**Table of Contents**

**List of Figures**

**1.0 Introduction**

1.1 Purpose

1.2 Scope of Project

1.3 Glossary

|  |  |
| --- | --- |
| ***Term*** | ***Definition*** |
| Array | An ordered arrangement of user input date |
| Candidate | Person who is nominated for the voting election |
| Droop Quota Algorithm |  |
| Plurality Algorithm |  |
| Ballot List | The list of ballots imported from the input CSV file |
| Elected List | A First-In-First-Out (FIFO) queue of elected candidates |
| Non-Elected List | A First-In-Last-Out (FILO) stack of elected candidates |
|  |  |
| User | Person who runs the main voting program |

1.4 Overview of Document

**2.0 Overall Description**

2.1 System Environment

**3.0 Requirement Specification**

3.1 Functional Requirement

System requirement:

1. Validate CSV file [page1]
2. Run on a CSE lab machine in command line. [page1]
3. Each ballot must have at least one candidate ranked(被validate csv file使用) [page2]

User requirement:

1. Run voting process [4 inputs] [核心] [page1]d
2. Read CSV file[page2]

Functional requirement:

System:

1. **Tie Solver**: The system should be able to randomly generate a number from a given range generate a winner if there is a tie (Plurality Algorithm)
2. **Shuffler**: The system should be able to shuffle a list at beginning and can be turned off. (Droop Quota Algorithm)
3. **Shuffler Terminator**: The system should be able to turn off the shuffle option.
4. **Determine Algorithm**: The system should accept user input, and determine running algorithm.
5. **~~Generate Lists~~**~~: The system put the losers into the non-elected list according to ranking and status determined sequence when using (Droop Quota algorithm).~~

5. **Generate Candidate Votes Count LIst**: The system generates a list to count and store the votes of each candidates.

1. **Voting Status Detection:** The system should automatically stop when all winners are chosen.
2. **~~Maintain List~~**~~: The system should be able to maintain and return the list of winners.~~
3. **Validate Input Parameter**: The system should be able to validate input paratermers (no\_candidate >= no\_spots, 4 input parameters.. ) and handle exceptions.
4. **Candidate Filter:** The system will filter out votes for candidates who are already in elected or non-elected lists (For Droop Quota algorithm).
5. **Ballot Invalidation**:The system should remove ballots whose votes contribute to an election when a candidate is elected. (For DQ Algorithm).
6. **Votes Redistribution**: The system should redistribute the votes when a candidate is put into non-elected list.
7. **Elected List Fill-in**: When the elected list is not full and there are no available votes for the remaining candidates to reach the Droop Quota, the system keeps putting candidates in the
8. **Voting System Testing**: The system should be able to run each algorithm correctly with a user-defined test file and possible expected results.
9. **Run Voting System**: the main method to run the voting system.

**terminologies:**

LCL Voting System

Plurality Algorithm

Droop Quota Algorithm

The elected list (赢家)

The non-elected list （输家）

Winner

Loser

**Ballot list**

**USER CASE 示例[注意大小写，缩写，缩进，list的表达方式等]**

注意MAIN COURSE, ALTERNATIVE COURSE和EXCEPTION的格式

3.1.1 Tie Solver



|  |  |
| --- | --- |
| **Use Case Name** | Tie Solver |
| **ID** | LCL\_01 |
| **Description** | The system should be able to randomly generate a number from a given range generate a winner if there is a tie when using plurality algorithm. |
| **Trigger** | There is a tie when the user is using plurality algorithm implementation of the voting system. |
| **Actors** | User |
| **Precondition** | 1. The plurality algorithm is chosen by user and executed. |
| **Main Course** | 1. The plurality algorithm is executed. 2. The system gets the number (N) of candidates that are in the tie condition. 3. The system randomly assigns an unique number in the range of 0 to N-1 to each candidate that is in tie. 4. The system randomly generates a number (N’) in the range of 0 to N-1. 5. The candidate assigned to the same number as N’ will be picked as the winner of the tie condition. 6. The system enters the next iteration to pick the next the winner. |
| **Alternative Courses** | AC1. The plurality algorithm is not executed as expected.   1. The system generates a error message to alert the user. 2. The system tries to re-run the program.   AC2. The number of candidates is less than 2   1. The system will return the candidate |
| **Postconditions** | The tie condition is broken, and the system continues the run the rest of the program. |
| **Exception** | EX1: The system fails to generate a valid random number   1. The system will try to re-run the program. 2. If fails again, the system will prompt an alert letting the user know the failure. 3. The program will be terminated. |

3.1.2 Shuffer



|  |  |
| --- | --- |
| **Use Case Name** | Shuffler |
| **ID** | LCL\_02 |
| **Trigger** | When Droop Quota Algorithm is chosen and executed by user. |
| **Actors** | User |
| **Precondition** | 1. The Droop Quota algorithm is chosen and executed |
| **Main Course** | 1. The Droop Quota algorithm is executed. 2. The system reads the number (N) of ballots. 3. The system reads the voting information and store the data into an array of size N. 4. The system randomly shuffles the array entries. 5. The shuffled array is returned. |
| **Alternative Courses** | AC1. The Droop Quota algorithm is not executed as expected.   1. The shuffler will never be executed. |
| **Postconditions** | The input voting information is randomly shuffled. The fairness of the voting system is increased. |
| **Exception** | EX1: The system fails to read input voting information.   1. The system tries to re-load the information, and run the program again. 2. If fails again, the system will prompt an alert letting the user know the failure. 3. The system will be terminated.   EX2: The shuffled array is the same as the input.   1. The system will re-shuffle the array. 2. The re-shuffled array will be returned. |

3.1.3 Shuffler Terminator



|  |  |
| --- | --- |
| **Use Case Name** | Shuffler Terminator |
| **ID** | LCL\_03 |
| **Trigger** | User enters <No> regarding system question <Would you like to shuffle the voting date?>. |
| **Actors** | User |
| **Precondition** | 1. The Droop Quota algorithm is chosen by user and executed. |
| **Main Course** | 1. The system detects <No> regarding system question <Would you like to shuffle the voting data?> 2. The system skips the shuffle function and continues to execute the rest of the program. |
| **Alternative Courses** | AC1. The user enters invalid answers (any words besides <Yes> or <No>).   1. The system detects the invalid input answer. 2. The system prompts an alert to user and asks the user to enter again. |
| **Postconditions** | The input voting data will not be shuffled before processing the data. |
| **Exception** | None? |

3.1.4 Determine Algorithm



|  |  |
| --- | --- |
| **Use Case Name** | Determine Algorithm |
| **ID** | LCL\_04 |
| **Trigger** | The initiation of the program |
| **Actors** | User |
| **Precondition** | 1. The main program is successfully initiated. |
| **Main Course** | 1. The system reads the user input algorithm parameters of the main program. 2. The system executes either plurality algorithm or Droop Quota algorithm based on user the input. |
| **Alternative Courses** | AC1. Invalid number of user input parameters is detected.   1. The system detects the invalid number of input parameter. 2. The main program will be terminated. |
| **Postconditions** | The corresponding algorithm is executed and the voting date is properly processed. |
| **Exception** | EX1: Invalid user input algorithm parameter is detected.   1. The system detects the invalid input parameter. 2. The system prompts an alert to user and asks the user to enter again. |

3.1.9 Candidate Filter



|  |  |
| --- | --- |
| **Use Case Name** | Candidate Filter |
| **ID** | LCL\_09 |
| **Description** | The system will skip votes for candidates who are already in elected or non-elected lists, and will continue collecting votes from the next priority when using Droop Quota algorithm. |
| **Trigger** | The ballot’s voted candidate at current priority is already in the elected or non-elected lists. |
| **Actors** | User |
| **Precondition** | The Droop Quota algorithm is chosen by user and executed. |
| **Main Course** | 1. The system reads the user input algorithm parameters of the main program. 2. The system executes either plurality algorithm or Droop Quota algorithm based on user the input. |
| **Alternative Courses** | AC1. Invalid number of user input parameters is detected.   1. The system detects the invalid number of input parameter. 2. The main program will be terminated. |
| **Postconditions** | The system collect votes for the next available candidate who is in neither elected list nor non-elected list. |
| **Exception** | EX1: The input file has invalid priority number/order   1. The system stops and go back to the initial state before running the voting process. 2. The system prompts an alert and notify the user that the input file contains invalid information at a specific location. |

3.1.10 Ballot Invalidation



|  |  |
| --- | --- |
| **Use Case Name** | Ballot Invalidation |
| **ID** | LCL\_10 |
| **Description** | The system should remove ballots whose votes contribute to an election when a candidate is elected when using Droop Quota algorithm. |
| **Trigger** | A new candidate’s vote reaches the Droop Quota and is to be put into the elected list. |
| **Actors** | User |
| **Precondition** | 1. The Droop Quota algorithm is chosen by user and executed. 2. The elected list is not full. |
| **Main Course** | 1. The system tries to collect the vote for a specific candidate who is not in the elected list or non-elected list. 2. The candidate’s vote reaches the Droop Quota. 3. The system adds the candidate to the elected list. 4. The system remove the ballots who contribute votes to the election of the candidate from the ballot list. |
| **Alternative Courses** | AC1. The elected list becomes full after the removing process in step 4 in Main Course.   1. The system stops the voting process and return the elected list.   AC2. The ballot list becomes empty while the elected list is not full after the removing process in step 4 in Main Course.   1. The system goes to 3.1.12 Elected List Fill-in Main Course step 1. |
| **Postconditions** | The system successfully removes ballots that become invalid in ballot list and continues the voting process. |
| **Exception** | EX1: The ballot list becomes empty   1. system prompts an alert to notify the user that the input file is not valid to generate proper result and suggests the user to test the input file. |

3.1.11 Votes Redistribution



|  |  |
| --- | --- |
| **Use Case Name** | Votes Redistribution |
| **ID** | LCL\_11 |
| **Description** | The system should redistribute the ballots to other candidates that vote for a candidate when a candidate is put into non-elected list. |
| **Trigger** | The system runs out of ballots and no candidate reaches the Droop Quota. |
| **Actors** | User |
| **Precondition** | 1. The Droop Quota algorithm is chosen by user and executed. 2. The elected list is not full. 3. No candidate reaches the Droop Quota. |
| **Main Course** | 1. After collecting a vote, the system has used all the ballots from the ballot list while no candidate has reached the Droop Quota. 2. The system count each candidate’s vote and selects the candidate with minimum votes. 3. The system puts the selected candidate to non-elected list and takes all ballots that vote for this candidate back to the ballot list. 4. The system runs 3.1.09 Candidate Filter Main Course step 1 and redistributes all ballots in the ballot list. 5. If the candidate reaches the Droop Quota, the system goes to 3.1.10 Ballot Invalidation Main Course step 1. |
| **Alternative Courses** | AC1. The ballot list becomes empty before any candidate reaches the Droop Quota.   1. The system goes to Main Course step 1.   AC2. There are no remaining candidates while the elected list is not full.   1. The system goes to 3.1.12 Elected List Fill-in Main Course step 1 |
| **Postconditions** | The system successfully redistributes ballots that become valid again in the ballot list and continues the voting process. |
| **Exception** | EX1: The system stops incorrectly   1. The system prompts an alert to notify the user about the systematic error and stop the voting process. 2. The system goes back to the initial state before running the voting system. |

3.1.12 Elected List Fill-in



|  |  |
| --- | --- |
| **Use Case Name** | Elected List Fill-in |
| **ID** | LCL\_12 |
| **Description** | When the elected list is not full and there are no available votes for the remaining candidates to reach the Droop Quota, the system keeps putting candidates in the non-elected to elected list. |
| **Trigger** | the elected list is not full and there are no available votes for the remaining candidates to reach the Droop Quota |
| **Actors** | User |
| **Precondition** | 1. The elected list is not full and there are no available votes for the remaining candidates to reach the Droop Quota. 2. The non-elected list contains greater or equal to the number of candidates that need to be filled in the elected list. |
| **Main Course** | 1. The system pops out the candidate that is last added to non-elected list. 2. The system tests if the elected list is full. 3. If the elected list is not full, the system offers the candidate to the end of the elected list. 4. The system goes back to step 1. |
| **Alternative Courses** | AC1. If the elected list is full   1. the system stops the voting process and return the elected list. |
| **Postconditions** | The system successfully takes candidates from the non-elected list to fill in the elected list. |
| **Exception** | EX1: There are not enough candidates in the non-elected list to fill in the empty seats in the elected list.   1. The system prompts an alert to notify the user about this situation and suggests the user to test the input file. |

3.1.13 Voting System Testing



|  |  |
| --- | --- |
| **Use Case Name** | Voting System Testing |
| **ID** | LCL\_13 |
| **Description** | The system should be able to run each algorithm correctly with a user-defined test file and possible expected results. |
| **Trigger** | The user clicks the <test> button for a given input file |
| **Actors** | User |
| **Precondition** | 1. There is an input file to run the test. 2. The input file is given in the format of either algorithm to use. 3. The user has the option to enter a list of elected list as possible expected outcome for testing purpose. |
| **Main Course** | 1. The user inputs expected outcome. 2. The system goes to 3.1.14 Run the Voting System Main Course step 1 to run the voting system. 3. The system prompts the return elected list and the expected outcome for testing. |
| **Alternative Courses** | AC1. The user does not input expected outcome in Main Course step 3:   1. The system skips Main Course step 3 and goes directly to step 4. 2. The system only prompts the return elected list and does not prompt expected outcome.   AC2. There voting system stops because of other exceptions.   1. The system prompts the alert message according to the specific exceptions of the specific functional requirement. |
| **Postconditions** | The system successfully run the test file and output the elected list to compare with the user-inputed expected result. |
| **Exception** | EX1: The system stops incorrectly   1. The system prompts an alert to notify the user about the systematic error and stop the voting process. 2. The system goes back to the initial state before running the voting system. |

3.1.14 Run the Voting System



|  |  |
| --- | --- |
| **Use Case Name** | Run the Voting System |
| **ID** | LCL\_14 |
| **Description** | The main method to run the voting system. |
| **Trigger** | The user clicks the <run> button for a given input file |
| **Actors** | User |
| **Precondition** | 1. There is an input file to run the test. 2. The input file is given in the format of either algorithm to use. |
| **Main Course** | 1. The user loads the input file. 2. The user selects which algorithm to use for testing. 3. The system goes to 3.1.6 Voting Status Detection Main Course step 1 to determine if the voting ends. 4. The system takes the first ballot from the ballot list. 5. The system goes to 3.1.9 Candidate Filter Main Course step 1 to filter out invalid votes. 6. The system goes to 3.1.5 Generate Candidate Votes Count List Main Course step 1. 7. The system goes back to Main Course step 3. |
| **Alternative Courses** | AC1. The user clicks the <stop voting process> button to stop the voting system.   1. The system stops the voting system and goes back to its initial state before running the voting process. |
| **Postconditions** | The system successfully run the test file and output the elected list. |
| **Exception** | EX1: The system stops incorrectly   1. The system prompts an alert to notify the user about the systematic error and stop the voting process. 2. The system goes back to the initial state before running the voting system.   EX2: No input file or input file with wrong format   1. The system prompts an alert to notify the user about the error of the input file. 2. The system stops the voting system. |

3.2 Non-Functional Requirement

3.3 Security

陈：Introduction, 9, 10 , 11

Li: 5, 6, 7, 8