
Software Requirements Specification

for

Voting System

Version 1.15 approved

**Prepared by Shunichi Sawamura (sawam002), Lysong Seang
(seang006), Fumisato Teranishi (teran015), and Crystal Wen
(wen00015)**

University of Minnesota Twin Cities

January 30, 2024

Table of Contents

Table of Contents	i
Revision History	ii
1. Introduction	1
1.1 Purpose	1
1.2 Document Conventions	1
1.3 Intended Audience and Reading Suggestions	1
1.4 Product Scope	1
1.5 References	1
2. Overall Description	2
2.1 Product Perspective	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	2
2.4 Operating Environment	3
2.5 Design and Implementation Constraints	3
2.6 User Documentation	3
2.7 Assumptions and Dependencies	3
3. External Interface Requirements	4
3.1 User Interfaces	4
3.2 Hardware Interfaces	5
3.3 Software Interfaces	5
3.4 Communications Interfaces	5
4. System Features	6
4.1 Election File Upload	6
4.2 Execute Election OPL	7
4.3 Execute Election CPL	11
4.3 Produce an Audit File	15
4.3 Displaying Results and Statistics	17
5. Other Nonfunctional Requirements	18
5.1 Performance Requirements	18
5.2 Safety Requirements	18
5.3 Security Requirements	18
5.4 Software Quality Attributes	19
5.5 Business Rules	19
6. Other Requirements	19
Appendix A: Glossary	19
Appendix B: Analysis Models	19
Appendix C: To Be Determined List	20

Revision History

Name	Date	Reason For Changes	Version
Crystal	2/06/2024	Initialized the SRS document	1.00
Shunichi	2/08/2024	Add Chapter 3 and 5	1.01
Crystal	2/08/2024	Add “File Input” section to Chapter 4	1.02
Crystal	2/09/2024	Add “Produce an Audit File” section to Chapter 4	1.03
Crystal	2/09/2024	Add “Display Results and Statistics” section to Chapter 4	1.04
Fumisato	2/09/2024	Add Chapter 2	1.05
Lysong	2/10/2024	Note the intended of this product is not intended for business for cooperation use	1.06
Crystal	2/10/2024	Add “Execute Election OPL” sections to Chapter 4	1.07
Crystal	2/10/2024	Add “Execute Election CPL” sections to Chapter 4	1.08
Crystal	2/10/2024	Edited use cases 1-9	1.09
Fumisato	2/11/2024	Edited Chapter 2	1.10
Crystal	2/11/2024	Revised grammar and wording	1.11
Lysong	2/11/2024	Rewrite the references	1.12
Shunichi	2/11/2024	Edited Chapter 3, 5-6	1.13
Crystal	2/11/2024	Revised grammar and wording	1.14
Shunichi	2/11/2024	Edited Chapter 2-3, 5	1.15

1. Introduction

1.1 Purpose

The purpose of this document is to lay out the details of the voting system. It focuses on parts of the system such as how people will use it and how the system handles different voting methods like open and closed party list voting. This document is also intended for all audiences such as developers, testers, and users which are the election officials.

1.2 Document Conventions

This Software Requirements Specification follows the IEEE standard. The typographical conventions include using italics for emphasis and bold for definitions. The priority for the requirements is clearly defined for each requirement rather than assuming inheritance from higher-level requirements.

1.3 Intended Audience and Reading Suggestions

The document is intended for a broad audience involved in the development and the users of the voting system.

- Developers - The SRS can help them understand the technical requirements for building the voting system which is described more in detail in section 4 about system features, functional and nonfunctional.
- Election Official - To get a better understanding of how the system works, They should focus on section 3 which is about the user interface.
- Testers - The tester can use the SRS to create test cases. They can look up Section 4 about system features, functional requirements, and nonfunctional requirements.

1.4 Product Scope

This stand-alone voting system software is designed to handle two specific voting processes which are open party list voting and closed party list voting. It is intended to automate vote counting and allocating seats according to the specific voting process. The voting system is not related to any businesses or other corporations. Its goal is to make vote counting and seat allocation easier and faster.

1.5 References

Watters, S. (n.d) Software Requirements Specification Documents for voting system. (February 10, 2024) from

https://canvas.umn.edu/courses/413086/files/41290146?module_item_id=11735324

Varvoutas, K. (February 2017) Software Requirements Specification template example (February 10, 2024) from

https://canvas.umn.edu/courses/413086/files/41290148?module_item_id=11735291

Teamleader, J , Adams, P, Backer, B and Charlie, C (April 15 2004) Software requirements Specification template example for Web Publishing System

https://canvas.umn.edu/courses/413086/files/41290150?module_item_id=11735292

2. Overall Description

2.1 Product Perspective

This self-contained voting system will provide the user with the result of the election conducted by either Open Party List Voting (OPL) or Closed Party List Voting (CPL) and allocate seats based on calculating the number of votes after reading a no-error file. This system will also output an audit file that contains all election information and election results. This system will be developed in Java and will run on CSE lab machines, operated by Windows 10 and Ubuntu 20.04.

2.2 Product Functions

- Information extraction: obtaining what the system needs from the given CSV files
- Choosing OPL or CPL: handling the two types with a single algorithm
- Seat allocation: based on the calculation of the number of votes
- Display: outputting the winners and information related to the election on the user's screen
- Saving result by producing an audit file

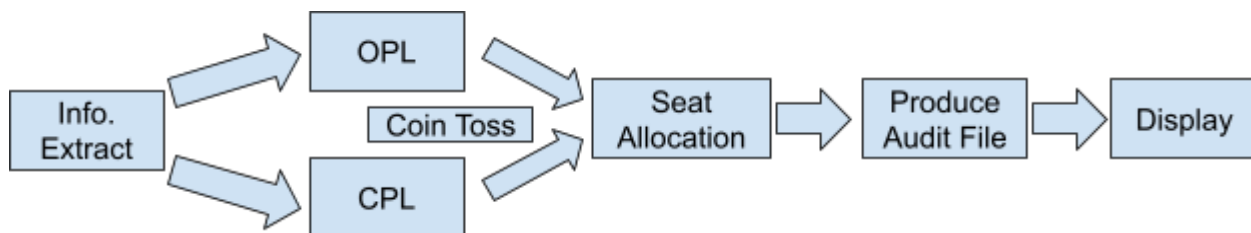


Figure 2.2: Voting System Overview Diagram

2.3 User Classes and Characteristics

- Election officials: This system is used in every election. All they need is to announce the results of the election to the media. This is the only class that is not allowed to change the code, except for the programmers and testers.

- Programmers: They should be able to code and create an algorithm in Java that can implement this system. They must also develop it and handle bugs after release.
- Testers: They are in charge of checking the system and its accuracy developed by the programmers. After each update, they should check what the programmers have changed.

2.4 Operating Environment

Machine recommendation:

- CSE lab machines at the University of Minnesota Twin Cities

Operating System:

- Ubuntu 20.04
- Windows 10

Memory:

- 8GB RAM
- 16GB RAM

Processor:

- Intel Core i7-6700
- Intel Xeon e5-2460
- Threadripper Pro 5945WX

Language:

- Java for development

2.5 Design and Implementation Constraints

This voting system is written in Java and runs on a CSE lab machine at the University of Minnesota Twin Cities. The prompt can be the text or GUI. The software runs 100,000 ballots in less than 4 minutes and has no security or safety requirements.

2.6 User Documentation

SSH: <https://cse.umn.edu/cseit/self-help-guides/secure-shell-ssh>

Command Line: <https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html>

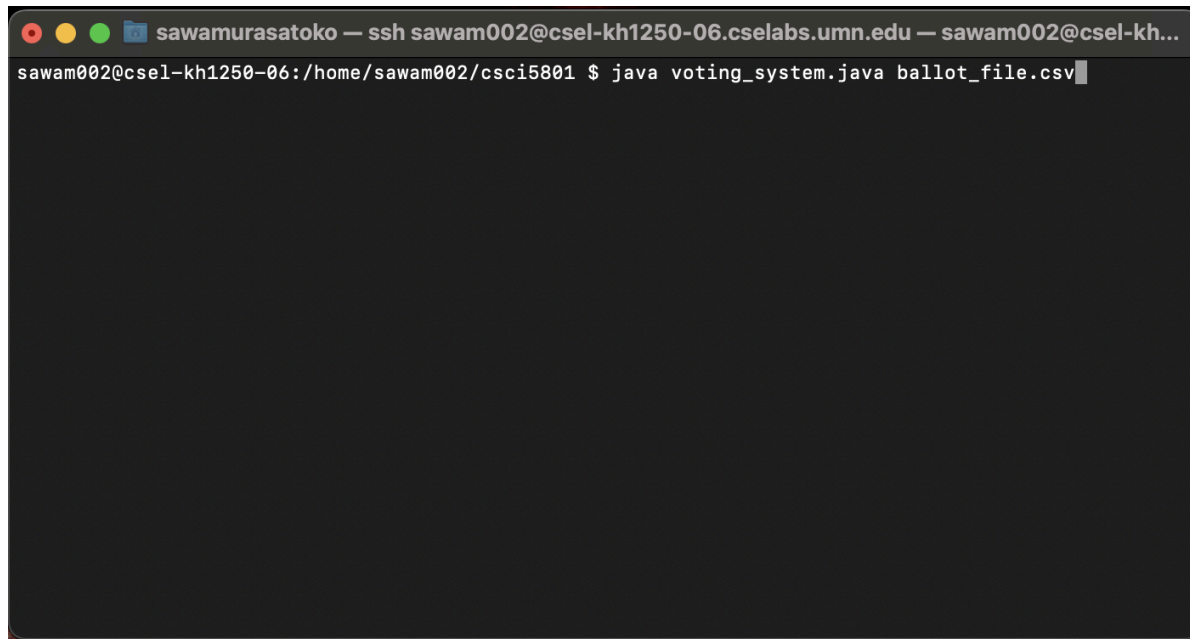
2.7 Assumptions and Dependencies

This should run on a CSE lab machine. Java is used for this system development, so we assume this has been installed on the user's environment before use. This system will process only one file at each time and requires execution by users who input the file correctly; moreover, an input file should be placed in the same directory as the program. It is assumed that the assigned CSV file has no numbering mistakes in the ballot. We anticipate the users should be able to run this system with the command line and text or GUI.

3. External Interface Requirements

3.1 User Interfaces

There are two ways to input a CSV file into this system. A user can provide a filename through a command line argument as shown in Figure 3.1. Alternatively, the user can input a filename with the text prompt as shown in Figure 3.2. After the system successfully receives the ballot file, it implements the voting algorithm to figure out the winners and stats for the given ballot. At the end of the process, the system produces an audit file that saves information regarding the voting result and displays the result on the screen.

A screenshot of a terminal window. The title bar at the top reads "sawamurasatoko — ssh sawam002@cse1-kh1250-06.cselabs.umn.edu — sawam002@cse1-kh...". The terminal content shows the prompt "sawam002@cse1-kh1250-06:/home/sawam002/csci5801 \$" followed by the command "java voting_system.java ballot_file.csv" with a cursor at the end of the line.

```
sawam002@cse1-kh1250-06:/home/sawam002/csci5801 $ java voting_system.java ballot_file.csv
```

Figure 3.1: Input a CSV File With Command Line Argument

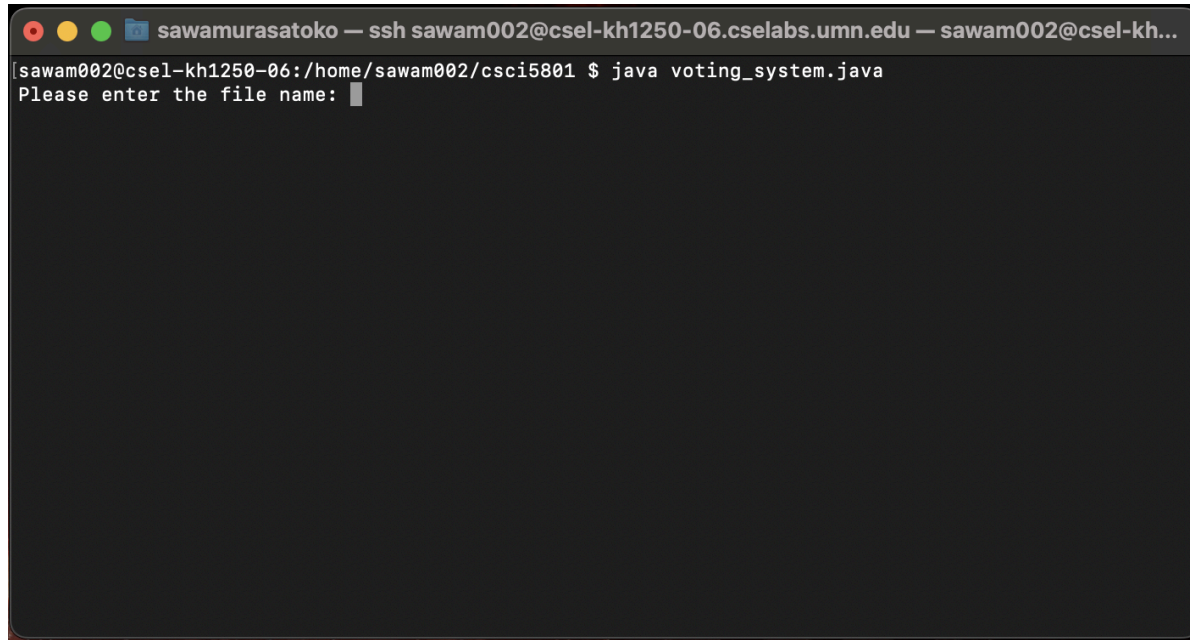


Figure 3.2: Input a CSV File with Text Prompt

3.2 Hardware Interfaces

The voting system is developed to run on the University of Minnesota CSE lab machines. The servers of CSE lab machines include Dell PowerEdge R740, Dell PowerEdge R815, and Dell PowerEdge R330.

3.3 Software Interfaces

- The voting system would be developed in Java 19 and implemented on Linux Terminal (Ubuntu 22.04.3).
- It receives CSV files to get information about the ballots
- The input CSV file should be placed on the same directory as this program
- The system creates .txt files to produce the voting result information including winners and stats.
- Java built-in packages
 - Math: implement coin toss function and break a tie
 - FileReader: read an input CSV file
 - IOException: use for exception handling when 1) reading an input CSV file that was no longer available, 2) trying to read/write a file, but don't have permission, 3) Trying to write to a file, but disk space was no longer available.

3.4 Communications Interfaces

There are no required communication interfaces in this voting system. The system would receive the CSV file through the local file system.

4. System Features

4.1 Election File Upload

4.1.1 Description and Priority

This feature will prompt the user for the filename. How the user inputs the file name depends on the type of user. If the user is a tester, they will input the file name in the command line. Otherwise, a user will input the file name in a GUI application. The file will always be in the same directory as the program and follow the same format. The priority of this system is high, followed by the rating of the specific components of the system. The rating ranges from a relative scale from a low of 1 to a high of 9.

Benefit	9
Penalty	9
Cost	3
Risk	9

4.1.2 Stimulus/Response Sequences

1. The user runs the program.
2. The system will prompt the user to input a filename.
3. The system will open the file for reading and extracting information.

4.1.3 Functional Requirements

Name	User Input Filename
ID	UC_001
Descriptions	The user inputs a file name and the system will open the file
Actors	Election officials, testers, programmers
Organizational Benefit	Allows the system to access the information from the file.
Frequency of Use	Once every time the program runs.
Triggers	System runs and the system prompts the user to input a filename.
Preconditions	The file exists, is readable, and can be opened successfully.
Postconditions	The file is opened and read successfully.
Main Course	<ol style="list-style-type: none"> 1. The system prompts the user to input a filename. 2. The user inputs the file name in a GUI (See AC1).

	3. The system opens the file. (See EX1)
Alternate Course	AC1 If the user is a tester. 1. The user inputs the file name in the command line. 2. Return to Main Course step 2.
Exceptions	Ex1 The file doesn't exist or the name is incorrect 1. The system displays an error message that the file doesn't exist and checks if the file name is accurate. 2. Return to Main Course step 1. Ex2 The file is unreadable. 1. The system displays an error message that the file is unreadable and to try again. 2. Return to Main Course step 1. Ex3 The file is not a .txt file. 3. The system displays an error message that the file is not a .txt file. 4. Return to Main Course step 1.

4.2 Execute Election OPL

4.2.1 Description and Priority

After reading the text file, the system will determine the results by election type and allocate the seats according to the largest remainder approach. If the file is an OPL type of election, where a vote for a candidate is a vote for their affiliated party, the system will count the votes for candidates and the votes for the parties. The system will then allocate seats and determine the winners.

Benefit	9
Penalty	9
Cost	9
Risk	9

4.2.2 Stimulus/Response Sequences

1. The user runs the program and inputs the text file.
2. The system opens the file for reading.
3. The system extracts information about election type, number of ballots, number of seats, number of candidates, the candidates and their affiliated parties, and the ballots.
4. The system will count the number of votes each party and each candidate received.

5. The system will then calculate the quota which is the number of ballots divided by the number of seats.
6. The system will allocate seats by dividing each party's votes by the quota to determine the number of seats given to each party in the first round.
7. If there are any remaining seats, these seats will be given to the parties with the largest remainders in a round-robin fashion.
8. The system will then determine the winners.

4.2.3 Functional Requirements

Name	Information Extraction
ID	UC_002
Descriptions	Extracting all information needed to run the voting algorithms
Actors	Voting system
Organizational Benefit	Provides information on election type, number of ballots, number of parties, number of seats available, parties, candidates, and the ballots
Frequency of Use	Whenever the program runs
Triggers	When the file is opened
Precondition	The file exists and is readable.
PostCondition	All information has been extracted
Main Course	<ol style="list-style-type: none"> 1. The system opens the file to read (See “File Input” use case). 2. The system extracts election type. 3. The system extracts the number of seats. 4. The system extracts the number of ballots. 5. The system extracts the number of parties. 6. The system extracts party information 7. The system extracts the ballots.
Alternate Course	None
Exceptions	None

Name	Vote counting OPL
ID	UC_003
Descriptions	Count the votes for both candidates and their affiliated party

Actors	Voting system
Organizational Benefit	Helps the system organize the ballots and is used in seat allocation
Frequency of Use	Every time the input file is OPL
Triggers	When OPL file is ready to parsed
Precondition	Election type is OPL
Postcondition	Report the count for the candidate in the party
Main Course	<ol style="list-style-type: none"> 1. The system extracts relevant election information (See “Information Extraction” use case) 2. The system counts the votes that each candidate received. 3. Each vote for a candidate is also counted as a vote for their party. The system counts the votes for each party.
Alternate Course	None
Exceptions	None

Name	Seat allocation
ID	UC_005
Descriptions	Allocating seats to political parties and their candidates based on the votes received using the Largest Remainder method
Actors	Voting system
Organizational Benefit	Allocate the seat according to the number of vote
Frequency of Use	Every time the program runs
Triggers	When all votes have been counted for each party or candidate
Precondition	All votes have been counted
PostCondition	Parties and candidates are awarded seats based on the votes received.

Main Course	<ol style="list-style-type: none"> 1. The system confirms the total number of seats available and the number of ballots. 2. The system calculates the quota 3. The system finds the floor of the quotient of each party's votes and the quota. 4. The system allocates seats based on the result. 5. The system checks if there are remaining seats. 6. The system allocates a seat to the parties with the largest remainders. (AC1)
Alternate Course	AC1 There is a tie between two or more parties and only 1 seat is left. <ol style="list-style-type: none"> 1. See the "Coin Toss" use case. 2. Return to Main Course step 6
Exceptions	None

Name	Coin Toss
ID	UC_006
Descriptions	This function randomly chooses a winner between 2 or more options based on a coin toss.
Actors	Voting system
Organizational Benefit	Fairly picks between 2 or more people who would win during a tie
Frequency of Use	Whenever there is a tie
Triggers	There is a tie
Precondition	There is a tie between 2 or more winners
PostCondition	One winner is chosen
Main Course	<ol style="list-style-type: none"> 1. A list of parties that are tied is sent to the function. 2. The system randomly chooses who wins based on a fair coin toss. 3. The function returns the winner.
Alternate Course	None
Exceptions	None

Name	Determine Winners
ID	UC_007
Descriptions	After allocating seats, the system determines who won the election
Actors	Voting system
Organizational Benefit	The system gets the final election result
Frequency of Use	Once the process of allocating seats has finished
Triggers	Seat allocation has been completed
Precondition	All seats have been allocated to the parties
PostCondition	Winners are determined
Main Course	<ol style="list-style-type: none"> 1. The system finds which party and candidate received the most seats and votes (AC1). 2. The system determines the winner.
Alternate Course	AC1 There is a tie <ol style="list-style-type: none"> 1. Call the coin toss function to fairly choose who won (see “Coin Toss” use case). 2. Return to Main Course step 2.
Exceptions	None

4.3 Execute Election CPL

4.3.1 Description and Priority

After reading the text file, the system will determine the results by election type and allocate the seats according to the largest remainder approach. If the file is an OPL type of election, where a vote for a candidate is a vote for their affiliated party, the system will count the votes for candidates and the votes for the parties. The system will then allocate seats and determine the winners.

Benefit	9
Penalty	9
Cost	9

Risk	9
------	---

4.3.2 Stimulus/Response Sequences

1. The user runs the program and inputs the text file.
2. The system opens the file for reading.
3. The system extracts information about election type, number of ballots, number of seats, number of parties, the parties and their candidates, and the ballots.
4. The system will count the number of votes each party received.
5. The system will then calculate the quota which is the number of ballots divided by the number of seats.
6. The system will allocate seats by dividing each party's votes by the quota to determine the number of seats given to each party in the first round.
7. If there are any remaining seats, these seats will be given to the parties with the largest remainders in a round-robin fashion.
8. The system will then determine the winners.

4.3.3 Functional Requirements

Name	Information Extraction
ID	UC_002
Descriptions	Extracting all information needed to run the voting algorithms
Actors	Voting system
Organizational Benefit	Provides information on election type, number of ballots, number of parties, number of seats available, parties, candidates, and the ballots
Frequency of Use	Whenever the program runs
Triggers	When the file is opened
Precondition	The file exists and is readable.
PostCondition	All information has been extracted
Main Course	<ol style="list-style-type: none"> 1. The system opens the file to read (See “File Input” use case). 2. The system extracts election type. 3. The system extracts the number of seats. 4. The system extracts the number of ballots. 5. The system extracts the number of parties. 6. The system extracts party information 7. The system extracts the ballots.
Alternate Course	None

Exceptions	None
------------	------

Name	Vote counting CPL
ID	UC_004
Descriptions	Describes the process by which the voting system counts the votes each political party has received
Actors	Voting system
Organizational Benefit	Accurate count the votes to ensure the integrity of election process
Frequency of Use	Whenever election results are to be processed from CPL file
Triggers	When a CPL file is ready to be processed after the poll closure
Precondition	Must be a CPL file
PostCondition	All parties have the correct number of votes given to that party
Main Course	<ol style="list-style-type: none"> 1. The system extracts relevant election information (See “Information Extraction” use case) 2. The system adds a vote to each party according to the placement of the vote. 3. The system updates the total number of votes for each party and saves the result
Alternate Course	None
Exceptions	None

Name	Seat allocation
ID	UC_005
Descriptions	Allocating seats to political parties and their candidates based on the votes received using the Largest Remainder method
Actors	Programmers, voting system
Organizational Benefit	Allocate the seat according to the number of vote
Frequency of Use	Every time the program runs
Triggers	When all votes have been counted for each party or candidate

Precondition	All votes have been counted
PostCondition	Parties and candidates are awarded seats based on the votes received.
Main Course	<ol style="list-style-type: none"> 1. The system confirms the total number of seats available and the number of ballots. 2. The system calculates the quota 3. The system finds the floor of the quotient of each party's votes and the quota. 4. The system allocates seats based on the result. 5. The system checks if there are remaining seats. 6. The system allocates a seat to the parties with the largest remainders. (AC1)
Alternate Course	<p>AC1 There is a tie between two or more parties and only 1 seat is left.</p> <ol style="list-style-type: none"> 1. See the "Coin Toss" use case. 2. Return to Main Course step 6
Exceptions	None

Name	Coin Toss
ID	UC_006
Descriptions	This function randomly chooses a winner between 2 or more options based on a coin toss.
Actors	Voting system
Organizational Benefit	Fairly picks between 2 or more people who would win during a tie
Frequency of Use	Whenever there is a tie
Triggers	There is a tie
Precondition	There is a tie between 2 or more winners
PostCondition	One winner is chosen
Main Course	<ol style="list-style-type: none"> 1. A list of parties that are tied is sent to the function. 2. The system randomly chooses who wins based on a fair coin toss. 3. The function returns the winner.

Alternate Course	None
Exceptions	None

Name	Determine Winners
ID	UC_007
Descriptions	After allocating seats, the system determines who won the election
Actors	Voting system
Organizational Benefit	The system gets the final election result
Frequency of Use	Once the process of allocating seats has finished
Triggers	Seat allocation has been completed
Precondition	All seats have been allocated to the parties
PostCondition	Winners are determined
Main Course	<ol style="list-style-type: none"> 1. The system finds which party and candidate received the most seats and votes (AC1). 2. The system determines the winner.
Alternate Course	AC1 There is a tie <ol style="list-style-type: none"> 1. Call the coin toss function to fairly choose who won (see “Coin Toss” use case). 2. Return to Main Course step 2.
Exceptions	None

4.4 Produce an Audit File

4.4.1 Description and Priority

After the system determines election results, the system will produce a text file that contains all of the election information. This includes election type, number of parties, number of ballots, number of seats, candidate's name and party affiliation, the quota for the largest remainder approach, a table of how the seats were allocated, and a list of seat winners and their party affiliation. If the election was an OPL type, the audit file will also provide the number of votes received by each candidate. This is an important feature in our system, and as such it has a high priority.

Benefit	9
Penalty	9
Cost	3
Risk	2

4.4.2 Stimulus/Response Sequences

1. The System finishes running the election and allocating seats.
2. The system creates an audit file with election information and results and saves it in the same directory.

4.4.3 Functional Requirements

Name	Produce Audit file
ID	UC_008
Descriptions	The audit file is generated to save the result of the ballot based on the input file. The audit file consists of the type of election (Open Party Listing or Closed Party Listing), number of parties, number of ballots, number of seats, the candidates' names and party affiliation, the calculation for the largest remainder approach to seat allocation, and a list of the seat winners and their party affiliation.
Actors	Voting system
Organizational Benefit	It helps record the official election results and share the results with media personnel.
Frequency of Use	Every time the system finishes running the CPL and OPL election and seat allocation.
Triggers	System finishes executing the election and allocating seats.
Precondition	The election results are calculated with the program algorithm, and all required information listed in the Description section is available in the program.
PostCondition	The audit file describing the election result is generated.
Main Course	<ol style="list-style-type: none"> 1. The system creates a new file. 2. The system writes the type of election in the file. 3. The system writes the number of parties in the file. 4. The system writes the number of ballots in the file. 5. The system writes the number of seats in the file.

	6. The system writes the candidates' names and party affiliations in the file. 7. The system writes the calculation for the largest remainder approach to seat allocation. 8. The system writes a list of the Seat Winners and their Party Affiliation in the file. (AC1) 9. The system saves all changes in the file.
Alternate Course	AC1 If it is an open party listing type of election, 1. The system writes a list of the seat winners, their party affiliation, and the number of votes received by each candidate in the file. 2. Return to Main Course step 9.
Exceptions	None

4.5 Displaying Results and Statistics

4.5.1 Description and Priority

When all the votes have been allocated and the seats have been allocated, the system will display the winner, statistics, and election information on the screen. This feature has a medium priority because this directly shows the voting results.

Benefit	5
Penalty	2
Cost	2
Risk	1

4.5.2 Stimulus/Response Sequences

1. After the winners are determined and the seats are allocated, the system will print the results, statistics, and election information to the screen.

4.5.3 Functional Requirements

Name	Display results
ID	UC_009
Descriptions	Displays information related to the election as listed below. - Winners - The number of ballots cast

	- Winners and the stats for all candidates, including number of votes received, the percentage of votes received, the winner(s).
Actors	Voting system
Organizational Benefit	Shows the results of the election and related statistics for users.
Frequency of Use	Whenever the program runs
Triggers	All information has been extracted and all calculations have been completed
Precondition	When required information has been provided Voting algorithm is completed. The election result is clarified.
PostCondition	Displays the results after the voting algorithm
Main Course	<ol style="list-style-type: none"> 1. Extract all information (See “Information Extraction” use case) 2. Count the votes (See “Vote counting OPL” or “Vote counting CPL) 3. Allocate seats (See “Seat Allocation” use case) 4. Calculate statistics for each party. 5. Display the results and related information on the screen.
Alternate Course	None
Exceptions	None

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The voting system must be able to handle 100,000 ballots in under 4 minutes.

5.2 Safety Requirements

There are no special safety requirements.

5.3 Security Requirements

There are no special security requirements.

5.4 Software Quality Attributes

- **Adaptability:** The voting system should handle two types of voting, OPL and CPL voting.
- **Availability:** The voting system is available unless the CSE lab machine is under maintenance.
- **Flexibility:** The system is assumed to only run with Java 19.
- **Interoperability:** The audit file is produced as a .txt file, and the information of the result is arranged in a predetermined order and separated by lines. For example, the first line indicates the type of voting (OPL or CPL).
- **Maintainability:** The system can be updated through the GitHub repository.
- **Portability:** The system is designed only to run on CSE lab machines.
- **Reusability:** This single voting system can handle both OPL and CPL voting. It reuses file input, displays results, and the coin toss function.
- **Testability:** The system will be tested with unit tests, and the code will be uploaded to the GitHub repository.
- **Usability:** The system can be run through a Linux terminal. Users can input a file using 1) command line argument and 2) text or GUI.

5.5 Business Rules

If there is a tie in ballots, the system should implement a coin toss function to make a fair system.

6. Other Requirements

There are no other requirements.

Appendix A: Glossary

- **Closed Party List Voting (CPL):** A voting system where parties put forward a list of candidates that is set by the party and the voters can only vote for the party itself.
- **Open Party List Voting (OPL):** A voting system where voters influence the candidates that they elected and it also influences the party.
- **Seat Allocation:** A formula to calculate seats for each voting system.
- **Audit File:** A file that will be produced after the user input the file which reports the result of the vote based on the provided input file.
- **CSV:** A file format that uses commas to separate values that are commonly used for representing table data.
- **GUI:** Graphical user interface which means a digital interface for a user to interact with graphical components such as icons, buttons, and menus.
- **CSE Lab Machines:** Remote access servers managed by the University of Minnesota.

Appendix B: Analysis Models

- Figure 2.2: Voting System Overview Diagram
- Figure 3.1: Input a CSV File With Command Line Argument
- Figure 3.2: Input a CSV File with Text Prompt

Appendix C: To Be Determined List

There is currently no to-be-determined list.