the number of ballots each candidate has received
5. Set a ranking for each candidate

Part 3 of algorithm: Determine winner

6. Look at each candidate ranking
7. The candidate who receives over 50% of the first-choice votes, is declared the winner.

Alternate Course 1 (AC1):

Run part 1 of algorithm;

Run part 2 of algorithm;

Part 3 of algorithm: Determine winner

1. Look at each candidate ranking
2. There is no candidate with a clear majority

Part 4 of algorithm: Drop candidate

3. Select the candidate ranked last
4. Now run the IR algorithm for the ballots that were given to the selected candidate.
5. This candidate is now dropped.

Alternate Course 1 (AC1):

1. When getting to ranking part of the algorithm there is a tie between 1 or more candidates
2. In this case refer to UC_14

Exception 1 (EX1):

1. Ballot is incomplete/missing information

Exception 2 (EX2):

2. Candidates information is incomplete/missing.

4.6.3 Functional Requirements

REQ-1: The ballots in the saved data structure must be readable.

REQ-2: The program must be able to parse through each ballot and determine which candidate to assign it to.

REQ-3: The results of the counting must be stored to be used in determining the winner.

REQ-4: The program must be able to rank the candidates in order of most ballots to least.

REQ-5: The program must be able to drop a candidate if there is no clear majority.

REQ-6: The program must be able to fairly determine a winner if the number of votes is tied.

REQ-7: There must be a way to reassign ballots to different candidates.

REQ-8: The dropped candidate must be removed from the rankings.

REQ-9: The program must stop and print out an error message if an error occurs.

4.7 Naming IR Audit

ID: UC_07

4.7.1 Description and Priority

After the winners of an IR election are determined, a text file is created and stored in the same directory as the election file. This file will be named IR_Audit.txt and will be used to store election information in the future. This is a High priority feature because this is how the audit file is originally created by the program.

4.7.2 Stimulus/Response Sequences

Main Course:

1. Create a .txt file in the same working directory as the program.
2. Name the file IR_Audit.txt and leave it open for editing.

Alternate Course 1 (AC1):

1. Create a file called IR_Audit.txt
2. Make sure to check that the audit file has been created
3. In this case it failed and it alerts the user of error
4. Retry to create a file

Exception 1 (EX1):

1. The is no need to create an IR audit file

4.7.3 Functional Requirements

REQ-1: The winner of the election has been determined.

REQ-2: The program is able to create text files in the directory.

REQ-3: If any error occurs, the program must stop and print an error message.

4.8 Producing an Audit File for an Instant Runoff Election

ID: UC_08

4.8.1 Description and Priority

After the audit file is created and named, the program stores information about the election in it. This information includes the number of candidates, number of ballots, names of each candidate, 1st count: original first choices, 2nd count: transfer of disqualified candidate and new total, 3rd count: transfer of disqualified candidate and new totals, and so on, and the winner. This is a High Priority feature because this is the way that Election Officials will be able to see exactly what happened during the processing of the election.

4.8.2 Stimulus/Response Sequences

Main Course:

1. Write the number of candidates who were in the election.
2. Write the number of ballots cast for the election.
3. Write the names of each of the candidates in the election.
4. Write next to the names of the first count results from the election.
5. Write other needed information such as the date of the election.
6. Output the winner at the end of the file.

Alternate Course 1 (AC1):

1. Write the number of candidates who were in the election.
2. Write the number of ballots cast for the election.
3. Write the names of each of the candidates in the election.
4. Write next to the names of the first count results from the election.
5. If there was more than one round, write next to the first count the results of the next count from the election.
6. Write other needed information such as the date of the election.
7. Output the winner at the end of the file.

Exception 1 (EX1):

1. There was no clear winner at the end.

4.8.3 Functional Requirements

REQ-1: The program must be able to create a text file.

REQ-2: The program must be able to write information into a text file.

REQ-3: The program must be able to retrieve information about the election.

REQ-4: The program must stop and print out an error message if any error occurs.

4.9 Saving Closed Party List Election Information

ID: UC_09

4.9.1 Description and Priority

Once the type of election has been determined to be CPL, the program will then read the file for more information, including the number of parties, the parties' names, the number of seats available, and the number of total ballots. This information is then stored for future use in the election audit file. If certain information is not found by scanning the header of the file, the user will be prompted to enter in the information manually. This is a High Priority feature because this information needs to be in the audit file produced at the end of processing the election file.

4.9.2 Stimulus/Response Sequences

Main Course:

1. The program reads in line 2 in the file. (EX1)
2. The program sets the number of parties from line 2. (AC1)
3. The program reads in line 3. (EX1)
4. The program sets the names of the parties. (AC1)
5. The program reads in line 4. (EX1)
6. The program sets the number of seats. (AC1)
7. The program reads in line 5. (EX1)
8. The program sets the number of ballots. (AC1)

Alternate Course 1 (AC1):

1. If any information is not found by scanning the header, the user will be prompted to enter the information into the program manually.

Exception 1 (EX1):

1. Stop all processes in the program. Print out an error message to the user and prompt them to enter a filename again.

4.9.3 Functional Requirements

REQ-1: The type of election must be known to be CPL by the program.

REQ-2: The file must be readable by the program.

REQ-3: The information obtained from reading the file must be stored for future use.

REQ-4: The program must be able to stop all processes and produce an error message if an error occurs.

4.10 Saving Closed Party List Ballot Information

ID: UC_10

4.10.1 Description and Priority

After processing the election information from the file, the ballots are read through in the file and saved. This information will be used in the future when counting the ballots and determining the winner of the election. There will be no write-in candidates for any of the CPL ballots. This is a High Priority feature because the ballots need to be saved in a way that the program can read and count them in the future.

4.10.2 Stimulus/Response Sequences

Main Course:

1. Create a data structure to store ballot information.
2. Read ballot information in the opened CSV file starting from line 5.
3. Save information to data structure.
4. Parse to the next ballot in file.
5. Continue for all ballots.

Alternate Course 1 (AC1):

4.10.3 Functional Requirements

REQ-1: A data structure must be able to store all of the ballot information.

REQ-2: The program must be able to stop and produce an error message if an error occurs when reading or saving the information.

REQ-3: The file must be known to be for an IR election.

4.11 Run Closed Party List Algorithm

ID: UC_11

4.11.1 Description and Priority

After the ballot information has been saved, the program will read through the information and count how many votes each candidate received. After the ballot information is gathered, the program will assign candidates to seats according to the amount of votes each party has received. This information will be used to determine the winner of the election in the future, given there is not a tie. This is a High Priority Feature because this is vital for determining the result of a particular election.

4.11.2 Stimulus/Response Sequences

Main Course:

Part 1 of algorithm: Processing Ballots

1. Look at each individual ballot
2. Process the ranking of each ballot.
3. Once all ballots have been processed the current state is saved

Part 2 of algorithm: Ranking candidates

1. Count the number of ballots each candidate has received
2. Set a ranking for each candidate

Part 3 of algorithm: Determine winner

1. Look at each candidate ranking
2. The candidate who receives over 50% of the first-choice votes, they are declared the winner.

Alternate Course 1 (AC1):

Run part 1 of algorithm;

Part 2 of algorithm: Determine Seats for each party

1. Count the number of ballots each Party has received

2. Parties receive seats depending on the percentage of votes determined by the CPL algorithm equation.
3. There is a tie between 2 or more parties when dividing seats see UC_14

Exception 1 (EX1):

1. Ballot is incomplete/missing information

4.11.3 Functional Requirements

REQ-1: The ballots in the saved data structure must be readable.

REQ-2: The program must be able to parse through each ballot and determine which candidate to assign it to.

REQ-3: The results of the counting must be stored to be used in determining the winner.

REQ-4: The program must be able to stop and print an error message if any error occurs.

4.12 Naming CPL Audit

ID: UC_12

4.12.1 Description and Priority

After the winners of a CPL election are determined, a text file is created and stored in the same directory as the election file. This file will be named CPL_Audit.txt and will be used to store election information in the future. This is a High priority feature because this is how the audit file is originally created by the program.

4.12.2 Stimulus/Response Sequences

Main Course:

1. Create a .txt file in the same working directory as the program.
2. Name the file CPL_Audit.txt and leave it open for editing.

Alternate Course 1 (AC1):

1. Create a file called CPL_Audit.txt
2. Make sure to check that the audit file has been created
3. In this case it failed and it alerts the user of error
4. Retry to create a file

Exception 1 (EX1):

2. The is no need to create an IR audit file

4.12.3 Functional Requirements

REQ-1: The winners of the election has been determined.

REQ-2: The program is able to create text files in the directory.

REQ-3: If any error occurs, the program must stop and print an error message.

4.13 Producing an Audit File for a Closed Party List Election

ID: UC_13

4.13.1 Description and Priority

After the audit file is created and named, the program stores information about the election in it. This information includes the number of parties, number of ballots, names of each party, number of ballots for each party, 1st allocation of seats, Remaining votes, second allocation of seats, final seat total, % of the vote to % of seats, and winner(s). This is a High Priority feature because this is the way that Election Officials will be able to see exactly what happened during the processing of the election.

4.13.2 Stimulus/Response Sequences

Main Course:

1. Write the number of parties who were in the election.
2. Write the number of ballots cast for the election.
3. Write the names of each of the parties in the election.
4. Write the number of ballots for each party.
5. Write the first allocation of seats.
6. Write the remaining votes after the first allocation.
7. Write the second allocation of seats.
8. Write other needed information such as the date of the election.
9. Output the winner at the end of the file.

Alternate Course 1 (AC1):

1. Write the number of parties who were in the election.
2. Write the number of ballots cast for the election.
3. Write the names of each of the parties in the election.
4. Write the number of ballots for each party.
5. Write the first allocation of seats.

6. Write the remaining votes after the first allocation.
7. Write the second allocation of seats.
8. Write if there was a tie between parties.
9. Write other needed information such as the date of the election.
10. Output the winner at the end of the file.

4.13.3 Functional Requirements

REQ-1: The program must be able to find the audit file.

REQ-2: The program must be able to write information into a text file.

REQ-3: The program must be able to retrieve information about the election.

REQ-4: The program must stop and print out an error message if any error occurs.

4.14 Determining Winner in Case of a Tie

ID: UC_14

4.14.1 Description and Priority

In the case of a tie, the program will flip a coin to determine the winner of an election, or in the case of more than two candidates, randomly decide who the winner is. This will be done by a random number generator deciding the winner, with each candidate being assigned a different number. This is a Medium Priority feature because ties will rarely happen during the course of an election.

4.14.2 Stimulus/Response Sequences

Main Course:

1. If more than two candidates are tied, use the alternative method. (AC1)
2. The system uses an algorithm that will choose one of two choices with an even 50-50 chance.
3. Assign the two candidates to the algorithm, and use it to choose one winner.
4. Save this winner in the election audit file.

Alternate Course 1 (AC1):

1. Assign each candidate a number to represent them, ranging from one to the number of tied candidates.

2. Use a random number generator with a range of one to the number of tied candidates to obtain a number.
3. Match this number with the candidate assigned to it to determine the winner of that particular election.
4. Save this winner in the election audit file.

4.14.3 Functional Requirements

REQ-1: The winner of the coin toss/pool coin toss must be chosen fairly.

REQ-2: The result will be saved for future use in the audit file.

REQ-3: The program must stop and print out an error message if any error occurs.

4.15 Displaying the Winners to the User

ID: UC_15

4.15.1 Description and Priority

After the file has been read and the votes have been tallied, the winner of the election is displayed to the user. This will be done with a text display using the same method as error messages or prompting the user. For an IR election, one candidate will be listed as the winner. For a CPL election, all of the candidates who fill the available seats are displayed as winners. This is a Low Priority feature because the result of the election is already stored in the audit file.

4.15.2 Stimulus/Response Sequences

Main Course:

1. Once the program is done determining the winner of the election, it will display the winning candidate's name(s) on the screen via text.
2. If the election was an Instant Runoff election, only one candidate will be displayed to the screen, that being the winner.
3. If the election was a Closed Party List election, all candidates who filled the open seats will be displayed on the screen.
4. Additional information about the election will also be displayed as well.

Alternate Course 1 (AC1):

1. The program confirms that the winner(s) have been determined.

2. It will display the winning candidate's name on the screen via text.
3. Error occurs when displaying information, printing an error message to the user.

4.15.3 Functional Requirements

REQ-1: The file has been read and the votes have been tallied.

REQ-2: The program has determined the winner of the election, either by votes or coin flip.

REQ-3: The program can display text to the user.

REQ-4: The program must stop and print out an error message if any error occurs.

4.16 Help Feature

ID: UC_16

4.16.1 Description and Priority

After the user has started the system, if they do not know what to do they can type in "help" to the input to be given a more detailed description of what to do.

4.16.2 Stimulus/Response Sequences

Main Course:

1. The user inputs "help"
2. The help message is printed to the screen.

4.16.3 Functional Requirements

REQ-1: The system has been started and prompted the user for input.

REQ-2: The user's input "help" has been read and the help message has been printed to the screen.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Voting system should run 100,000 ballots in under 4 minutes. The voting system needs to be efficient in counting ballots as determining the winner for elections has to be an improvement as opposed to processing ballots by hand in IR and CPL algorithms.

5.2 Safety Requirements

At this time there are no special safety requirements indicated by the users for this voting system. Unless the requirements for the voting system change per the user's request, the project should not face any safety concerns.

5.3 Security Requirements

At this time there are no security requirements indicated by the users for this voting system. Unless the requirements for the voting system change per the user's request, the project should not face any security impacts.
The following will be handled by voting centers and not by the system being developed:

- Ensuring there is one vote for one person
- Ensuring there are no errors in ballots.
- Ensuring that the CSV file is intact at the moment that it is processed by the system.
- Ensuring that the ballots and CSV files are handled by voting officials.

5.4 Software Quality Attributes

- The voting system will reliably run during the election times and during testing.
- The voting system must run on the most up-to-date CSE lab machine.
- This system will be developed using Java, In which the project will Include source files and class files used for the entire project.
    - This also includes running the Java project through either a command line or eclipse.

5.5 Business Rules

Voting:

- The actual voting process will be done separately from the system being developed for this project. Therefore there will be no voting in this system as of this current version.
- The voting system will be available multiple times during the year these times will be during
    - Normal election times
    - Special election times
- The voting system will be able to handle the two specified election algorithms:
    - Instant runoff
    - Closed party list
        - This means the system will not process open party list election algorithms
- There will be no write-in candidates for this current version of the voting system.
    - This means that if specified by the user in the future it may be implemented.

Instant runoff:

- If there is no clear majority after all votes have been distributed then the winner will be determined by a candidate having 50% or above.
- Each ballot will have at least one of the candidates ranked as their top choice.
- Instead of voting for one candidate, the voter is given the option to rank the candidates in order of preference.
    - Each candidate that is ranked can only be given 1 ranking.
    - The ranking will be from 1 to the number of candidates available

Closed party list voting:

- There will be a predetermined order of candidates under the specified parties.
- The winners will be calculated based on the number of seats and the order of these candidates.
- All independent groups will only have a single person in their group.
- There may be several independent groups which will have a unique name to identify that group.
  e.g. Independent1, Independent2, etc…

File:

- The voting system will provide a comma-delimited text file from ballots cast online.
- All of the preprocessing of the file will be done before it is received by the system.
- The first line of the file will contain the type of voting (i.e. IR, CPL)
- There will never be more than one file given to be processed by the system per election.
- The election file will be located in the same directory as the program.
- The election files provided to the voting system will have no error.
- The election files will come in a predetermined format and the system can not change the structure of the file being processed.
    - The system should be able to process this predetermined format.

Audit:

- There will be audit files created by the voting system once all calculations and winner has been determined.
- The format of the audit file will be .txt
- The audit file will display the calculations made by the specified algorithm
    - e.g. It will show the order of removal of candidates in IR and what ballots were redistributed.

Access:

The system should be developed by assigned programmers and developers. This class of users should have access to the requirements specifications in order to implement the system. Programmers and developers have full access to all of the functionalities in this system.

The system should be tested and accessed by the testers assigned to this project. The testers should have access to test files that simulate an election in order to assert the correctness of this program. The testers also have full access to every function in this voting system.

The system should be accessed and put into practice by voting officials. The voting officials should be able to access the actual voting files produced by voting centers. The voting officials should be able to access all of the functionalities in this system.

6. Other Requirements

The program will need a data structure to hold the information for each election such as the number of candidates, number of ballots, number of parties, names of candidates, and names of parties, calculations produced by IR and CPL algorithm.

The program will need a data structure to hold each ballot encountered in the election CSV file to later be processed by each election algorithm.

Appendix A: Glossary

Algorithm - a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

IR/IRV - Instant Runoff voting

CPL - Closed party list voting

Data structure - A storage format that is usually chosen for efficient access to data.

CSV - A file that contains comma separated values.

NULL - having no legal or binding force; invalid.

Command Line - This provides a means of setting parameters for invoking a program and providing information to it as to what actions it must perform.