Software Requirements Specification

for

Voting System

Version 6.00 approved

Composed by

Sunny Qin (qing0002), Yingjin Zhang (zhan4973)

Rex Zhu (zhu00100), Xiaohui Chao (chao0070)

University of Minnesota

10/10/2019

Table of Contents

1. Revision History

Name	Date	Reason For Changes	Version
Rex	09/26/2019	Initialize the SRS document	1.00
Sunny, Yingjin	09/27/2019	Write two use cases	1.01
Rex	09/28/2019	write three use cases	1.02
Sunny	09/30/2019	Write chapter 5 of SRS document	1.03
Rex	09/30/2019	Write chapter 1 and 2.4, 2.5, 2.6 of SRS document	1.04
Yingjin	10/2/2019	Modify the use case 1,2	2.00
Sunny	10/2/2019	Write section 3.1 of SRS document	2.01
Rex	10/3/2019	Modify chap1 and section 3.1	3.00
Xiaohui	10/4/2019	Write two use cases	3.01
Xiaohui	10/9/2019	Write Chap 4, 6, section 2.1, section 2.2, revise all use cases	3.02
Xiaohui	10/9/2019	Adjust font size and font type of the whole document	4.00
Yingjin	10/9/2019	Review chap 4, all use cases and modify them	4.01
Sunny	10/9/2019	Review Chap 1, 2, 3 and modify chap 2.3-2.7	4.02
Rex	10/10/2019	Draw a use case diagram and an external interface diagram	5.00
Xiaohui	10/10/2019	Review and check grammar, add a user interface screenshot	6.00
Sunny Qin	10/10/2019	Review the doc, push to github	6.01
Rex Zhu	10/10/2019	Final review the doc, fix format, push to github again	6.02

1. Introduction

1.1 Purpose

The Software Requirement Specification comprises of concrete information about Voting System which includes its intended use, architecture design, deployment method, and other relevant resources. This document faces users and developers interacted with Voting System.

1.2 Document Conventions

This document was created based on the IEEE template for System Requirement Specification Documents.

1.3 Intended Audience and Reading Suggestions

The audiences for this article are programmers, testers, election officials and stakeholders.

Reading the document, readers can follow the sequence of Chapter 1->Chapter 2->Chapter 4->Chapter 5-> Chapter 3->Chapter 6 and Appendix.

1.4 Product Scope

Countries and regions require a more and more transparent, efficient, and scientific method to hold elections. Under this circumstance, the Voting System is designed and put into use. The Voting System has encapsulated algorithms for two types of election: OPL (Open Party List) and CPL (Closed Party List).

1.5 References

Textbook:

Software Engineering, Ian Sommerville, Tenth Edition, ISBN 10: 0-13-394303-8, ISBN 13: 978-0-13-394303-0

System Requirement Specification:

https://github.umn.edu/umn-csci-5801-f19/repo-Team12/blob/master/Homework1/docs/CSci5801HW1_Fall2019_v1.pdf

Software Requirements Specification for Gephi:

https://canvas.umn.edu/courses/134519/pages/software-requirements-specification-supporting-documents?module_item_id=2883887

Software Requirements Specification for Web App:

https://canvas.umn.edu/courses/134519/pages/software-requirements-specification-supporting-documents?module_item_id=2883887

Priority Levels:
http://www.jumpmind.com/services/support/priority-levels

2. Overall Description

2.1 Product Perspective

The Voting System is a useful, self-contained product. It is designed to count the voting ballots and decide the winners and winning parties. This system is supposed to be used after the actual voting is finished and a voting file is provided to the election officials. It allows election officials to transfer the original voting file into a final election result with the election information and output the result. Also, this system helps the election officials send results to media and let the public know the election results. All the functions mentioned above can be accomplished by this system itself.

The Voting System is mainly constructed by the following components:

1. a file upload feature
2. election execution, which includes ballot counting, seat allocation, and winner generation features
3. audit file generation feature
4. result display feature
5. result saving feature

2.2 Product Functions

Major functions' summary:

- File upload: upload the original voting file into the system
- Ballot counting: count the ballot. For different list systems, the counting algorithm will be different
- Seat allocation: allocate the seats for each party
- Winner generation: generate the winners' list
- Audit file generation: generate an audit file for supervision
- Result display: display the election result to screen
- Result saving/sending: save the result for media personnel

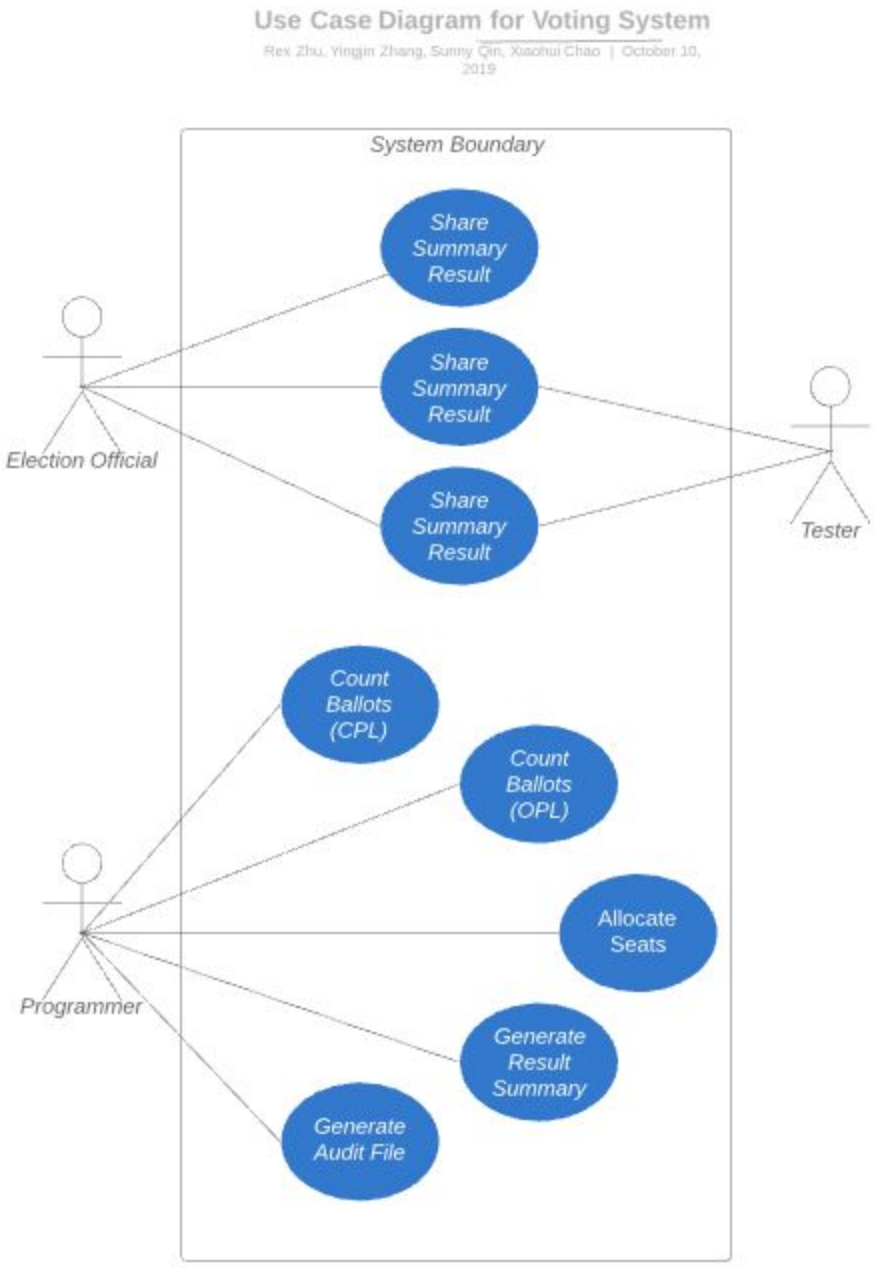

Figure 1.1 The use case diagram of the Voting System.

2.3 User Classes and Characteristics

The system users can be divided into three classes. Each one has its functions.

a) Programmer level
Programmers have access to run the voting system and generate the voting results based on the input file. They will also generate an audit file and save it in the same directory to help others check on the results.

b) Officials level
Officials can upload their voting file to generate the winners. They can also share the result summary report to the media and check the audit file to see whether the algorithm works well.

c) Testers level
Testers can generate test cases based on use cases and test on the system to check whether it works well. They have access to the audit file to help them check the selection process.

2.4 Operating Environment

- Platforms: Ubuntu 16.04/ Windows 10
- Workstations: Dell OptiPlex 9020 - Intel Core i7 @ 4 GHz
- Monitors: 27" LCD
- Peripherals: HP LaserJet 600 M602 B&W Printer, HP LaserJet M651 Color Printer

2.5 Design and Implementation Constraints

The Voting System is developed in Java.

Constraints for Design:

The Voting System is deployed in the CSE lab, therefore the computation ability of this system is limited.

Constraints for Implementation:

We have 2 weeks to implement this system and get familiar with the middleware and framework of this system.

2.6 User Documentation

We will provide a detailed user documentation after the system is released. The documentation may include the instruction for different levels of users.

2.7 Assumptions and Dependencies

The voting system is developed in Java, so it requires users to have Java installed on their system. The latest version of the Voting System requires Java version 1.8.0 or higher. This applies to the Linux system in CSE labs.

3. External Interface Requirements

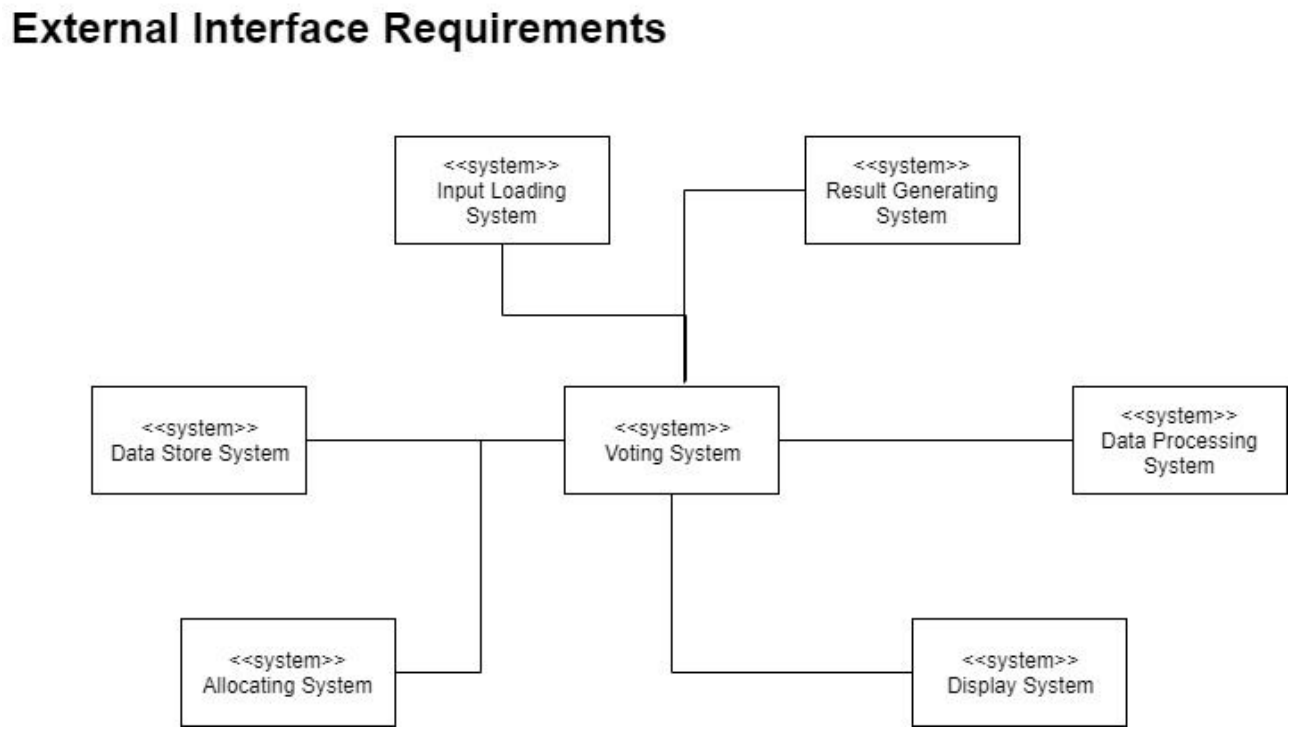

Figure 3.1 External interface requirement of the Voting System.

3.1 User Interface

To start, the user may need to add their CSV file path in the command line to help run the program.

After uploading the CSV file, the system starts to read the first line of the file and based on the information they included in the first line, the file will be the input for corresponding election methods. The system should print out the information they read from the first line to let the user know the progress.

When the system starts to count the ballots, there should be a line printed out to inform the user the progress of the counting and allocating process. After the system is finished, the audit file is already generated in the same directory, so there should be information about that file to indicate where it is and what it includes. The summary report of the election will be printed after the program finishes, and there will also be a text file describing the result saved in the same directory. The user can send it to media later.

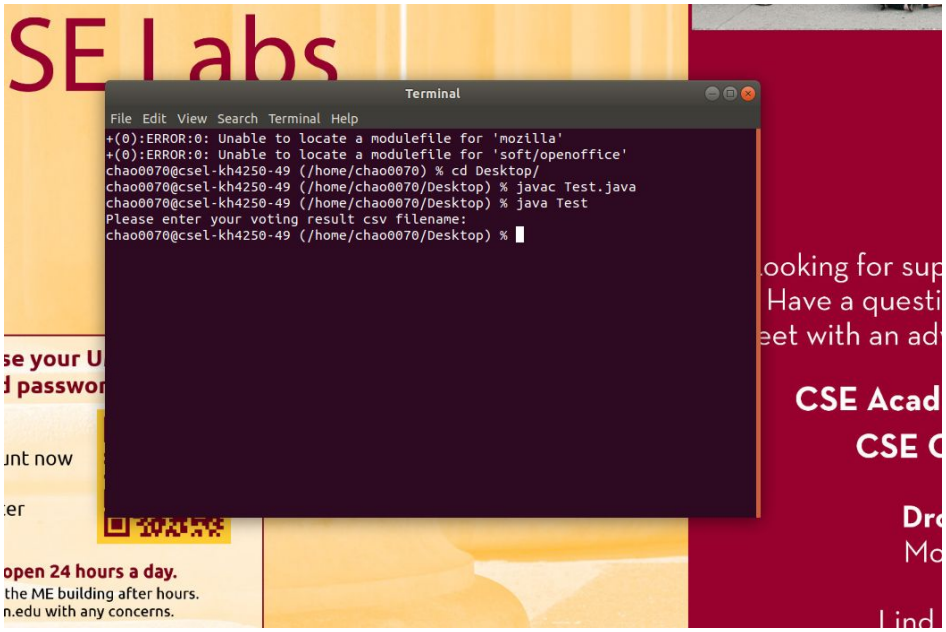

Figure 3.2 An example of the Voting System user interface .

3.2 Hardware Interfaces

The Voting System relies on the University of Minnesota CSE lab machines.

The CSE lab machines include Dell OptiPlex 9020, Dell Precision T7910, Dell Precision T3620 and Dell Precision T3420. All the working stations are equipped with Intel i7 CPUs, wide screens and printer.

3.3 Software Interfaces

Voting System requires java to be installed. The latest version of the Voting System requires Java version 1.8.0 or higher. More information can be found in section 2.7.

3.4 Communications Interfaces

The voting system is an offline system. It only deals with the file saved in the local computer, so it does not need any communication interfaces in our system.

4. System Features

4.1 Uploading a CSV File

4.1.1 Description and Priority

The actual voting will be done separately from this voting system. Ballots will be cast online and a comma delimited text file will be provided. This feature will prompt the user for the filename. The file will always be in the same directory as the program. The file will be

exported from Excel into the CSV format. The first line of the file will provide the type of voting (i.e. CPL, OPL). Since all information that is needed to run the program will come through this file, this feature has a high priority.
Priority component ratings:
benefit(9), penalty(9), cost(3), risk(9)

4.1.2 Stimulus/Response Sequences
System: prompts the user to upload a CSV file.
User: types the file name to load the file

4.1.3 Functional Requirements

Name	Upload a voting result file
ID	UC_001
Description	The user uploads an input CSV file in the correct format
Actors	Programmer, tester, election officials
Organizational Benefits	The organization can view the election result after uploading the result file
Frequency of Use	Once for each using of this system
Triggers	System prompts users to upload a CSV file from local computer dictionary
Preconditions	The input file is readable and can be opened successfully The input file has the correct format
Postconditions	The file is opened and read successfully
Main Course	1. System prompts user to upload a partial voting result file as CSV format (see AC1) 2. User uploads a voting result file (see EX1) 3. The file is uploaded into the system 4. System goes to User Case 2
Alternative Course	AC1 User decides not to upload the file and click “cancel” 1. Return to Main Course step UC_002
Exceptions	EX1. Uploading is interrupted 1. Kill the process and print an error message, then request re-uploading.

4.2 Execute the Election

4.2.1 Description and Priority

After reading in the CSV file, the system will calculate the results according to the voting type. Based on the voting results, the system will allocate the seats to each party and generate winners. For different voting types (i.e. CPL, OPL), the system will generate different results. For CPL, the system will return the party list, the ballot counting list, and lists of candidates for parties. For OPL, the system will return one dictionary with party names as the key, the sorted candidate order as the first value and votes for the party as the second value. Then, the system will allocate seats to parties and generate winners.
This feature has a high priority since this is the most critical part of the system.
benefit(9), penalty(9), cost(9), risk(9)

4.2.2 Stimulus/Response Sequences

Programmer: Chooses to count ballots
System: Read the ballots, count the ballots for each party, allocate seats to parties, and generate winners

4.2.3 Functional Requirements

Name	Count ballots (CPL)
ID	UC_002
Description	Read the ballots and count the ballots for each party
Actors	Programmer
Organizational Benefits	Help the system organize the ballots and use the results in the next step
Frequency of Use	Every time when running a closed party list voting
Triggers	The programmer chooses to count the ballots
Preconditions	The voting result file is loaded to the system successfully.
Postconditions	The ballots of each party have been counted
Main Course	1. System reads the first line of the file and determines the voting type 2. System determines the voting type is closed party list (see AC1) 3. System reads the number of the parties (see EX1) 4. System reads the parties in order of ballot ordering 5. System reads the number of seats 6. System reads the number of ballots

	7. System reads the number of candidates 8. System reads the candidates in each party in order 9. System reads the next vote and adds 1 to the count of the corresponding party 10. System writes parties and their ballots counting into an audit file 11. System repeats step 9 and step 10 till reaches the end of the file 12. Return the parties and their corresponding ballot countings lists of candidates for parties
Alternative Course	AC1 The voting type is open party list 1. System goes to Use Case UC_003
Exceptions	EX1. The reading process is interrupted 1. Request reading again.

Name	Count ballots (OPL)
ID	UC_003
Description	Read the ballots file and count the ballots for candidates and parties
Actors	Programmer
Organizational Benefits	Help the system organize the ballots and use the results in the next step
Frequency of Use	Every time when running an OPL
Triggers	The programmer chooses to count the ballots
Preconditions	The voting result file is loaded to the system successfully
Postconditions	The ballots for each party and each candidates have been counted
Main Course	1. System determines if the voting type is open party list 2. System reads the number of seats 3. System reads the number of ballots 4. System reads the number of candidates 5. System reads candidates' names and corresponding parties 6. System reads the names of candidates and their corresponding parties 7. Traverse each voting and add 1 to the corresponding candidate and party in a dictionary 8. System write the ballots of parties and candidate in an audit file 9. Repeat step 6,7 until reach the last line of the file 10. System sorts the candidates' votes for each party 11. System returns the dictionary

Alternative Course	None
Exceptions	None

Name	Allocate seats to parties
ID	UC_004
Description	After counting the total ballots for each candidate or each party, we need to allocate the seats for each party.
Actors	Programmers
Organizational Benefits	Help the system organize the election and get the results
Frequency of Use	Once after finishing ballots counting based on the given file
Triggers	System gets the counting result
Preconditions	Counting process has been finished, and already had the countings for parties or candidates
Postconditions	Allocates seats to each party based on their ballots counting
Main Course	1. System calculates the "floor (quota)" for the selection which is determined by taking the total number of votes in the district and dividing this by the number of seats. 2. System reads the names of parties as first column and corresponding ballots countings 3. System calculates and stores the first allocation of seats for each party by dividing the number of the votes by the quota(see AC1) 4. System stores the reminders as remainder votes 5. System sorts the remaining votes from largest to smallest 6. System adds 1 seat as second allocation seats to the party which has the largest remainders then remove the largest until no more seats are remained (see AC2) 7. System calculates and stores the addition of first and second allocation results for parties as final seats total
Alternative Course	AC1. The first allocation seats are greater or equal to the total number of candidates in that party 1. System changes the first allocation vote of that party to the number of candidates and the corresponding reminder vote to 0. 2. Return to main course step 5

	AC2. Two or more parties have the same remainders and only one of them can be allocated seat 1. See "Flip coin" use case 2. Return to main course step 7
Exceptions	None

Name	Flip a coin
ID	UC_005
Description	If the reminder votes of two or more parties reach a tie but the remaining seat number is less than the tied parties, then the system will flip coins to determine the seat allocation
Actors	Programmers
Organizational Benefits	A helper user case for seat allocate user case
Frequency of Use	Once a tie is reached
Triggers	Two or more parties have the same reminder votes
Preconditions	During the seat allocation, there is a tie between parties
Postconditions	Coin(s) flipped and the winner(s) of flipping coins generated
Main Course	1. System picks two of the parties randomly. and assigns head or tail to these two parties 2. System flip a coin and pick the one of the party (see EX1) 3. System adds 1 to the second allocation vote of the picked party 4. System repeat step 1-3 among the winner of last flipping and the remaining parties (see AC1) 5. Return to User Case UC_004 Main Course step 7
Alternative Course	AC1 There are no more seats 1. Returns to Main course step 5
Exceptions	EX1. Process interrupted. 1. Flip it again

Name	Generate winners
ID	UC_006
Description	After processing the seat allocation, the system needs to generate the winners
Actors	Programmers
Organizational Benefits	Winners come up, System gets the final election result
Frequency of Use	Once the process of allocating seats finished.
Triggers	The process of allocating seats has been finished
Preconditions	The process of allocating seats has been finished
Postconditions	The winners are generated and be displayed on the screen
Main Course	1. For a specific party, system reads the allocation votes total(n) for that party and reads first n candidates as the winners in that party. 2. System repeats step 1 for every party to generate the results of the final winners.
Alternative Course	None
Exceptions	None

4.3 Write an Audit File

4.3.1 Description and Priority

After the system generates the voting results, the user will need to produce an audit file with the election information at the time (e.g. Type of Voting, Number of Candidates, Candidates, Number of Ballots, calculations, how many votes a candidate or party had, etc). The audit file should also list the winner(s), and show how the election progressed so that the audit could replicate the election itself. It should also show who got what ballot and its order of being received if applicable.

This is also a core system feature in our system because audit is one of the most important parts of voting. Thus, this has a high priority.

benefit(9), penalty(9), cost(2), risk(2)

4.3.2 Stimulus/Response Sequences

Programmers: Authorize the system to generate a template of the audit file

System: Generate an audit file template and update it with seats

4.3.3 Functional Requirements

Name	Write in the audit file (OPL)
ID	UC_007
Description	Write the step-by-step result into the audit file
Actors	Programmers
Organizational Benefits	The programmers, testers, and officials are able to see the process of delivering seats
Frequency of Use	Every time a main function returns a result
Triggers	System prompts to write into the audit file
Preconditions	System has already started
Postconditions	The audit file is saved in the local directory with the input file. Everything related to the voting process will be included in the audit file.
Main Course	1. System is authorized to create the audit file (see EX1) 2. System starts to count the ballot for parties and candidates 3. The audit file is updated with total ballots for each party and each candidate 4. System starts to allocate seats based on the ballots for parties 5. The audit file is updated with every process of allocating seats (see AC1) 6. System starts to generate the winners 7. The audit file is updated with the winners 8. System shuts off the access to edit the audit file (see EX2) 9. System saves the file in the same local directory with input file
Alternative Course	AC1: system detects the process of allocation failed: 1. the audit file will include some relevant information.
Exceptions	EX1. System is not authorized to manipulate the audit file: 1. Halt the process of allocating seats 2. Give off error notifying messages EX2. System can not shut down the access of the audit file: 1. Output the current result 2. Shut off the abnormal process by killing relevant signal

Name	Write in the audit file (CPL)
ID	UC_008

Description	Write the step-by-step result into the audit file
Actors	Programmers
Organizational Benefits	The programmers, testers, and officials are able to see the process of delivering seats
Frequency of Use	Every time the main function returns a result
Triggers	System prompts to write into the audit file
Preconditions	System has already started
Postconditions	The audit file is written step by step to show every progress of allocating seats.
Main Course	1. System is authorized to create the audit file (see EX1) 2. The audit file is updated with the position of each candidate in each party 3. System starts to count the ballot for parties 4. The audit file is updated with total ballots for each party 5. System starts to allocate seats based on the ballots for parties 6. The audit file is updated with every process of allocating seats (see AC1) 7. System starts to generate the winners 8. The audit file is updated with the winners 9. System shuts off the access to edit the audit file (see EX2) 10. System saves the file in the same local directory with input file
Alternative Course	AC1: system detects the process of allocation failed: 1. the audit file will include some relevant information.
Exceptions	EX1. System is not authorized to manipulate the audit file: 1. Halt the process of allocating seats 2. Give off error notifying messages EX2. System can not shut down the access of the audit file: 1. Output the current result 2. Shut off the abnormal process by killing relevant signal

4.4 Display and Save Results

4.4.1 Description and Priority

After the election is complete, the results should be shared with the media and the public. Therefore, the system should be able to generate a summary report about the results. The summary report is a text file, which includes result details, i.e., winners, parties, percentages of votes, etc. This summary report is different from the audit file since the media and public do not need a huge audit file. After the results file is generated, the system should display the result to the screen the winner(s) and the information about the election (e.g. type of election,

number of seats). It will show the number of ballots cast, the winners and the stats for everyone or the party, such as the number of votes received.
This feature has a medium priority since this feature is a direct way to show the voting result.
benefit(5), penalty(5), cost(2), risk(2)

4.4.2 Stimulus/Response Sequences

System: After the seats are allocated and winners are generated, the system will print the results to the screen

4.4.3 Functional Requirements

Name	Display and save the results
ID	UC_009
Description	Print the results on screen. And save it as a txt file in local directory.
Actors	Programmers
Organizational Benefits	Can show the election result to users
Frequency of Use	Once after the winners have been generated.
Triggers	Once run the election.
Preconditions	The seats are allocated and winners are generated
Postconditions	The results are displayed on the screen, and a txt file is saved in local.
Main Course	1. System creates a new txt file for result 2. System retrieves and writes the type of voting to the file 3. System retrieves and writes the number of seats to the file 4. System retrieves each applicants' names and corresponding ballots for them 5. System writes the information in step 3 to the screen grouped by different parties 6. System retrieves the winners' names and writes the names to the file 7. System saves the result txt file to the local computer 8. System prints the file to screen line by line (see EX1)
Alternative Course	None
Exceptions	EX1. Display equipment meets error 1. Request displaying again 2. Reboot the device 3. Initialize backup device

5. Other Nonfunctional Requirements

5.1 Performance Requirements

There is a requirement on the runtime. The system should be able to run 100000 ballots under 5 minutes.

Time constraints for coding: 2 weeks to implement the system.

No other specific performance requirement for this system.

5.2 Safety Requirements

No special safety requirement. The specific safety requirement will be done in the voting center before sending the information to us.

5.3 Security Requirements

The voting file will be located in the same directory as the program. There are no specific security requirements, and everything required for security will be handled in the voting center.

5.4 Software Quality Attributes

Our system should be easy to replicate for different levels of users. This system also requires users to have a basic knowledge of two kinds of voting procedures. A new user can replicate the voting results after following the audit file given to them. To achieve this goal, the audit file should be detailed and understandable for new users to follow.

Also, for a new user who wants to use this system, we will provide a detailed user manual which will provide every level of users a general overview of the functions in this system.

5.5 Business Rules

We do not need any specific business rules to follow.

6. Other Requirements

Appendix A: Glossary

- Voting System: Election Voting System, which is used by election officials during the election.
- CSV: a kind of file format, which can be generated by Excel
- CPL: stands for “closed party list”. In a closed party list system--the original form of party list voting--the party fixes the order in which the candidates are listed and elected, and the voter simply

casts a vote for the party as a whole. Voters are not able to indicate their preference for any candidates on the list but must accept the list in the order presented by the party.

- OPL: stands for "open party list". In an open party list system, voters are allowed to express a preference for particular candidates, not just parties. It is designed to give voters some say over the order of the list and thus which candidates get elected.
- Allocate Seats: After voting, each party receives the number of seats they win. There are lots of different formulas for the allocation of seats to the parties. One is called the "largest remainder formula," which is also used in this voting system. In this approach, the first step is to calculate a quota, which is determined by taking the number of valid votes and dividing it by the number of seats. The party wins one seat for each whole number produced. After the first allocation of seats is complete, the remainder numbers for the parties are compared and the parties with the largest remainders are allocated the remaining seats. Finally, all the parties win the number of seats that as closely as possible approximate their percentage of the vote.

Appendix B: Analysis Models

None

Appendix C: To Be Determined List

None