Software Requirements Specification

for

CSCI 5801 Voting System

Version 1.9

Prepared by Team 25:

Josh Trimble, trim0039, Team25

Myat Mo, mo000007, Team25

Caden Potapenko, potap009, Team25

Roman Woolery, woole022, Team25

CSCI 5801, University of Minnesota

February 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Josh Trimble	2/14/21	Added all use cases to section 4 and filled in descriptions for half	1.0
Josh Trimble	2/15/21	Finished last half of the use cases for section 4	1.1
Josh Trimble	2/16/21	Added pointers in use cases with references to other use cases	1.2
Myat Mo	2/17/21	Added section 3 and 6	1.3
Roman Woolery	2/18/21	Added parts of section 1 and 5	1.4
Josh Trimble	2/19/21	Updated table of contents	1.5
Caden Potapenko	2/19/21	Adding section 2	1.6
Myat Mo	2/19/21	Update section 3 and example CSV files	1.7
Roman Woolery	2/19/21	Update sections 1 and 5	1.8
Josh Trimble	2/19/21	Updated Glossary with terms	1.9

1. Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements needed to operate voting software that runs both IR and OPL style elections on vote data.

The first section details the format of this document, recommendations to utilize the information in this document correctly, giving an introductory summary of the voting software's purpose that the SRS will detail.

Section 2 details the uses and functionality of the voting software. This includes charts of the software's components and how it is managed when running an election, what operating environment it runs on, and which individuals utilize the product for specific purposes.

Section 3 discusses the external interface of the voting software. It contains explanations of how to upload voter data, the voting system's management of specific election scenarios, how the software produces election results, and how to interpret various responses from the software during use. All the hardware requirements needed to run the voting software are also detailed.

Section 4 explains the product features in terms of its use cases that describe the software's functionality.

Section 5 outlines non-functional requirements of the product, such as performance, safety, security, quality, and business rules involved in permission.

Section 6 covers additional requirements not mentioned elsewhere in the document, such as the prerequisites for using the software and voting data organization requirements.

The glossary defines the terms used and components of the voting system referenced throughout this document.

1.2 Document Conventions

This document is based on the IEEE template for SRS documents.

1.3 Intended Audience and Reading Suggestions

The intended audience of this document includes election officials, programmers, and testers.
To understand this document and the voting system, readers should check Section 1.
To understand the software's functionality, read Sections 2 through 4.

Election workers may read this to understand how to format the voting data compatible with the voting system’s specifications. Section 5 contains non-functional requirements of the voting software, and section 6 contains additional formatting instructions for voter data.

1.4 Product Scope

The Voting System Software manages a voting system that processes election results compiled into a single CSV file and determines an election’s outcome based on this data. Election officials can share the software’s election results with the media after the election count completes.

This voting system software is designed to implement two voting algorithms: Instant Runoff Voting (IR) and Open Party List Voting (OPL).

1.5 References

IEEE Template for System Requirement Specification Documents: IEEE Software Requirements Specification Template
Referenced in sub-section 1.2.

2. Overall Description

2.1 Product Perspective

This product will be replacing the older/analog voting systems with a computerized system to manage elections. This product will connect with several other sub-systems/databases, including the Online Ballot System, the Ballot Database, and the Vote Count system. See Figure 1.

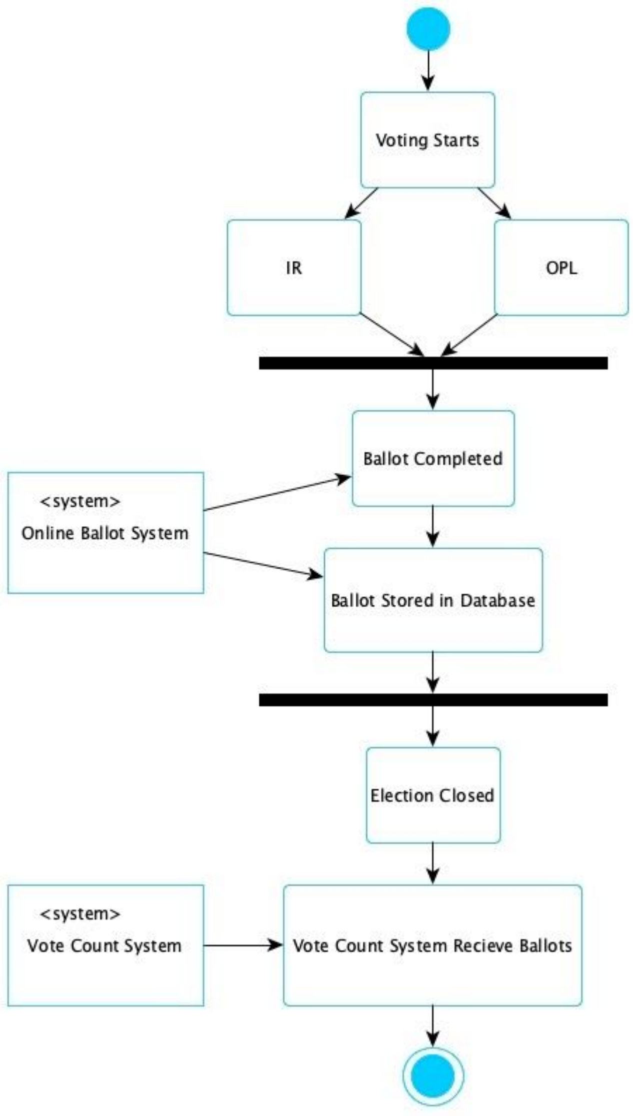


Figure 1. Major components of the system

2.2 Product Functions

The product's primary functions are as follows:

- Prompt the user for election type (Instant Runoff vs. Open Party List)
  - This will be a text-based user interface.

- Upload formatted CSV file
  - This file must be pre-formatted to work successfully, including having the first line of the file match the election type per requirements.

- Counting votes
  - Each vote will be counted, and an on-screen message will be displayed to inform the user during this step.
  - Determine the distribution of votes across candidates and track the majority
  - This may cause a redistribution of vote counts.

- Display vote count
  - There are several cases for how this will occur; see Section 3.1 User Interface.

- Saving Election Results
  - Save the result/exit file on the local computer and share the result with other authorized users.

2.3 User Classes and Characteristics

Several user classes will likely use this product

- Election Officials - Highest importance
  - These members would be given certain privilege levels to handle election ballots and would likely have training in using the product being developed.
  - These members would have access to the complete functionality of the product.

- Programmer/Tester
  - These members would have full read/write access to the program itself
  - These members would not have the privilege to handle official election ballots themselves

2.4 Operating Environment

Operating Systems Requirement

- Windows 7
- Windows 8
- Windows 10
- Mac OS X 10.14 or newer
- Linux Ubuntu 18.04 or newer

Hardware Systems Requirement

- Minimum System Requirements
    - RAM: 16GB or more
    - Processor: Intel Core i3 550 3.2 GHz or equivalent
    - Graphics: Intel 7490 or equivalent
- Recommended System Requirements
    - RAM: 32GB or more
    - Processor: Intel Core i5 3.2 GHz or equivalent
    - 2 x NVIDIA® GeForce GT 755M 1024MB or equivalent

2.5 Design and Implementation Constraints

The program must handle processing 100,000 results in 8 minutes or less (see Section 5.1). The input data (CSV files) must be pre-formatted (see Section 6). The security of data will be handled by the company itself, not this program.

2.6 User Documentation

There will be instructions delivered with the software on how to pre-format the data, navigate through the text-based user interface, and the software required to run this program.

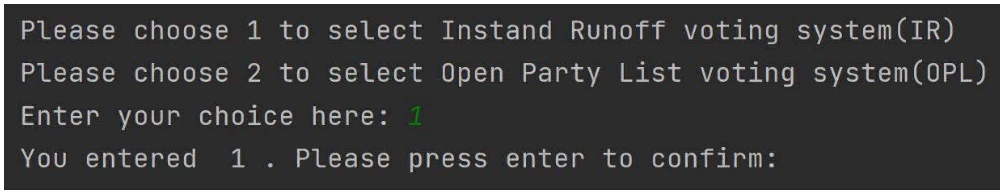
2.7 Assumptions and Dependencies

It is assumed that the computer will have the necessary software pre-downloaded to run these programs. It is assumed that the user running this program will know about its functionality and essential formatting.

3. External Interface Requirements

3.1 User Interfaces

This section illustrates the two voting systems' user interfaces: Instant Runoff voting system (IR) and Open Party List voting system (OPL). Since we are going to implement this project in C/C++, this will be the text-based user interface. Our voting system starts by prompting the user to choose between Instant Runoff voting system or Open Party List voting system.



```
Please choose 1 to select Instand Runoff voting system(IR)
Please choose 2 to select Open Party List voting system(OPL)
Enter your choice here: 1
You entered  1 . Please press enter to confirm:
```

User input for choosing the type of voting system

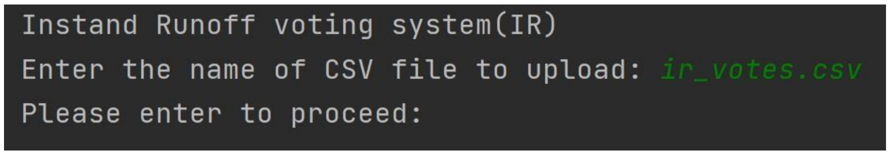
Instant Runoff (IR) Voting Interface

Uploading CSV file

The system asks to provide the name of the CSV file for IR voting. An error message will display for the following cases:

- the file does not exist
- the file is not .csv format
- the file is loaded successfully, but the first line of the file does not match "IR"

Following the error message, the user will be asked to re-enter the file's name until it is successfully loaded into the system and the first line matches "IR."



```
Instand Runoff voting system(IR)
Enter the name of CSV file to upload: ir_votes.csv
Please enter to proceed:
```

Interface for uploading CSV file for IR voting

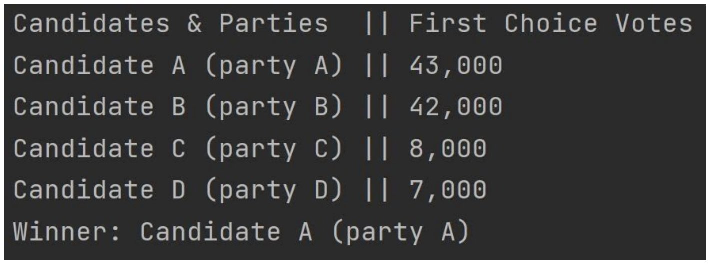
Counting votes

After the user confirms to continue, the system will begin counting the first-choice votes for each candidate. The message "Counting votes in progress" will be displayed to inform the user during this process.

Displaying the result

After the vote-counting process, two cases can result.

Case 1:
One candidate receives over 50% of the first-choice vote. Then, the system will display the result of the election.



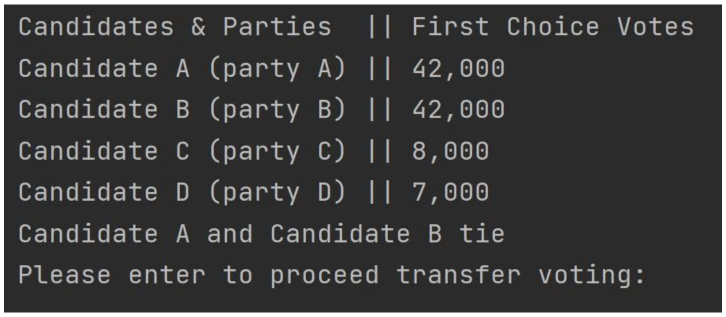
```
Candidates & Parties  || First Choice Votes
Candidate A (party A) || 43,000
Candidate B (party B) || 42,000
Candidate C (party C) || 8,000
Candidate D (party D) || 7,000
Winner: Candidate A (party A)
```

Result of the election for case 1

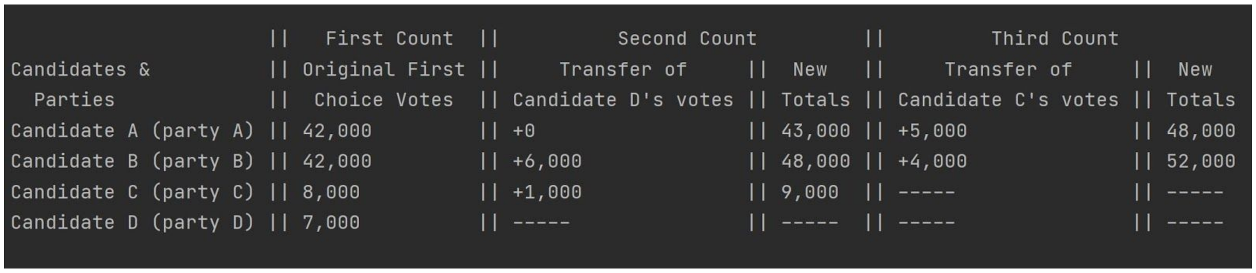
Case 2:
If no candidate received a majority, the candidate with the fewest votes is eliminated, and the system will begin the transfer voting process.

Transfer voting

The voter's first-choice for the candidate with the fewest votes will be collected, and those votes will be transferred to the second-choice candidates. If the result is case 2, the system will repeat the transfer voting process.



```
Candidates & Parties  || First Choice Votes
Candidate A (party A) || 42,000
Candidate B (party B) || 42,000
Candidate C (party C) || 8,000
Candidate D (party D) || 7,000
Candidate A and Candidate B tie
Please enter to proceed transfer voting:
```



```
                      ||   First Count  ||         Second Count         ||         Third Count
Candidates &          || Original First ||     Transfer of     ||  New   ||     Transfer of     ||  New
  Parties             ||  Choice Votes  || Candidate D's votes || Totals || Candidate C's votes || Totals
Candidate A (party A) || 42,000         || +0                  || 43,000 || +5,000              || 48,000
Candidate B (party B) || 42,000         || +6,000              || 48,000 || +4,000              || 52,000
Candidate C (party C) || 8,000          || +1,000              || 9,000  || -----               || -----
Candidate D (party D) || 7,000          || -----               || -----  || -----               || -----
```

Result of the election for case 2

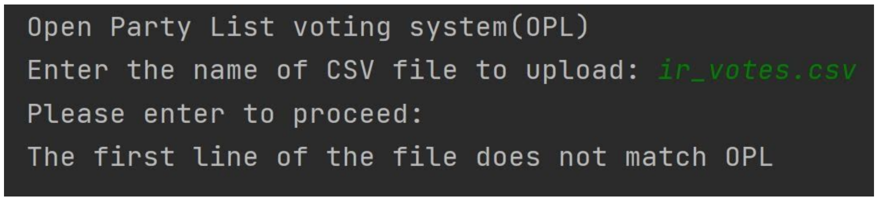
Open Party List (OPL) Voting Interface

Uploading CSV file

The system asks to provide the name of the CSV file for OPL voting. An error message will display for the following cases:

- the file does not exist
- the file is not .csv format
- the file is loaded successfully, but the first line of the file does not match “OPL”

Following the error message, the user will be asked to re-enter the file’s name until it is successfully loaded into the system and the first line matches “OPL”.



```
Open Party List voting system(OPL)
Enter the name of CSV file to upload: ir_votes.csv
Please enter to proceed:
The first line of the file does not match OPL
```

Interface for uploading CSV file for OPL voting

Counting votes

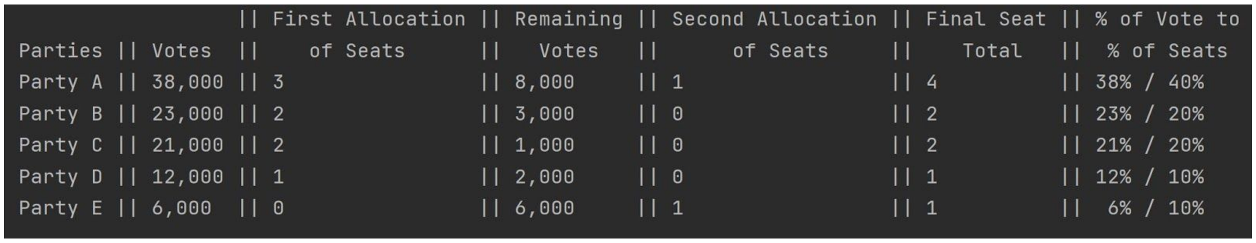
After the file is uploaded successfully and the user confirms to continue, the system will begin the process of counting the votes for candidates from within the same party. The message “Counting votes in progress” will be displayed to inform the user during this process.

Seat allocation

To calculate the seat allocation for OPL voting, the largest remainder formula will be implemented. The system will calculate the quota and the allocation of seats for each party using the number of votes counted and the number of seats. The message “Calculating seat allocation” will be displayed to inform the user.

Displaying the result

At the end of the vote-counting and seat allocation process, the system will display the result of OPL voting along with the information of the total number of seats each party won and the percentage of votes to the percentage of seats for each party.



```
                  || First Allocation || Remaining || Second Allocation || Final Seat || % of Vote to
Parties || Votes  ||    of Seats      ||   Votes   ||      of Seats     ||    Total   ||  % of Seats
Party A || 38,000 || 3                || 8,000     || 1                 || 4          || 38% / 40%
Party B || 23,000 || 2                || 3,000     || 0                 || 2          || 23% / 20%
Party C || 21,000 || 2                || 1,000     || 0                 || 2          || 21% / 20%
Party D || 12,000 || 1                || 2,000     || 0                 || 1          || 12% / 10%
Party E || 6,000  || 0                || 6,000     || 1                 || 1          ||  6% / 10%
```

Result of the election for OPL voting

Saving/Sharing Election Result

After the result is displayed, the system will create and open an audit file to be viewed to verify the results. A message will be displayed to inform the user of this process. Then, the user will be asked whether to share the result with other authorized users.

Fair coin toss

In a special case when there is a tie in either IR or OPL voting, fair coin toss will be run by the program. In this case, the user will be informed that the fair coin toss process has begun and the result will be shown at the end.

3.2 Hardware Interfaces

The minimum hardware system requirements of the voting system are shown in the following:

- RAM: 16GB or more
- Processor: Intel Core i3 550 3.2 GHz or equivalent
- Graphics: Intel 7490 or equivalent

The recommended system requirements are as follows:

- RAM: 32GB or more
- Processor: Intel Core i5 3.2 GHz or equivalent
- 2 x NVIDIA® GeForce GT 755M 1024MB or equivalent

3.3 Software Interfaces

The software includes CSV files that contain the ballot information for the election.

3.4 Communications Interfaces

A stable internet connection is required to send the result of the voting to authorized users.

4. System Features

This section describes the system features through use cases and is organized in a hierarchical fashion. The first to appear is what the program will do. Branches are present for the event of OPL voting type, ID reflects this (i.e. OPL_XX). Use cases that occur in both voting types are also reflected in this way (i.e. VS_XX), main course will default to IR, alternate will reflect any changes for OPL.

4.1 Obtaining the CSV File

Name	Obtaining the CSV File
ID	VS_01
Description	The user obtains the file and places it in the working directory
Actor(s)	Programmer, Tester, Election Official
Organization Benefits	Allows the voting system to process the ballot data in the file
Frequency of Use	The file will be obtained once
Trigger	An election has closed A test of the system
Precondition(s)	CSV file format File is without error A file has not already been obtained File has been preprocessed and has not been changes since
Postcondition(s)	A file is in the programs current directory
Main Course	1. File is obtained by the user (SEE EX1) 2. File is placed into the programs working directory by the user (SEE EX2) 3. Program is run with command line executable followed by the file name
Alternate Courses	User decides to give an absolute path to the directory located outside the program's directory 1. Program is run with command line executable followed by the absolute file path User decides to share results with media 1. Program is run with command line executable followed by the

	absolute file path > text-file-to-hold-results.txt (redirection)
Exceptions	EX1 User selects an empty file 1. Go back to Main 1 EX2 User places file in wrong directory 1. Go back to Main 2

4.2 Reading the File

Name	Reading the File
ID	VS_02
Description	Program reads the file input by the user on the command line
Actor(s)	Programmer, Tester, Election Official
Organization Benefits	Allows the system to process the ballots stored in the file
Frequency of Use	Once for every line in the file until EOF (SEE VS_03, VS_04, VS_05, OPL_01, VS_06, VS_10)
Trigger	VS_01 has been done successfully
Precondition(s)	VS_01 has been done successfully File in CSV format File without error
Postcondition(s)	Vote type obtained Information associated with vote type obtained All ballots tallied and stored
Main Course	1. Program is initiated (SEE VS_01) 2. File obtained from command line (SEE VS_01) 3. File opened (SEE EX1) 4. File read one line at a time 5. Stores data in proper structures
Alternate Courses	None
Exceptions	EX1 File could not be opened 1. Program exits with error code EX2 User Cancels the program 1. Program closes the file 2. Frees memory 3. Exits normally

4.3 Determine the Vote Type

Name	Determine Vote Type
ID	VS_03
Description	Program prompts user for the vote type from the open file and matches it with what is in the file
Actor(s)	Programmer, Tester, Election Official
Organization Benefits	Prevents user error
Frequency of Use	Once
Trigger	File has been successfully opened
Precondition (s)	File has been successfully opened Vote Type has not already been set
Postconditio n(s)	Vote type is set
Main Course	1. Read first line from file 2. Accept user input for voting type 3. Check for match (SEE EX1) 4. Set vote type 5. Engage corresponding vote count algorithm 6. Update audit file 7. Update display results stream
Alternate Courses	None
Exceptions	EX1 User input does not match the file 1. Output to user incorrect file type 2. Return to main 2

4.4 Determine Number of Candidates

Name	Determine Number of Candidates
ID	VS_04
Description	Program prompts user for the number of candidates from the open file

	and matches it with what is in the file
Actor(s)	Programmer, Tester, Election Official
Organization Benefits	Prevents user error
Frequency of Use	Once
Trigger	Vote type has been determined
Precondition(s)	File successfully opened VS_03 successful Number of candidates has not been set
Postcondition(s)	Number of candidates is set
Main Course	1. Read current line from file 2. Prompt user for number of candidates 3. Check for match with file (SEE EX1) 4. Set number of candidates 5. Update audit file 6. Update display results stream
Alternate Courses	None
Exceptions	EX1 User input does not match the file 3. Output to user incorrect file type 4. Return to main 2

4.5 Determine Name of Candidates and Their Party

Name	Determine Name of Candidates and Their Party
ID	VS_05
Description	Program automatically reads all candidates and their party
Actor(s)	Programmer
Organization Benefits	Prevents user error
Frequency of Use	Once
Trigger	Number of candidates has been read
Precondition(s)	File is open

	VS_04 successful Names and parties of candidates have not been set
Postcondition(s)	Names and parties of candidates have been set
Main Course	1. File is read 2. Sets up new candidate in system before each comma in CSV 3. Audit File is updated 4. Display results stream is updated
Alternate Courses	None
Exceptions	None

4.6 Determine Number of Seats

Name	Determine Number of Seats
ID	OPL_01
Description	Program prompts user for the number of seats from the open file and matches it with what is in the file
Actor(s)	Programmer, Tester, Election Official
Organization Benefits	Prevents user error
Frequency of Use	Once
Trigger	If OPL and candidates and their party have been set
Precondition(s)	OPL vote type (SEE VS_03) VS_05 successful Number of seats has not been set
Postcondition(s)	Number of seats has been set
Main Course	1. Current line from file is read 2. Prompt user for number of seats 3. Match user input with file (SEE EX2) 4. Sets number of seats 5. Audit file is updated 6. Display results stream is updated
Alternate Courses	IR is the voting type 1. This does not execute

Exceptions	EX1 IR is the voting type 1. Skip main course EX1 User input does not match the file 1. Output to user incorrect file type 2. Return to main 2

4.7 Determine Number of Ballots

Name	Determine Number of Ballots
ID	VS_06
Description	Program will automatically determine the number of ballots
Actor(s)	Programmer
Organization Benefits	Speeds up process Prevents user error
Frequency of Use	Once
Trigger	OPL - Number of seats has been read IR - Candidates have been read
Precondition(s)	File is open OPL (SEE VS_03) - OPL_01 successful IR (SEE VS_03) - VS_05 successful Number of ballots has not been set
Postcondition(s)	Number of ballots has been set
Main Course	1. Current line from file is read 2. Updates number of ballots 3. Audit file is updated 4. Display results stream is updated
Alternate Courses	None
Exceptions	None

4.8 Calculate the Quota

Name	Calculate the Quota

ID	OPL_02
Description	Calculates the quota in OPL voting algorithm
Actor(s)	Programmer
Organization Benefits	Quota is determined to allocate number of seats to a party in OPL
Frequency of Use	Once
Trigger	Number of ballots has been determined
Precondition(s)	Voting type is OPL (SEE VS_03) OPL_01 successful VS_06 successful Quota has not been set
Postcondition(s)	Quota is set
Main Course	1. Integer divide ballots by number of seats 2. Store as quota
Alternate Courses	IR is the voting type 1. This does not execute
Exceptions	EX1 IR is the voting type 1. Skip main course

4.9 Fair Coin Toss

Name	Fair Coin Toss
ID	VS_07
Description	Used to determine winner or elimination in the event of a tie.
Actor(s)	Programmer
Organization Benefits	Helps in the event of a tie for a fair choice of winner or eliminated candidate
Frequency of Use	Eveytime there is a tie in an election, winner or eliminated candidate
Trigger	A tie has occurred in the ballot counts of an election
Precondition(s)	Two or more candidates are tied A fair coin toss has not already determined winner or eliminated candidate

Postcondition(s)	A winner or eliminated candidate has been chosen
Main Course	1. Give each candidate a unique number, 1 or 2 2. Initiate a coin toss 3. Repeat step 2 1,000 times 4. On 1,000th time 5. If heads, candidate 1 is selected a. else candidate 2 is selected
Alternate Courses	Three way or more tie 1. Give each candidate a unique number between 1 and the number of candidates in the tie 2. Roll die having sides equal to the number of candidates in the tie 3. Repeat step 2 1,000 times 4. If the candidates number is rolled on the 1,000th roll, that candidate is selected
Exceptions	None

4.10 Produce Audit File

Name	Produce Audit File
ID	VS_08
Description	Creates and opens an audit file to be viewed after election count to verify the results
Actor(s)	Programmer, election official, tester
Organization Benefits	Verifies a fair election was run
Frequency of Use	Anytime the election results are questioned
Trigger	Election count has been run Fair coin toss has been defined
Precondition(s)	Vote count system has been run (SEE VS_02) Ballot file is open VS_07 successful Audit file has not yet been created and opened
Postcondition(s)	Audit file is open and ready for voting information
Main Course	1. Create and open audit file in current directory 2. Verify audit file is open (SEE EX1)

	3. Continue with program execution 4. Manually inspect file after voting system completion
Alternate Courses	None
Exceptions	EX1 File was not opened 1. Close ballot file 2. Free memory 3. Exit with error code

4.11 Share Results with Media

Name	Share Results with Media
ID	VS_09
Description	Results will be stored in a defined text file
Actor(s)	Programmer, tester, election official
Organization Benefits	Places election results in a readable text file to be shared with media
Frequency of Use	Once
Trigger	Audit file has been opened
Precondition(s)	Ballot file is open VS_08 successful Media file is not open
Postcondition(s)	Media file is open
Main Course	1. Create and open a media text file in current program directory 2. Verify it has opened (SEE EX1) 3. Continue with program execution 4. Manually inspect file after voting system execution
Alternate Courses	None
Exceptions	EX1 File could not be opened 1. Close audit file 2. Close ballot file 3. Free any allocated memory 4. Exit program with error code

4.12 Process Ballots

Name	Process Ballots
ID	VS_10
Description	Ballots will be read line by line and tallied
Actor(s)	Programmer
Organization Benefits	Automated read of all ballots Prevents user error
Frequency of Use	Once
Trigger	VS_09 successful
Precondition(s)	File is open VS_09 successful Ballots have not been read and counted
Postcondition(s)	Ballots have been read and counted
Main Course	1. Read in ballots line by line 2. Store each ballot's first choice candidate a. update audit file (SEE VS_08) 3. Return to step 1 until EOF 4. If a candidate has over 50% of votes, skip to 10 5. Else, eliminate lowest scoring candidate a. If tie, flip a coin for elimination 6. recount eliminated ballots for a second choice a. If the second choice has been eliminated, use third choice, etc. 7. update the tallies of remaining candidates 8. If over 50% majority for a candidate, skip to 10 9. Else return to step 5 10. update audit 11. update display results stream
Alternate Courses	Voting type is OPL 1. Read in ballots line by line 2. Store each candidates vote a. Keep track of total party votes 3. Return to step one until EOF 4. Allocate number of seats for each party a. The quotient of total party votes / quota 5. Award top scoring candidates allocated seats per party a. If tie, do fair coin toss 6. Allocate remaining seats for each party a. The remainder of total party votes / quota

	7. Award top scoring candidates not already selected per party a. If tie, do fair coin toss 8. Update audit file 9. Update display results stream
Exceptions	None

4.13 Display Results

Name	Display Results
ID	VS_11
Description	Displays results of the election for viewing
Actor(s)	Programmer
Organization Benefits	Automated brief description of election and winner Prevents user error
Frequency of Use	Once
Trigger	Election winner has been selected
Precondition(s)	Election winner has been selected Results have not been displayed
Postcondition(s)	Results have been displayed
Main Course	1. Print display results stream to standard output 2. Print display results stream to media file 3. Exit voting system with success
Alternate Courses	None
Exceptions	None

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The program must be able to process 100,000 ballots within an 8 minute time window.
It has to run both IR and OPL elections multiple times as one piece of software. The software uses no more than one file to count the votes in an election.

5.2 Safety Requirements

No safety or use policy requirements are necessary for this product. There are not any safeguards or required/unallowed actions specified. Voting centers handle safety issues before voting data is used.

5.3 Security Requirements

There are no security requirements present in this software. Security issues are handled in advance by separate voting centers.

5.4 Software Quality Attributes

The voting system software is designed to be easy for election officials to interface with when running an election, given that they have reviewed the instructions and specifications in this document. The software is also designed to be simple and straightforward to test to confirm proper functionality and accuracy.

5.5 Business Rules

Election officials, programmers, and those testing the product will use the voting software directly. The appropriate owner of the data may use the results of an election generated by this voting software.

6. Other Requirements

6.1 Prerequisites

- The information needed for the voting system (such as number of candidates, number of seats, etc.) is provided.
- The comma-separated values (CSV) files are provided and contain no error.
- All files are saved under the same directory as the voting system.

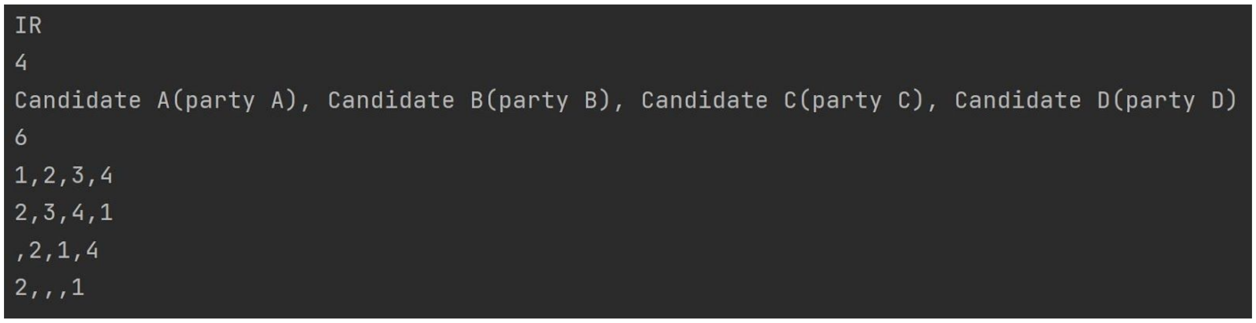
6.2 CSV File Format

The first line of the files contains the type of voting, either IR or OPL. A newline separates each row, and commas separate the column values.

IR voting system

1st Line:	IR
2nd Line:	Number of candidates
3rd Line:	Names of candidates (along with their party names) separated by commas
4th Line:	Number of ballots in the file

From the 5th line onward, each line represents the vote cast by each voter, and the ballots are saved as numbers from 1 to n (the total number of candidates), which represents each candidate's rank. Each ballot must have at least one of the candidates ranked as their top choice.



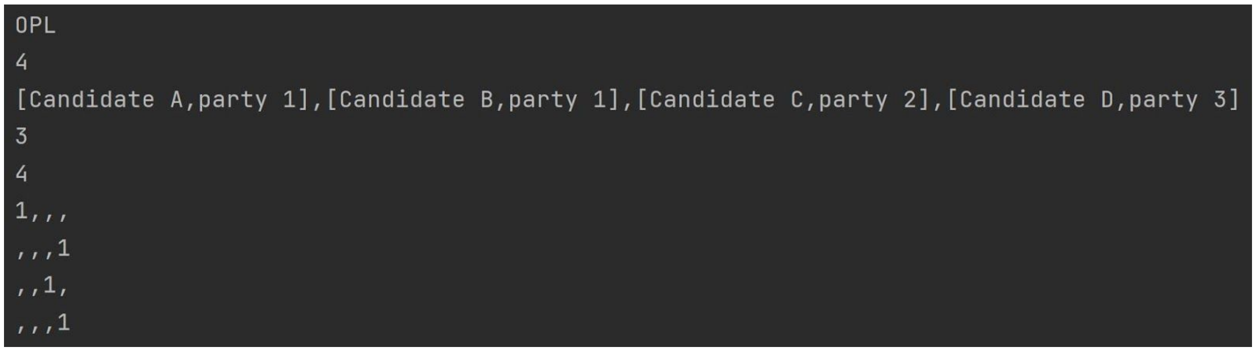
```
IR
4
Candidate A(party A), Candidate B(party B), Candidate C(party C), Candidate D(party D)
6
1,2,3,4
2,3,4,1
,2,1,4
2,,,1
```

Example of CSV file for IR voting

OPL voting system

1st Line:	OPL
2nd Line:	Number of candidates
3rd Line:	Names of candidates (along with their party names) in [ ] separated by commas
4th Line:	Number of seats
5th Line:	Number of ballots in the file

From the 5th line onward, the votes are saved as number 1 for the selected candidate, and only a single one is placed in the position of the selected candidate on each line.



```
OPL
4
[Candidate A,party 1],[Candidate B,party 1],[Candidate C,party 2],[Candidate D,party 3]
3
4
1,,,
,,,1
,,1,
,,,1
```

Example of CSV file for OPL voting

Glossary

- Text-Based User Interface: a system in which the user can interact with the program by using the terminal and keyboard
- CSV File: Comma-separated value file; this file type is the required format for the input data.
- Working Directory: The directory that contains the program executable file