**USE\_CASE 1**

|  |  |
| --- | --- |
| Name | User Enters Filename |
| ID | UC\_001 |
| Description | System opens the file from the filename given by the user, it will consist of comma-separated values for the ballots where each row is separated by a newline. |
| Actors | Programmers, Testers, Government Officials |
| Organizational Benefit | Reduces the time needed to accurately count ballots, improving productivity for Government Officials. |
| Frequency of Use | This will be used at least once a year for major elections |
| Triggers | The user starts the application. |
| Preconditions | * A window opens prompting the user for a filename. * The file the user is giving the name of is in the same directory as the program. * The input file is readable. |
| Postconditions | * The structure of the given file cannot be changed. * The input file data is not altered. |
| Main Course | 1. Display the “Input File” prompt 2. The user enters the filename that they want to open. (see AC1 and EX1) 3. System opens the file. (see UC\_002) |
| Alternate Courses | AC1: User decides not to enter a file name   1. Jump to Main Course step 1. |
| Exceptions | EX1: Cannot find the file        1. Prompt the user to retype filename.        2. Return to the Main Course step 2. |

**USE\_CASE 2**

|  |  |
| --- | --- |
| Name | System Opens File |
| ID | UC\_002 |
| Description | After the user enters the filename, then the system will open the file. |
| Actors | The System |
| Organizational Benefit | Opening the input file allows the system to read through it, allowing the ballots to be processed. This enables the system to computationally determine the election winner, improving productivity for Government Officials. |
| Frequency of Use | Whenever the program is run, and the user enters a file name. |
| Triggers | After the user enters a file name. |
| Preconditions | * The filename can be searched in the system successfully |
| Postconditions | * Input data is not altered. * The system can read the file. |
| Main Course | 1. User log into the system. 2. Users enter the filename that they want to open. (see UC\_001) 3. System opens the designated file. (see EX1, UC\_007) |
| Alternate Courses |  |
| Exceptions | EX1: System cannot open the file.   1. Inform user that the given filename could not be opened. 2. Return to Main Course step 2. |

**USE\_CASE 3**

|  |  |
| --- | --- |
| Name | Produce Audit File |
| ID | UC\_003 |
| Description | An audit file is created when UC\_004 begins to run, where all the election information is recorded into. Audit information includes the type of election, number of candidates in the election, candidate names, number of ballots, any calculations made, total number of votes a candidate has, and the winner of the election. |
| Actors | Programmers, Testers, Government Officials, Voting System |
| Organizational Benefit | This will reduce the time needed to audit the election if necessary, improving productivity for Government Officials. |
| Frequency of Use | This will be used at least once a year for major elections |
| Triggers | UC\_004 or UC\_005 are in effect. |
| Preconditions | * The input file has already been successfully loaded in by the user prior to this use case. * All necessary information to run the processing is contained in the input file (ie. closed vs. open party list). * Independent candidates are grouped into a single independent party. * System can create and write to a file. * Voting system is operating in conjunction. |
| Postconditions | * A single output file is produced in the current directory. * Output file created does not overwrite any existing files * Input data is not altered. * All proper steps of the voting process are recorded into the output file. * Output file is readable |
| Main Course | 1. New output text file is created. (see AC2) 2. Ballot is processed by UC\_004: (see UC\_004 Main Course step 2) 3. Ballot processing steps are recorded in the output text file. (see AC1) 4. Candidate (and party) with the most votes are recorded into the audit file. (See EX1) 5. Data stream to output file is closed. (See UC-006) |
| Alternate Courses | AC1: There are no ballots remaining.   1. Each candidate/party is recorded into the audit file with their total vote counts. 2. Return to Main Course step 4.   AC2: File currently exists with the designated filename.   1. Create new file with ‘(x)’ at the tail end, where x represents the increasing integer iteration of the file that has does not exist yet starting from ‘(1)’. 2. Jump to Main Course step 2. |
| Exceptions | EX1: There are tied candidates/parties   1. Presence of a ties is noted in audit file. 2. Results of the coin flip are recorded. 3. Jump to Main Course step 4. |

**USE\_CASE 4**

|  |  |
| --- | --- |
| Name | Run Vote Processing (Open Party) |
| ID | UC\_004 |
| Description | After all ballots for an election have been cast and converted into a comma-delimited text file, that file has been inputted into the system, and the election has been determined to be open party, a user runs the vote processing software. |
| Actors | Programmers, Testers, Government Officials |
| Organizational Benefit | This will reduce the time needed to accurately count ballots and determine the winner of an election to improve productivity for Government Officials. |
| Frequency of Use | This will be used once each time the software is run for a closed party election. There will be several major and special elections per year. |
| Triggers | The user clicks the “Process Votes” button. |
| Preconditions | * There are no errors in the input file. * There is exactly one input file. * The input file has already been successfully loaded in by the user prior to this use case. * All necessary information to run the processing is contained in the input file (ie. closed vs. open party list). * The first line of the input file specifies that the election uses open party list voting. * There is at least one ballot in the input file. * Independent candidates are grouped into a single independent party. |
| Postconditions | * The input file has not been altered. * An audit log detailing a vote-by-vote account of the election has been produced. * The results of the election, including the number of seats per party and the candidates appointed to those seats have been determined. |
| Main Course | 1. The input file is accessed 2. For each ballot contained in the input file, the system will:    1. Increase the vote count for the selected candidate by 1.    2. Increase the vote for the candidate’s party by 1.    3. Write the ballot number and selected candidate to the audit log. 3. Once the last ballot has been recorded, determine the number of seats that should go to each party based on the percentage of the total vote each party received. (See EX1) 4. Determine which candidates in each party are appointed to each seat based on the total number of votes each received. The top X candidates will be appointed, where X is the number of seats the party earned. (See EX2) 5. Display the relevant results to the user. 6. Prompt the user to send the results to the media. (See UC\_005) |
| Alternate Courses |  |
| Exceptions | EX1: There are tied parties   1. Simulate a fair coin flip to determine the winner. The user is not notified of this directly, although it is recorded in the audit log. 2. Jump to main course step 3.   EX2: There are tied candidates   1. Simulate a fair coin flip to determine the winner. The user is not notified of this directly, although it is recorded in the audit log. 2. Jump to main course step 4. |

**USE\_CASE 5**

|  |  |
| --- | --- |
| Name | Run Vote Processing (Closed Party) |
| ID | UC\_005 |
| Description | After all ballots for an election have been cast and converted into a comma-delimited text file, that file has been inputted into the system, and the election has been determined to be closed party, a user runs the vote processing software. |
| Actors | Programmers, Testers, Government Officials |
| Organizational Benefit | This will reduce the time needed to accurately count ballots and determine the winner of an election to improve productivity for Government Officials. |
| Frequency of Use | This will be used once each time the software is run for a closed party election. There will be several major and special elections per year. |
| Triggers | The user clicks the “Process Votes” button. |
| Preconditions | * There are no errors in the input file. * There is exactly one input file. * The input file has already been successfully loaded in by the user prior to this use case. * All necessary information to run the processing is contained in the input file (ie. closed vs. open party list). * The first line of the input file specifies that the election uses closed party list voting. * Independent candidates are grouped into a single independent party. * Each party has determined the order of their candidates prior to the election. * There is at least one ballot in the input file. |
| Postconditions | * The input file has not been altered. * An audit log detailing a vote-by-vote account of the election has been produced. * The results of the election, including the number of seats per party and the candidates appointed to those seats have been determined. |
| Main Course | 1. The input file is accessed 2. For each ballot contained in the input file, the system will:    1. Increase the vote for the candidate’s party by 1.    2. Write the ballot number and selected party to the audit log. 3. Once the last ballot has been recorded, determine the number of seats that should go to each party based on the percentage of the total vote each party received. (See EX1) 4. Determine which candidates in each party are appointed to each seat based on the order determined by the party prior to the election. The top X candidates will be appointed, where X is the number of seats the party earned. (See EX2). 5. Display the relevant results to the user. 6. Prompt the user to send the results to the media. (See UC\_006) |
| Alternate Courses |  |
| Exceptions | EX1: There are tied parties   1. Simulate a fair coin flip to determine the winner. The user is not notified of this directly, although it is recorded in the audit log. 2. Jump to main course step 3.   EX2: There are tied candidates   1. Simulate a fair coin flip to determine the winner. The user is not notified of this directly, although it is recorded in the audit log. 2. Jump to main course step 4. |

**USE\_CASE 6**

|  |  |
| --- | --- |
| Name | Request Media Results |
| ID | UC\_006 |
| Description | Once the results of an election have been calculated, the user would like to send the relevant data on the election to the media in a convenient report. |
| Actors | Programmers, Testers, Government Officials |
| Organizational Benefit | This will allow the efficient sharing of the important statistics of an election to be displayed by the media as soon as possible without forcing anyone to sift through the audit log. |
| Frequency of Use | This will be used at least once a year for major elections, and for any other special elections as well. |
| Triggers | The user clicks the “generate media report” button on the displayed results of an election. |
| Preconditions | * The results of the election have been determined. * The user’s computer has write access to the directory the software is run from. |
| Postconditions | * The results of the election have been condensed into a readable file containing the winner of the election and the percentage of the vote taken by each candidate and party. * The results of the election are not affected in any way. |
| Main Course | 1. Create a new text file called “<current date>\_<input filename>.txt” in the existing folder “media-reports”. This folder should be located in the same directory as the software. (See AC1 and EX1, EX2) 2. Write the total number of votes cast to the file. 3. The election was open-party-list. (See AC2). 4. For each party in order of votes received, the system writes the total number and percentage of votes and seats, as well as the candidates appointed to and not appointed to seats to the file. 5. Finalizes the text file and alerts the user of its location. (See EX3) 6. The results of the election are displayed. |
| Alternate Courses | AC1: The folder “media-reports” is not present   1. Create the folder “media-reports” in the same directory the software is being run from. (See EX1) 2. Jump to main course step 1.   AC2: The election was closed-party-list.   1. For each candidate in order of vote received, print the number and percentage of votes earned. 2. For each party, display the order of its candidates by votes received. 3. Jump to Main Course step 4. |
| Exceptions | EX1: There is an error creating the text file or the folder   1. Alert the user of this issue through a dialog box. 2. Jump to Main Course step 8.   EX2: A file with the same name as the text file already exists   1. Rename the existing file to “<filename>\_old.txt”. 2. Jump to main course step 1.   EX3: There is an error finalizing the text file.   1. Alert the user of this issue through a dialog box. 2. Jump to Main Course step 8. |

**USE\_CASE 7**

|  |  |
| --- | --- |
| Name | System Reads File |
| ID | UC\_007 |
| Description | System reads the given input file from the user. Each read line is then sent to Use Case 4 for processing. |
| Actors | The System |
| Organizational Benefit | This allows the system to process the lines from the input file, allowing the system to operate as intended and improving productivity for Government Officials. |
| Frequency of Use | This will be used in every case the program is run. |
| Triggers | User input a correct file name and the system was able to open it. |
| Preconditions | * The input file was able to be opened/ * All necessary information to run the processing is contained in the input file (ie. closed vs. open party list). * No errors are contained in the file. * Independent candidates are grouped into a single independent party. |
| Postconditions | * Each line of the file that is read is processed by the system. * Input data is not altered. |
| Main Course | 1. System opens file. (see UC\_002)  2. Each line is read, sending the results for processing. (see UC\_004/UC\_005, AC1)  3. Input file is closed. |
| Alternate Courses | AC1: There are no lines remaining.  1. Signal is sent to UC\_004/UC\_005 that there are no more lines in the file and to complete the election process.  2. Return to Main Course step 3. |
| Exceptions |  |

|  |  |
| --- | --- |
| Name | Make a run-off between tied candidates |
| ID | UC\_008 |
| Description | After the open party run vote processing(UC\_004) or closed party run vote processing(UC\_005), if there is ever a tie, then that the user needs to flip a coin to make a run-off between these tied candidates. |
| Actors | Programmers, Testers, Government Officials |
| Organizational Benefit | This will keep the fairness of the election. |
| Frequency of Use | This will be rarely used only when there are tied candidates. |
| Triggers | Tied candidates generated. |
| Preconditions | * The input file has already been successfully loaded in by the user prior to this use case. * All necessary information to run the processing is contained in the input file (ie. closed vs. open party list). * System can create and write to a file. * There are no errors in the input file. * There is exactly one input file. * There are at least two candidates in the input file. * A tie is generated. |
| Postconditions | * Input data is not altered. * All proper steps of the voting process are recorded into the output file. * Output file is readable. * Presence of a tie is noted in audit file. |
| Main Course | 1.Flip a coin.  2. Record the result of the coin flip. |
| Alternate Courses |  |
| Exceptions | EX1: The coin is upright.  3. Flip a coin again. |

**USE\_CASE 8**

**USE\_CASE 9**

|  |  |
| --- | --- |
| Name | System Identifying the Voting Type |
| ID | UC\_009 |
| Description | When accessing the ballot file, get the first line and classify the file as CPL if the fist line is “Closed Party List Ballot”, classify the file as OPL if the fist line is “Open Party List Ballot”. |
| Actors | Programmers, Testers, Government officials |
| Organizational Benefit | This will Improve the speed of extracting information and the speed at which files are read |
| Frequency of Use | This will be used at least once a year for major elections |
| Triggers | When the system accessing the vote file |
| Preconditions | * All necessary information to run the processing is contained in the input file (ie. closed vs. open party list). * Independent candidates are grouped into a single independent party. * The file is exported from Excel into the CSV format. |
| Postconditions | * System classifies the type of file * Process the CPL or OPL based on the classified type |
| Main Course | 1. User log into the system. (SEE AC2) 2. Users upload that they want to classify. (SEE AC 2) 3. System classify the file that they upload. (SEE AC1) |
| Alternate Courses | AC1: There are no ballots remaining.   1. Each candidate/party is recorded into the audit file with their total vote counts.   AC2: Users cannot log in to the system   1. Users back to the Main course1 and re-entered ID. |
| Exceptions | EX1: There are no ballots cast on the ballot   1. Each candidate/party is recorded into the audit file with their total vote counts. |