Use Cases

for

Project 1: Waterfall Methodology

Version 1.0 approved

Prepared by Team 7: Marcus Rana (rana0066), Justin Lau (lau00054), Liam Mcguigan (mcgui479), Lucas Olsen (olse0280)

CSCI 5801 Team 7

February 2023

Table of Contents

Use Case 1: Starting the System

Name	Staring the System
ID	UC_1
Description	The voting system is used to determine the results of an election. The first step of this is to start the voting system.
Actors	Programmers, testers, and election officials
Organizational Benefits	The system increases the efficiency (100,000 ballots can be counted in under 4 minutes) and accuracy of vote counting.
Frequency of Use	Each time an actor wants to run the voting system.
Triggers	An actor wants to run the voting system to count votes in an election.
Preconditions	The user is in the directory of the system and they know the command that needs to be entered to be able to run the system.
Postconditions	The system has begun and is ready for a ballot file to be inputted.
Main Course	1. The user logs into a CSE lab machine 2. The user opens the terminal 3. The user navigates to the directory where the voting system is stored 4. The user enters the command to start the system
Alternate Course	AC1: Command line argument ballot file 1. The actor follows the Main Course Steps 1, 2, and 3 2. The user enters the command to start the system with the ballot file name as an argument AC2: The user is not near a CSE lab machine 1. The user is not close to a CSE lab machine 2. The user navigates to *https://cse.umn.edu/cseit/self-help-guides/virtual-online-linux-environment-vole* and clicks “Connect to VOLE” on the right-hand side 3. Return the user to step 2 of the main course
Exceptions	EX1: The user does not know where the voting system is located in the directory 1. The user asks a colleague about the location of the voting system EX2: The user does not know the command to start the system 1. The user asks a colleague about the command to start the voting system

Use Case 2: Ballot File Input & Identification

Name	Ballot File (CSV) Input & Identification
ID	UC_2
Description	After the system has started, the user will be prompted to enter a valid ballot file
Actors	Programmers, testers, and election officials
Organizational Benefits	The system can count the votes efficiently and accurately with a condensed ballot file.
Frequency of Use	Each time an actor wants to count the number of votes in an election.
Triggers	The system has been started by the user
Preconditions	The system must be run with an existing, properly formatted CSV file with read permissions. The ballot file should not be able to be edited in any way.
Postconditions	The system has the ballot file and is now able to read the file
Main Course	1. The actor identifies the name of the ballot file 2. The actor starts the system (UC_1) 3. The user is prompted to enter the location and filename of the ballot file 4. The system finds and opens the file
Alternate Course	AC1: Command line input 1. The user has started the program with the location and name of the ballot file as a command line argument
Exceptions	EX1: Nothing is inputted when prompted (no file is provided) 1. A message is displayed to notify the user that a file needs to be provided 2. Return the user to step 3 of the main course 3. After 3 prompts to enter a filename and the user does not enter anything, the system is reset EX2: File is not found (incorrect name/type/etc.) 1. The inputted file is not valid (incorrect filename, incorrect file type, incorrect file location, permissions, etc.) 2. A message is displayed to the user to double-check the file location, filename, and file type as the inputted filename was not found 3. Return the user to step 3 of the main course 4. After 3 prompts to enter a filename and the user does not enter a valid filename, the system is reset

Use Case 3: Determining the Election Type

Name	Determine the Election Type (IR or CPL)
ID	UC_3
Description	The type of election is determined by the first line of the provided ballot file, so the system determines the election type by reading the first line of the ballot file.
Actors	Programmers, testers, and election officials
Organizational Benefits	It improves efficiency as the actor does not need to input the type of election manually.
Frequency of Use	Each time an actor wants to count the number of votes in an election.
Triggers	The system is started by a user and a ballot file has been inputted. It is then found and opened by the system.
Preconditions	The inputted ballot file is found and opened, and the file is correctly formatted with read permissions.
Postconditions	The type of election is determined by the system, either IR or CPL.
Main Course	1. The system has been started by the user (UC_1) 2. The user enters the ballot file and it is found and opened by the system (UC_2) 3. The first line of the ballot file is read 4. The election type is determined and stored by the system
Alternate Course	AC1: The ballot file's first line does not have the election type 1. The system has opened the ballot file but the election-type information is missing 2. The system prompts the user for the missing information 3. The user enters either "IR" or "CPL" depending on the type of election
Exceptions	EX1: No valid ballot file 1. The user starts the system but does not provide a valid ballot file 2. The user is again prompted for a ballot file

Use Case 4: Creating the Audit File

Name	Creating the Audit File
ID	UC_4
Description	An audit file is automatically created when ballot data is processed so that the election process remains transparent. The audit file is a text file, and it should not be able to be edited by anyone other than the voting system.
Actors	Programmers, testers, and election officials
Organizational Benefits	Provides a file for documentation on how the election calculations were determined to provide accuracy and transparency.
Frequency of Use	An audit file is created for each election. They are used when an election is audited to verify the results.
Triggers	Once the user starts the system and has inputted a valid ballot file
Preconditions	The system has been provided a valid ballot file and the directory has write permissions to create and write to the audit file.
Postconditions	The audit file is ready to be named
Main Course	1. The user has started the system (UC_1) and entered a valid ballot file (UC_2) 2. The system goes to the directory where the Voting System is located 3. The system sees there is a folder named “AuditFiles” 4. The audit file is named (UC_5) 5. The audit file is created in the “AuditFiles” folder with write permissions
Alternate Course	AC1: The “AuditFiles” folder has not been created 1. The user has started the system (UC_1) and entered a valid ballot file (UC_2) 2. The system goes to the directory where the Voting System is located 3. The system sees there is no folder named “AuditFiles” 4. The system creates the folder, “AuditFiles” 5. The audit file is named (UC_5) 6. The audit file is created in the “AuditFiles” folder with write permissions
Exceptions	EX1: The device does not have enough storage to make another file 1. The system recognizes the device does not have enough storage 2. An alert is sent to the user to free up memory before running the system again EX2: The system cannot create a new folder if the “AuditFiles” folder does not exist 1. The system recognizes there is an error in creating the new folder 2. An alert is sent to the user that there is an error in creating a new folder to hold audit files

Use Case 5: Naming the Audit File

Name	Naming the Audit File
ID	UC_5
Description	An audit file is automatically created when ballot data is processed so that the election process remains transparent. The audit file is a text file, and it should not be able to be edited by anyone other than the voting system. There is a certain naming convention the system uses when creating audit files.
Actors	Programmers, testers, election officials
Organizational Benefits	The audit file provides documentation on how the election calculations were determined and allows for easy verification of the election process. The naming convention allows actors to easily find the desired audit file.
Frequency of Use	An audit file is created and named for each election. The files are used when an election is audited to verify the results.
Triggers	After the program is started by the user, the audit file is generated and named.
Preconditions	The system has started, it has been provided with a valid ballot file, and it is in the "AuditFiles" folder.
Postconditions	The audit file is named and can be created
Main Course	1. The user has started the system (UC_1), and it has been provided with a valid ballot file (UC_2) 2. The type of election has been determined as IR (UC_3) 3. The audit file is in process of being created (UC_4) 4. The system starts the audit file name with "IR" 5. It then adds an underscore (_) 6. The date of processing then follows the underscore a. The date will be in the format: mmddyyyy 7. An example of an audit file name: IR_02152023 8. The system moves to step 6 of the Main Course in UC_4
Alternate Course	AC1: The type of election has been determined as CPL (UC_3) 1. The user has started the system (UC_1), and it has been provided with a valid ballot file (UC_2) 2. The type of election has been determined as CPL (UC_3) 3. The audit file is in process of being created (UC_4) 4. The system starts the audit file name with "CPL" 5. It then adds an underscore (_) 6. The date of processing then follows the underscore a. The date will be in the format: mmddyyyy 7. An example of an audit file name: CPL_02152023 8. The system moves to step 6 of the Main Course in UC_4
Exceptions	EX1: The audit file does not have write permissions 1. The system recognizes the file does not have write permissions

	2. A new audit file is created with write permissions (UC_4) EX2: The filename already exists 1. The system recognizes the file already exists in the "AuditFiles" folder 2. The system cancels the creation of the audit file 3. It follows either the main course steps 2-6 (IR) or AC1 steps 2-6 (CPL) depending on the type of election 4. After the type of election, an underscore, and the date are added to the filename, and an additional underscore with a number is added at the end of the filename a. The number will begin at 2 b. Example: IR_02152023_2 5. If the filename recognizes this file already exists too (with an additional underscore and number), the number at the end of the filename is incremented by 1 a. Example: IR_02152023_3

Use Case 6: Ballot File Processing

Name	Ballot File Processing
ID	UC_6
Description	After the system has identified the type of election and created the audit file, it can read the rest of the ballot file.
Actors	Programmers, testers, and election officials
Organizational Benefits	The system increases efficiency (100,000 ballots can be counted in under 4 minutes) and accuracy in vote counting.
Frequency of Use	Each time an actor wants to count the number of votes in an election
Triggers	The system must be provided with an existing, properly formatted CSV file with read permissions
Preconditions	The system has read the first line of the ballot file to find the election type.
Postconditions	The system has read the rest of the ballot file and has stored the necessary information. The stored information can then be used to determine a winner.
Main Course	1. The system has been started by the user (UC_1) and it has been provided with a valid ballot file that is identified and opened (UC_2) 2. The election type is determined to be IR (UC_3) 3. The system reads the 2nd line and stores it as the number of candidates 4. The system reads the 3rd line and stores it as the candidates (separated by commas) 5. The system reads the 4th line and stores it as the number of ballots in the ballot file 6. The system reads the rest of the ballot file and stores the ballot information for the election results and the audit file 7. The file is closed
Alternate Course	AC1: CPL Election type 1. The system has been started by the user (UC_1) and it has been provided with a valid ballot file that is identified and opened (UC_2) 2. The election type is determined to be CPL (UC_3) 3. The system reads the 2nd line and stores it as the number of parties 4. The system reads the 3rd line and stores it as the parties (separated by commas) 5. The system reads the next n (n = the number of parties as found by line 2 in the file) lines and stores them as the candidates in ranked order for each party 6. After the n lines are read, the next line is read and stored as the number of seats 7. The next line is read and stored as the number of ballots 8. The system reads the rest of the ballot file and stores the ballot information for the election results and the audit file 9. The file is closed

	AC2: Not all information is provided in the header of an IR ballot file 1. The system expects a certain format for an IR ballot file 2. The system recognizes it is missing information in the header of the provided ballot file 3. The user is prompted to enter the missing information 4. The information from the user is parsed by the system and stored 5. The file is closed AC3: Not all information is provided in the header of a CPL ballot file 1. The system expects a certain format for a CPL ballot file 2. The system recognizes it is missing information in the header of the provided ballot file 3. The user is prompted to enter the missing information 4. The information from the user is parsed by the system and stored 5. The file is closed
Exceptions	EX1: The system is unable to determine what information is missing 1. The system recognizes there is missing information in the header of the provided ballot file 2. The system cannot determine what information is missing 3. The system sends an alert to the user that an error has occurred because there is missing information in the ballot file and the system cannot determine which information is missing 4. The system resets EX2: The system is unable to determine what information has been inputted 1. The system recognizes there is missing information in the header of the provided ballot file 2. The user is prompted to enter the missing information 3. The system is unable to determine how the inputted information should be stored 4. The system sends an alert to the user that an error has occurred because the inputted information could not be stored properly 5. The system resets

Use Case 7: Calculating a Winner in an Instant Runoff Election

Name	Calculating a Winner in an Instant Runoff Election
ID	UC_7
Description	The winner of an IR election is chosen as the candidate who receives more than 50% of the vote after rounds of candidate elimination.
Actors	Programmers, testers, election officials
Organizational Benefits	A winner of an IR election can be determined based on input data
Frequency of Use	Whenever the votes for an IR election need to be counted, either for the original election results or for recount purposes.
Triggers	Data for an IR election has been processed by the system and a winner needs to be calculated for the system to continue.
Preconditions	In an IR election, all votes have been cast and the corresponding ballot data has been processed by the system as described in UC_6.
Postconditions	The winner of the processed IR election shall be determined.
Main Course	1. Ballot data has been processed by the system 2. The system scans candidate voter percentages and finds a candidate with over 50% of the votes 3. The candidate with more than 50% of the votes is declared the winner and the system moves on to UC_10
Alternate Course	AC1: No remaining candidate has >50% of the votes 1. The candidate with the lowest percentage of votes is determined and is eliminated from election contention. 2. Votes previously counting towards the eliminated candidate are changed to now count towards the voter's next highest choice among the remaining candidates (if a voter ranks the following contending candidates as: Alice=1st, Bob=2nd, and Charlie=3rd then Alice is eliminated, this voter's vote now counts towards Bob). 3. If a candidate breaks the 50% vote threshold they are declared the winner, otherwise, AC1 is repeated until only 2 candidates remain
Exceptions	EX1: Tie occurs in candidate elimination 1. No candidate has reached the >50% vote threshold and of the lowest remaining candidates, each has an equal vote percentage. 2. Proceed to UC_9 (AC1) to determine the candidate to eliminate EX2: Tie occurs in candidate election 1. Of the two remaining candidates, each has exactly 50% of the vote. 2. Proceed to UC_9 (Main course) to determine the winner of the election.

Use Case 8: Calculating the Winner(s) in a Closed Party List Election

Name	Calculating the Winner(s) in a Closed Party List Election
ID	UC_8
Description	Seats in CPL elections are allocated to parties that receive vote totals above the calculated 'seat quota' (1 party can win multiple seats), then assigned to candidates within a party based on their order of appearance on the original ballots.
Actors	Programmers, testers, and election officials
Organizational Benefits	Winners in a CPL election can be determined from input data
Frequency of Use	Whenever the winners of a CPL election need to be calculated, either by original election results or by a recount.
Triggers	Data for a CPL election has been processed by the system and a winner needs to be calculated for the system to proceed.
Preconditions	In a CPL election, all votes have been cast and the corresponding ballot data has been processed by the system as described in UC_6
Postconditions	All available seats in the election have been allocated to valid election candidates
Main Course	1. Ballot data has been processed by the system 2. A `seat quota` is calculated as [total number of votes / total candidates]. 3. For each party, calculate the floor of [party's vote total / seat quota] to determine how many seats to allocate to a party. Store the remainder of this calculation for later evaluation. (Example: Party L has 23 votes in an election for 10 seats with 100 total votes. Quota = 100/10 = 10, Party L seats = floor(23/10) = 2 and a remainder of 3) 4. Each party's remainders (calculated in step 3) are compared and the parties with the largest vote remainders are allocated the remaining seats. 5. Within each party, allocated seats will be given to party candidates by order of appearance on voting ballots and the system moves onto UC_10.
Alternate Course	N/A
Exceptions	EX1: Remainder-allocated seats encounter parties tied for highest remainder 1. A seat to be allocated by remainder finds multiple parties tied for the highest remainder. 2. Proceed to UC_9 (AC2) to determine the seat's allocation EX2: A party is allocated more seats than candidates within the party 1. A party has received enough votes to obtain more seats than candidates within the party. 2. Proceed to UC_9 (AC3) to determine the seat's allocation

Use Case 9: Calculating a Winner in a Tie

Name	Calculate a Winner in a Tie
ID	UC_9
Description	If a tie happens between multiple parties, the system shall decide a winner fairly to break the tie.
Actors	Programmers, Testers, Election Officials
Organizational Benefits	Ties in election results will be broken fairly and efficiently
Frequency of Use	During any election whenever a tie occurs
Triggers	1. In IR, the remaining two candidates each receive 50% of the votes 2. In IR, no candidate receives >50% of the vote and candidates with the lowest vote totals receive the same amount of votes 3. In CPL, allocating seats by remainder encounters multiple parties tied with the same highest remainder. 4. In CPL, a party is allocated more seats than candidates within the party
Preconditions	The remaining two candidates in an IR election each receive 50% of the votes
Postconditions	A winner has been selected by the system for the IR election
Main Course	1. Two candidates tie for 50% of the vote in an IR election 2. The system 'flips a coin' to determine a winner fairly and efficiently. 3. The system proceeds with the results of the tie break
Alternate Course	AC1: In IR, multiple candidates tie for the lowest vote total during elimination 1. Two or more candidates hold the same lowest vote totals when the system attempts to eliminate a candidate from the election 2. Of these candidates facing elimination, the system will use a fair and efficient algorithm to determine which candidate to eliminate 3. The system proceeds to UC_7 (AC1) AC2: In CPL, multiple parties tie for having the highest vote remainder when allocating a seat by remainder 1. Two or more parties hold the same, largest vote remainders 2. The system uses a fair and efficient algorithm to determine which one of the tied parties will receive the seat 3. The selected party wins the seat, and the system proceeds to UC_8 (MC5) AC3: In CPL, a party is allocated more seats than candidates 1. A party has been allocated more seats than candidates within the party 2. The seats not used by the winning party's candidates will be allocated to other parties via a lottery, i.e. the system will determine a different party in contention to receive the extra seats through a fair and efficient algorithm 3. The system continues allocating seats based on UC_8
Exceptions	EX1: The algorithm used by the system is found to hold bias 1. The system detects bias in its tie-breaking algorithm 2. The system alerts the users to this fault and halts execution in UC_11

Use Case 10: Display the Winner(s) to the Screen

Name	Display the Winner(s) to the Screen
ID	UC_10
Description	After processing the results of the election, the program shall display the results of the election, including the winner(s), individual vote tallies, and vote percentages to the terminal for viewing
Actors	Programmers, testers, and election officials
Organizational Benefits	The results of an election will be readily available in an easily understandable display to the operator of the program. Election officials will see the results of the election shortly after processing.
Frequency of Use	Anytime an election occurs or the program is run with valid data.
Triggers	1. A winner has been determined by the system in an IR election 2. Seat allocation and winners have been determined by the system in a CPL election
Preconditions	In an IR election, a winner has been determined through UC_7
Postconditions	The results of this IR election will be printed to the terminal along with critical vote data.
Main Course	1. The results of the IR election have been determined (UC_7) 2. Information such as the winner of the election and a list of candidates with their vote statistics are printed on the screen for actors to view
Alternate Course	AC1: CPL election results 1. The results of the CPL election have been determined (UC_8) 2. Information regarding the seats received by parties, the candidates named to fill those seats, and party vote statistics will be printed on the screen for actors to view.
Exceptions	EX1: Error compiling data 1. Election results determined by UC_8 or UC_9 have somehow compiled problematic data 2. The program attempts to print this data, but fails and proceeds to UC_11

Use Case 11: System Shutdown During Execution

Name	System shutdown during execution
ID	UC_11
Description	If the system is stopped/shutdown while it is in the process of processing a ballot file, it will reset to the main menu
Actors	Programmers, testers, election officials
Organizational Benefits	Provides an option to confirm if the actor wants to stop the system, and gracefully resets the system if it is confirmed rather than crashing which provides a better user experience for actors
Frequency of Use	Each time an actor wants to reset the system while it is processing a ballot file
Triggers	An actor wants to run the voting system to count votes in an election but the system is stopped or shut-down by the actor
Preconditions	The system has been started by the user
Postconditions	The system is at the main menu and can be given a ballot file for processing.
Main Course	1. The user starts the system (UC_1) 2. The user stops the system 3. The system provides a prompt to the user confirming if they would like to stop the system 4. If confirmed, the system resets to the main menu 5. If not confirmed, the system continues where it left off
Alternate Course	AC1: The user stops the system after the election type has been determined 1. The user starts the system (UC_1) 2. The user enters a valid ballot file (UC_2) 3. The system determines the election type (UC_3) 4. The user stops the system 5. The system provides a prompt to the user confirming if they would like to stop the system 6. If confirmed, the system resets to the main menu 7. If not confirmed, the system continues where it left off AC2: User stops the system after the audit file has been created 1. The user starts the system (UC_1) 2. The user enters a valid ballot file (UC_2) 3. The system determines the election type (UC_3) 4. The system creates and names the audit file (UC_4, UC_5) 5. The user stops the system 6. The system provides a prompt to the user confirming if they would like to stop the system 7. If the audit file is confirmed to be empty by the system, it is deleted 8. If confirmed, the system resets to the main menu 9. If not confirmed, the system continues where it left off AC3: The user stops the system while the system is processing the ballot file

	1. The user starts the system (UC_1) 2. The user enters a valid ballot file (UC_2) 3. The system determines the election type (UC_3) 4. The system creates and names the audit file (UC_4, UC_5) 5. The system begins processing the ballot file (UC_6) 6. The user stops the system 7. The system provides a prompt to the user confirming if they would like to stop the system 8. If confirmed, the system notes on the audit file that the process was interrupted and it is reset to the main menu 9. If not confirmed, the system continues where it left off
Exceptions	N/A