Software Requirements Specification

for

Sunshine Vote Aggregation System

Version 1.0 approved

Prepared by Cole Thompson(thom6401), Ioana Munteanu(munte029), Thomas Rooney(roone194), and Trevor Guy(guyxx080)

UMN CSCI5801 Group 5

February 19th, 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Trevor	2/2/2021	Document Synthesis	0.1
Trevor	2/17/2021	Information add	0.2
Ioana	2/19/2021	Adding use cases and final formatting	1.0

1. Introduction

1.1 Purpose

This document stands as a detailed description of the Sunshine Vote Aggregation System Version 1.0. The software will implement two distinct voting algorithms, Instant Runoff Voting and Party List Voting. The act of voting will be done outside this system, and the product will use a given comma delimited text file as input to apply the voting algorithms. The system will be able to detect which voting algorithm is being used. Following the conclusion of the system run, two distinct output files will be generated, a media file and an audit file. These files will both contain the results of the election, and the audit file will additionally possess data on the intermediate steps in the aggregation process. The winners and information about the election will be displayed to the screen.

1.2 Document Convention

This document's formatting follows that of the IEEE template for System Requirements Specifications.

1.3 Intended Audience and Reading Suggestions

This document is intended for all members of the delivery team, including but not limited to developers, project managers, and testers. Audit and media files produced by the system will be available for all to view.

1.4 Product Scope

This software will function as a means of establishing results following an election held in either IR or OPL format. The software will produce an audit file upon completion which may be viewed by voters, election officials, or government officials. This audit file may be of use to future campaigns, and data can be collected from within for political advertising. Additionally, a report of the election results will be generated. This will be shared with the media. The winners and information about the election will be displayed to the screen.

1.5 References

Several references to other documents have been made during the synthesis process of this Software Requirements Specifications. Firstly, major guidance has been drawn from the assignment document that was posted to canvas via Dr. Watters. Second, software and hardware specifications were drawn from the University of Minnesota -- Twin Cities College of Science and Engineering Information Technology Department website that discusses the discrete specifications of all the existing CSE lab machines. An example SRS on Gephi was referenced for use with formatting. Example use cases were also referenced to create the use cases that exist within the appendix. Algorithms were constructed based on election type definitions defined at FairVote.org.

2. Overall Description

2.1 Product Perspective

The Sunshine Vote Aggregation System is a new election software that will assist with running both Instant Runoff and Open Party List elections. The system will be able quickly and accurately run both types of elections while presenting audit and media files upon conclusion to allow for future references to previous elections. The algorithms that dictate the election process mirror those described at FairVote.org.

2.2 Product Functions

- User will start software
- User will input the ballot data via a CSV file
- System will determine the type of election
- System will run appropriate election algorithm
- Winner(s) of the election will be displayed
- Audit and media files will be produced

2.3 User Classes and Characteristics

- Voting officials who will be using the system to determine the winner/winners.
- System testers who will be writing unit and integration tests to validate the system.
- Programmers who are interested in bug fixes and further development of the system.

2.4 Operating Environment

- Windows 10 (64 bit)
    - Dell Precision 3630, Intel® Core™ i7 @ 3.2GHz (x6), 32 GB RAM
    - Dell Precision T3420, Intel® Core™ i5 @ 3.4GHz (x4), 32 GB RAM
    - Dell OptiPlex 9020, Intel® Core™ i7 @ 4.2GHz (x4), 32 GB RAM
- Ubuntu
    - Dell XPS 8910, Intel Core i7-6700 CPU 3.40GHz, 16GB RAM, Nvidia GeForce GTX 1080
    - Dell Precision 3630, Intel Core i7-4790 (Quad-Core) 3.60GHz, 32 GB RAM, DVD+/-RW, Intel® UHD Graphics 630
    - Dell OptiPlex 7050, Intel Core i7-7700 3.6GHz, 32 GB RAM
- Ubuntu 18.04
    - Dell Optiplex 9020, Intel® Core™ i7 @ 900Hz (x3), 32 GB RAM
    - Dell Precision 3630 Intel® Core™ i7 @ 3.2GHz (x6) 32 GB RAM
    - Dell Precision T1700 Intel® Core™ i7 @ 3.6GHz (x4) 32 GB RAM
- Mac OS X
    - iMac, Intel Core i5 3.2 GHz, 16 GB RAM, DVD-RW, 2 x NVIDIA® GeForce GT 755M 1024MB

Project requirements state that the voting system must be able to run on a CSELab machine. This implies that the system will be compatible with the above listed Operating Systems.

2.5 Design and Implementation Constraints

Depending on the number of votes in a single election, the Sunshine Vote Aggregation System may take longer to complete counting. The counting system will have a run time of O(n) for OPL elections and O(n) for IR elections. The Sunshine Vote Aggregation System is able to process 100,000 votes in 8 or less minutes.

2.6 User Documentation

We will have a README file and a Javadoc document.

2.7 Assumptions and Dependencies

The Sunshine Vote Aggregation System is developed in Java and thus requires Java to be installed on the users system.

3. External Interface Requirements

3.1 User Interfaces

The Sunshine Vote Aggregation System utilizes Java Swing to make its GUI. The system displays a prompt to the user for them to input the name of the file to read. Once completed, the system prompts the user to continue, and the vote aggregation system is run.

3.2 Hardware Interfaces

The Sunshine Vote Aggregation System is only to be run on the University of Minnesota lab machines and must be compatible with them.

3.3 Software Interfaces

The Sunshine Vote Aggregation System uses the latest version of Java (version 8 update 281) and requires it to be installed on the system.

3.4 Communications Interfaces

The Sunshine Vote Aggregation System requires a connection to the University of Minnesota's lab machines to run. If run directly from a lab machine, no internet connection is required as long as the program is installed.

4. System Features

This section demonstrates our vote aggregation system's most notable features and elaborates on its functions and use cases.

4.1 Input and Parsing of an Election Data CSV File

4.1.1 Description and Priority

This system will take in a CSV file complete with election data that has been gathered at the election centers. The file will contain the type of election on the first line ("IR" or "OPL"). The second line will contain the number of candidates. The third line will contain the candidates and the parties they belong to (separated by comma in an IR election and between brackets in a OPL election). The fourth line for an IR election will contain the number of ballots in the file, whereas the same line for a OPL election will contain the number of seats. The fifth line in an OPL election file will contain the number of ballots. For both types of election, the rest of the file will contain the ballots, one per line. Next, the system will parse the data, determine what type of election has been run, assign the votes to different candidates, and move onto the next system features. This feature is a high priority garnishing a benefit of 9, cost of 5, and risk of 5.

4.1.2 Stimulus/Response Sequences

A user will input a CSV file of election data. The system will then read the first line of the file and determine what type of voting system that will be used to calculate the results. The system will then move onto the next system features based on the result of the first line.

4.1.3 Functional Requirements

REQ-1: File input. The system properly takes in a file and reads the data from it.
REQ-2: Voting system is correctly discerned. The system reads the first line and correctly determines what voting system code to execute.

4.2 Instant Runoff Voting

4.2.1 Description and Priority

Instant runoff voting is a voting system that never requires a revote, a winner will always be selected within the “instant runoff”. This type of voting calls for the ranking of candidates in the order of your preference that they win the election. If a candidate receives a greater than 50% majority of votes then the candidate is declared the winner. If there is no majority then the candidate with the least number of votes is excluded and the alternate vote is carried to the remaining candidates. This process repeats until a winner is declared. If a tie occurs then a coin is flipped to determine a winner. This feature is a high priority garnishing a benefit of 9, cost of 6, and risk of 5.

4.2.2 Stimulus/Response Sequences

The system will read all the ballots and update a data structure for a given candidate if they received a vote. The system will determine if a greater than 50% number of primary votes is cast to one candidate, if so the candidate will be determined the winner. The other case is to exclude the candidate with the least number of votes and carry the alternate votes to the remaining candidates. This process is then repeated until a winner is determined. In the case of a tie between two candidates a coin is flipped.

4.2.3 Functional Requirements

REQ-1: Read ballots in the form of parsing the CSV file.
REQ-2: Update the data structure with the candidate class if the candidate received a vote.
REQ-3: Determine if a candidate got the majority number of votes.
REQ-4: Exclude the candidate with the least number of votes.
REQ-5: Carry over the alternate vote.
REQ-6: Flip a coin if there is a tie.
REQ-7: Determine a winner.

4.3 Open Party List Voting

4.3.1 Description and Priority

Open party list voting is a voting system used in most European democracies. It consists of parties providing a list of candidates, then the voter casts one vote which counts toward the candidate and the party. After all the votes are in, a quota is determined based on dividing the total number of votes cast by the number of open seats being voted for. The number of votes for the party is then integer divided by the quota and the quotient is how many seats are rewarded to the party. Then within the party the candidates are ranked based on how many votes they received and the top candidates are awarded the seats that the party earned. This is repeated with all the parties. If seats remain after this, the remainders are compared and the highest remainders win seats. This feature is a high priority garnishing a benefit of 9, cost of 6, and risk of 6.

4.3.2 Stimulus/Response Sequences

The system will read all the ballots and update a data structure for a given candidate and the candidates party if they received a vote. The system will determine the quota by which to run an integer division and modulus operation on each of the parties vote counts. The quotient grants the party with the number of seats they have won. If there are still remaining seats then the remainders of the parties are ranked and the parties with the highest remainder fill the remaining seats.

4.3.3 Functional Requirements

REQ-1: Read ballots in the form of parsing the CSV file.
REQ-2: Update the data structure with the candidate class if the candidate received a vote.
REQ-3: Update the data structure of the party class if their candidate received a vote.
REQ-4: Calculate the quota.
REQ-5: Integer divide candidate vote count by quota.
REQ-6: Perform modulus operation on candidate vote count by quota.
REQ-7: Rank candidates within party by vote count.
REQ-8: Rank remainder vote counts by party.
REQ-9: Assign the correct number of seats to each party to the party members with the most votes.

4.4 Results Generation

4.4.1 Description and Priority

The results of the election will be generated by showing the result on the screen and producing two files: an audit file and a media file. The results on the screen will show the winner/winners and information about the election (type of election, number of seats). The audit and media files will also be generated at the completion of the voting aggregation. The audit file produces the results and data from an election such as the type of voting, the number of candidates, who the candidates are, the number of ballots, the calculations that were made, and winner/winners of the election. This file is an in-depth look at how the program executed the aggregation. The media file provides the same relevant information about the results as the result on the screen shows. This feature is a high priority garnishing a benefit of 10, cost of 6, and risk of 5.

The names of the files generated will be unique. The audit file will be named "AUDITyyyymmddhhMMss" and the media file will be called "MEDIAyyyymmddhhMMss", where "yyyy" is the current year, "mm" is the current month, "dd" is the current day, "hh" is the current hour, "MM" is the current minute and "ss" is the current second.

4.4.2 Stimulus/Response Sequences

The results of the election will be gathered in the previous system features. The relevant data points will be written to the screen, to the audit file, and to the media file. The results will be shown to the users screen and the user will have the option to view and export the audit and media files.

4.4.3 Functional Requirements

REQ-1: Data gathered are written to the screen.
REQ-2: Data gathered are written to the audit file.
REQ-3: Data gathered are written to the media file.
REQ-4: Results are shown to the user's screen.
REQ-5: The audit file can be viewed by the user.
REQ-6: The audit file can be exported by the user.
REQ-7: The media file can be viewed by the user.
REQ-8: The audit file can be exported by the user.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

An election should be able to run 100,000 ballots in under 8 minutes.

5.2 Safety Requirements

There are no special safety requirements.

5.3 Security Requirements

There are no special security requirements. Security such as ensuring one vote for one person is handled at the voting centers. Thus any type of user can use it without any additional privileges.

5.4 Software Quality Attributes

Our voting system program is very easy to use. The user does not need to have any knowledge about the voting system itself. They just need to know the name of the file containing the ballots, introduce it into our program, and our program will generate the results.

5.5 Business Rules

Programmers, testers, and election officials will run and use this program. The results of the election could be shared with media personnel.

6. Other Requirements

Appendix A: Glossary

CSE	College of Science and Engineering
CSV	Comma Separated Values
GUI	Graphical User Interface
IR	instant runoff voting (a type of voting where majority wins)
OPL	open party list voting

Appendix B: Analysis Models

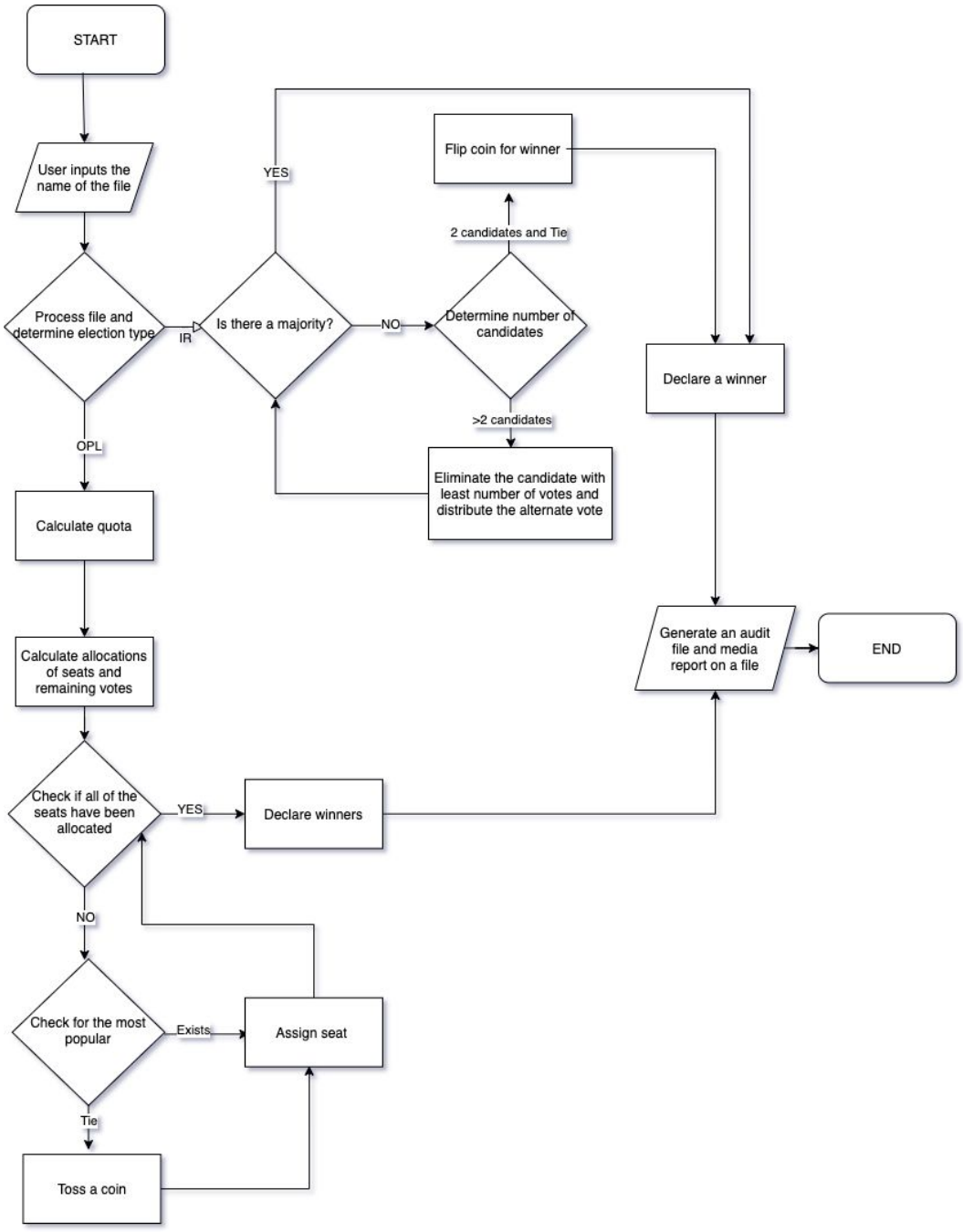

Figure 1 *Flowchart of Sunshine Vote Aggregation System*

Appendix C: Use Cases

Name	Identify the file name
ID	IDENTIFY_FILE_001
Description	The user inputs the name of the CSV file containing election data.
Actors	Election official
Organizational Benefits	This step starts the process of running the election.
Frequency of Use	This will be done once for each election run.
Triggers	An election has been conducted, and ballots were stored in a CSV file as described in the assignment write-up.
Preconditions	It was a fair IR/OPL election. The CSV file has the right format. The election official has authorization to count the votes.
Postconditions	The file will be read into the system for the Sunshine system to use.
Main Course	1. The user opens the Sunshine Vote Aggregation System. 2. The system prompts the user for the name of the file with a GUI. 3. The user correctly inputs the name of the file. (see EX2)
Alternate Courses	None
Exceptions	EX1 The system crashes. 1. The user should restart the system EX2 The file name was incorrect or the wrong file was inputted. 1. Return to MC2.

Name	Process the File IR
ID	PROCESS_FILE_IR_001
Description	The input file containing the election data is processed. Candidate objects are created for use with future functions with correct attributes such as name, party, and current votes.
Actors	Election official
Organizational Benefits	This process synthesizes the necessary objects to run the remainder of the algorithms present within the Sunshine system.
Frequency of Use	This will be done once for every election which has been determined to be an Instant Runoff election.
Triggers	A CSV file has been inputted into the system. The first line of the file reads “IR”.
Preconditions	It was a fair IR election. The CSV file has the right format. The election official has authorization to count the votes. The user has inputted the file name into the system and it has been accepted.
Postconditions	All candidates present within the ballot file will be instantiated into objects. Each candidate object will contain the candidates correct name, party affiliation, and number of votes.
Main Course	1. The system will parse through the second and third line of the ballot file and instantiate all present candidates. 2. Write the number and names of candidates in the audit and media files. 3. The system will parse through the rest of the file and add votes to each candidate as deemed by the ballot file. 4. Write the number of votes each candidate has in the audit file.
Alternate Courses	None
Exceptions	EX1 The system crashes. 1. The user should restart the system.

Name	Check if there is a winner IR
ID	CHECK_WINNER_IR_001
Description	An election official checks if there is a winner in an IR election.
Actors	Election official
Organizational Benefits	Allows election officials to declare a winner of the election.
Frequency of Use	It will be used during an IR election after processing the file and every time after the votes are redistributred.
Triggers	The file has been processed or a candidate has been eliminated.
Preconditions	It was a fair IR election. The CSV file has the right format. The election official has authorization to count the votes.
Postconditions	The number of votes each candidate has remains the same.
Main Course	1. The system calculates what number represents the majority of the votes. 2. The system checks if any candidate has the majority of votes and returns the answer. (see AC1). 3. Write the number of the check and its answer in the audit file.
Alternate Courses	AC1 There are two candidates left, with the same number of votes. (Tie) 1. The system will return that there is a winner. (that will be determined by flipping a coin; See “Flip a coin” use case)
Exceptions	EX1 The system crashes. 1. The user should restart the system.

Name	Eliminate unpopular candidate IR
ID	ELIMINATE_UNPOPULAR_IR_001
Description	An election official checks which candidate has the least number of votes in an IR election and redistributes their votes to the remaining candidates.
Actors	Election official
Organizational Benefits	Allows election officials to eliminate the least popular candidate's votes, redistribute their votes, and be closer to declaring the winner of the election.
Frequency of Use	It will be used during an IR election only if no candidate has the majority.
Triggers	No candidate has the majority.
Preconditions	It was a fair IR election. The CSV file has the right format. The election official has authorization to count the votes.
Postconditions	The candidate/One of the candidates with the least number of votes is eliminated. The number of votes some candidates have may increase.
Main Course	1. The system orders the candidates based on the number of votes. 2. The system finds the candidate with the least number of votes. 3. The system eliminates the candidate. 4. The system redistributes the eliminated candidate's votes (using the next preference expressed on the ballot). 5. Write the number of the elimination, the new list of candidates and the new distribution of votes in the audit file.
Alternate Courses	AC1 There are multiple candidates with the least number of votes. 1. Flip a coin to determine who will be eliminated. (See "Flip a coin" use case) AC2 There is no next preferred candidate on a ballot. 1. The ballot won't be used in the election.
Exceptions	EX1 The system crashes. 1. The user should restart the system.

Name	Process the File OPL
ID	PROCESS_FILE_OPL_001
Description	The input file containing the election data is processed. Candidate objects are created for use with future functions with correct attributes such as name, party, and current votes. Party objects are instantiated containing attributes that track the name of the party and the sum of votes received for all candidates that belong to it.
Actors	Election official
Organizational Benefits	This process synthesizes the necessary objects to run the remainder of the algorithms present within the Sunshine system.
Frequency of Use	This will be done once for every election which has been determined to be an Open Party List election.
Triggers	A CSV file has been inputted into the system. The first line of the file reads “OPL”.
Preconditions	It was a fair OPL election. The CSV file has the right format. The election official has authorization to count the votes. The user has inputted the file name into the system and it has been accepted.
Postconditions	All candidates present within the ballot file will be instantiated into objects. Each candidate object will contain the candidates correct name, party affiliation, and number of votes. Also, party objects will be created. These objects will contain the name of the party, and the total number of votes won by that party.
Main Course	1. The system will parse through the second and third line of the ballot file and instantiate all present candidates and parties. 2. Write the number and names of candidates in the audit and media files. 3. The system will use the fourth line to determine the number of available seats. 4. Write the number of seats available in the audit and media files. 5. The system will parse through the remainder of the file and add votes to each candidate as deemed by the ballot file. Each vote will also be given to a specific party. 6. Write the number of votes each candidate has in the audit file.
Alternate Courses	None
Exceptions	EX1 The system crashes. 1. The user should restart the system.

Name	Group Independents into a Single Party
ID	GROUP_INDEPENDENTS_001
Description	Independents that are present within a given OPL election will be grouped into a single party.
Actors	Election official
Organizational Benefits	This allows for seats to be accurately handed out to independent candidates following OPL protocol.
Frequency of Use	This will be done once for every election which has been determined to be an Open Party List election.
Triggers	There is at least one candidate that doesn't belong to a major party.
Preconditions	It was a fair OPL election. The CSV file has the right format. The election official has authorization to count the votes. The user has inputted the file name into the system and it has been accepted.
Postconditions	A new party object will be instantiated named "Independent". This will have an attribute that denotes the total number of votes received by all independent candidates in the election.
Main Course	1. The system will check the currently instantiated candidates for any independents. (SEE AC1) 2. The system will instantiate a new object for the conglomerate "Independent" party. 3. The system will sum the votes of all independent candidates, the sum will be placed in an attribute of the object. 4. Write the number of votes each party has in the audit file.
Alternate Courses	AC1 1. There are no Independent candidates in the election. 1. No object will be created.
Exceptions	EX1 The system crashes. 1. The user should restart the system.

Name	Calculate quota for OPL
ID	CALCULATE_QUOTA_OPL_001
Description	An election official calculates the quota for OPL.
Actors	Election official
Organizational Benefits	Allows election officials to know what to divide and mod each party's votes to gather seats and remaining votes.
Frequency of Use	It will be calculated once during an OPL election.
Triggers	The total number of votes cast is calculated and the total number of seats to be filled is given.
Preconditions	It was a fair OPL election. The CSV file has the right format. The election official has authorization to count the votes. All the votes cast have been properly accounted for. The correct number of seats to be filled is provided by the csv file.
Postconditions	The quota is correctly calculated.
Main Course	1. The system counts the total number of votes cast 2. The system stores the total number of seats to be filled. 3. The system divides total votes by total seats to be filled rounded down. 4. The system writes the quota to the audit file.
Alternate Courses	None
Exceptions	EX1 The system crashes. 1. The user should restart the system.

Name	Calculate allocation of seats and remainders for OPL
ID	CALCULATE_ALLOCATIONS_AND_REMAINDERS_OPL_001
Description	An election official calculates the allocation of seats and remaining votes for OPL.
Actors	Election official
Organizational Benefits	Allows election officials to determine seat allocations.
Frequency of Use	It will be used once during an OPL election after integer dividing and modding the party votes by the quota.
Triggers	The quota has been properly calculated.
Preconditions	It was a fair OPL election. The CSV file has the right format. The election official has authorization to count the votes. The quota has been properly calculated. The number of votes for each party is calculated. The number of votes for each candidate is calculated.
Postconditions	The allocations of the seats have been properly declared.
Main Course	1. The system integer divides each party's vote count by the quota. 2. The system allocates seats to each party based on the result of MC1. For every multiple of the quota obtained, a party will be awarded a seat. 3. The system ranks party candidates by vote and selects the top candidates for how many allocations that party has been awarded. These candidates occupy the seats. 4. If there are still seats to be allocated, the system mods each party's vote count by the quota. 5. The system allocates the remaining seats by the highest results from MC4. 6. The system will write the first allocations, remaining votes, and second allocations to the audit file.
Alternate Courses	AC1 A party gets awarded more seats than they have candidates. 1. The party fills all their candidates in seats and coins are flipped for the remaining seats. (See "Flip a coin" use case) AC2 There are more seats than candidates. 1. Every candidate gets a seat.
Exceptions	EX1 The system crashes. 1. The user should restart the system.

Name	Flip a coin for a tie
ID	FLIP_COIN_001
Description	A coin is flipped if there is a tie.
Actors	Election official
Organizational Benefits	Allows election officials to declare a winner of the election in the case of a tie.
Frequency of Use	It will be used during the IR and OPL elections.
Triggers	It will be used during an IR election when two candidates remain and each earned 50% of the vote. It will be used during an OPL election, during the allocation of the remainders, when there is a tie between parties with less seats available than parties.
Preconditions	It was a fair election. The CSV file has the right format. The election official has authorization to count the votes. A tie occurred.
Postconditions	A winner from the coin toss is fairly determined.
Main Course	1. The system calculates if there is a tie in IR or OPL. 2. The system produces a random number generator for the candidates and selects a winner.
Alternate Courses	AC1 There are more than two candidates or parties tied. 1. OPL: The system will determine how many seats are left to fill and run the coin flip until all the seats are filled. 2. IR: The system will determine the candidate to be eliminated if no majority has been reached.
Exceptions	EX1 The system crashes. 1. The user should restart the system.

Name	Display Winner(s)
ID	DISPLAY_WINNER_001
Description	If an IR election takes place, then 1 winner will be displayed to the screen at the end of the counting process. If it's an OPL election, multiple winners will be displayed to the screen equal to the number of seats available. Other information about the election will also be displayed (type of election, number of seats).
Actors	Election official
Organizational Benefits	Doing this clearly displays to the user who the winner/winners are, and other useful information
Frequency of Use	The winner(s) will be displayed once at the end of the counting.
Triggers	The voting system successfully found a winner/winners.
Preconditions	A correctly formatted CSV file must be passed into the system.
Postconditions	For IR only 1 winner is displayed, whereas for OPL multiple winners will be displayed.
Main Course	1. After a winner/winners have been found, the system displays them to the screen. 2. The system prints information about the election (see AC1) 3. Append this information to the audit and media files.
Alternate Courses	AC1 System determines there's more seats than candidates in OPL election 1. System also displays how many seats are still open
Exceptions	EX1 System crashes 1. The user should restart the system.

Name	Produce audit file
ID	PRODUCE_AUDIT_001
Description	After the system is ran, an audit file with information on the election is produced
Actors	Voting official
Organizational Benefits	All the information about the election is organized neatly in one space
Frequency of Use	An audit file will be produced once per election
Triggers	The voting system is successfully run.
Preconditions	A correctly formatted CSV file must be passed into the system
Postconditions	An audit file is generated and contains information about the election that is useful to election officials.
Main Course	1. When a candidate is selected or eliminated, the system updates an ongoing audit file (see AC1) 2. Once the system finishes determining all winners, the audit file is released to voting officials
Alternate Courses	AC1 More seats than candidates in OPL election 1. System will note in audit file of open spaces
Exceptions	EX1 System crashes 1. User needs to rerun the system

Name	Produce media report
ID	MEDIA_REPORT_001
Description	After the system is ran, a media report is produced with information about the election for the media
Actors	Voting official
Organizational Benefits	Collects the information that the media cares about and puts it in one place, such as who won, how many votes they received, the time the election took place, etc.
Frequency of Use	A media file will be produced once per election.
Triggers	The voting system is successfully run.
Preconditions	A correctly formatted CSV file must be passed into the system.
Postconditions	A media report is created that contains information that is useful to the media.
Main Course	1. Once the system declares a winner/winners, a media file is generated (see AC1) 2. Media file is distributed to media personnel.
Alternate Courses	AC1 System determines more seats than candidates in OPL election 1. System notes open seats in media report
Exceptions	EX1 The system crashes 1. User needs to rerun the system