Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Team 14:

Alex Gude (gudex009)

Colin Hommerding (homme093)

Chris Walaszek (walas013)

Fan Ding (ding0322)

University of Minnesota

February 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Voting System	2/19/21	Initial version	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the Voting System. Our voting system contains 2 different voting algorithms, Instant Runoff (IR) and Open Party List (OPL). This document will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate, and how the system will react to users. This document is intended for both users and the developers of the system.

1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement Specification Documents provided by Prof.Shana Watters. (See Section 1.5 reference for links)

1.3 Intended Audience and Reading Suggestions

· Typical Users, such as poll officers who want to use this Voting System for analyzing the election results.

· Programmers who are interested in working on the project by further developing it or fixing existing bugs.

· Testers who are interested in testing the project by trying to find and report potential bugs.

1.4 Product Scope

The Voting System is a tool that people can use to calculate and analyze results for voting. This system will be designed to maximize productivity by automating the counting of ballots and generating an accurate result, which would otherwise have to be performed manually. By maximizing the users' work efficiency and production the system will meet the users' needs while remaining easy to understand and use. Also, users can use it to give results to the media and produce a detailed audit file that allows the user to review the development of the election.

1.5 References

gephi_srs_document.pdf:
https://canvas.umn.edu/courses/217913/files/18708905?wrap=1

IEEE Template for System Requirement Specification Documents.docx:

https://canvas.umn.edu/courses/217913/files/18708873?wrap=1

SRSExample-webapp.docx:
https://canvas.umn.edu/courses/217913/files/18708947?wrap=1

2. Overall Description

2.1 Product Perspective

The Voting System was developed to provide automated election tabulation for democratic nations using Instant Runoff or Open Party List voting systems without write-in candidates. Given an input file containing election and ballot information, it outputs the result of the election in the form of a detailed audit file for verification and a summary file for publishing results. It is self contained and can run independently on machines without an internet connection.

2.2 Product Functions

- Start Program - Starts the program.
- Input File - The user is prompted and able to enter the file name of the voting file.
- Read File - The inputted file is opened and general information is read from it.
- Confirm information - Information read from the file is confirmed by the user before calculation begins.
- Run Election
    - Run Instant Runoff Election - Performs a process that calculates the winner of an Instant Runoff election.
    - Run Open Party List Election - Performs a process that calculates the winner of an Open Party List election.
- Show Preliminary Results - A basic overview of the election results is shown on screen including the winner.
- Produce Media Report
    - Produce IR Media Report - A media report file is produced that includes a basic breakdown of IR election results and statistics.
    - Produce OPL Media Report - A media report file is produced that includes a basic breakdown of OPL election results and statistics.
- Produce Audit File

- Produce IR Audit File - An audit file is produced that maintains a record of how each vote was allocated in the election. It contains detailed specifics of the course of the election counting.
- Produce OPL Audit File - An audit file is produced that maintains a record of each vote in the election. It contains detailed specifics detailing the course of the election counting.

2.3 User Classes and Characteristics

User Classes listed from most frequent to least frequent.

1. Election Officials
   - Those tasked with running and verifying elections. They must be able to easily use the software and be able to verify that the election was run as intended.

2. Media Personnel
   - Those who publish election results to the public. They must be provided a summary of the election results along with relevant statistics important for analyzing the election.

3. Testers
   - Those who test the software to verify it is working properly. They must require the software to run in a predictable and verifiable manner.

2.4 Operating Environment

Operating Systems:
64-bit Ubuntu 16.04, 18.04, 20.04
64-bit Windows 10
64-bit Mac OS X

Minimum Hardware Requirements:
CPU: Intel Core i5 @ 800 Hz
RAM: 8 GB

Software components:
C++ 11 or later w/ g++ compiler

Machines:
Will function for use on all CSE Lab Machines

2.5 Design and Implementation Constraints

The Voting System will be able to calculate elections with 100,000 ballots in at most 8 minutes. System maintenance will be handled by the software authors. System will only accept CSV files as input. Example input files are given in Section 5.6.

2.6 User Documentation

A quick-start guide and system documentation will be created using Doxygen and delivered along with the voting system.

2.7 Assumptions and Dependencies

As the voting system is written in C++, the user's computer is expected to have C++ 11 or later. A suitable compiler such as g++ must also be installed. The system is standalone and does not require any third-party applications. We assume that the input file is preprocessed as according to Section 5.6. The user should be able to use the terminal.

3. External Interface Requirements

3.1 User Interfaces

The main user interface that users will be interacting with is the terminal. The voting system program is run on CSV files via the terminal. Users will provide validation of election information via the terminal. All error messages and any other messages will be displayed to the terminal.

3.2 Hardware Interfaces

The minimum hardware requirements that will be accepted are an Intel Core i5 @ 800 Hz and 8 GB of ram. These specifications are mandatory in order for the voting system to be able to process up to 100,000 ballots in under 8 minutes.

3.3 Software Interfaces

This product requires a running operating system of 64-bit Ubuntu 16.04, 18.04, 20.04, 64-bit Windows 10, or 64-bit Mac OS X. It also requires that C++ 11 or later w/ g++ compiler be installed on the system.

3.4 Communications Interfaces

The Voting System has no current need for any communication interfaces. No internet connection is required as all CSV files will be provided locally.

4. System Features

4.1 Input a File

4.1.1 Description and Priority

The system shall allow the user to input an election file by typing its name after the system prompts: "Please type input file name:". The file should be a comma separated values (CSV) file. After inputting the election file, the system will be able to read the ballot information and process further counting and calculation. The priority of this system feature is high.

4.1.2 Stimulus/Response Sequences

1. The system prompts: "Please type input file name"
2. The user types the input file name and presses enter.
3. The system tries to find the input file in the local repository.
4. The system opens the input file for reading.

4.1.3 Functional Requirements

REQ-1: The system shall receive the input file name from the user via a prompt displayed on the terminal.

REQ-2: Upon failing to find a given input file, the system will prompt the user to try again.

4.2 Run OPL Election

4.2.1 Description and Priority

Running an Open Party List election is a system feature of high priority. The system is able to run this feature on any properly formatted file (see section 5.6.1). Each vote is counted towards both an individual candidate and their party. Available seats are then divided amongst the parties based on their number of votes, and assigned to the candidates in those parties who received the most votes. Once none of the parties have enough votes to get another seat, seats are assigned sequentially to those parties which have the highest remaining votes until no seats are left (see UC_006).

4.2.2 Stimulus/Response Sequences

1. The feature is started by a user prompt that confirms the basic information of the election, i.e. the number of candidates and type of election. By typing "yes", the user acknowledges these stipulations and the program will run.
2. The feature runs and tabulates the winner of the election using the OPL election style.
3. The preliminary results of the election are displayed to the screen, including the winner of the election, and the program terminates.

4.2.3 Functional Requirements

REQ-1: The system is able to read basic election information from the header of the input file and confirm that information with the user via a terminal prompt.
REQ-2: Any tie in votes is decided by a fair coin flip.
REQ-3: All independent candidates are grouped into one party.
REQ-4: Display preliminary results including the winner of the election and basic election statistics to the screen upon completion.
REQ-5: Seats are divided fairly to parties based on number of party votes, and assigned to candidates in those parties based on their number of votes. Remaining seats are given to parties with the highest remaining votes.

4.3 Run IR Election

4.3.1 Description and Priority

Running an Instant Runoff election is a feature of high priority. The system is able to run this feature on any properly formatted file (see section 5.6.2). Votes for each candidate are counted based on a ranked system. If no candidate has a majority of the votes, the candidate with the lowest number is eliminated and the votes that had been for them are redistributed based on the rank of other candidates on each respective ballot. The process then repeats itself as seen in UC_005.

4.3.2 Stimulus/Response Sequences

1. The feature is started by a user prompt that confirms the basic information of the election, i.e. the number of candidates and type of election. By typing "yes", the user acknowledges these stipulations and the program will run.
2. The feature runs and tabulates the winner of the election using the IR election style.
3. The preliminary results of the election are displayed to the screen, including the winner of the election.

4.3.3 Functional Requirements

REQ-1: The system is able to read basic election information from the header of the input file and confirm that information with the user via a terminal prompt.
REQ-2: Any tie in votes is decided by a fair coin flip.
REQ-3: If a ballot has run out of eligible ranked candidates, their vote will be disqualified and not redistributed.

REQ-4: If there is no majority when only two candidates are left, the candidate with more votes is declared the winner.
REQ-5: Display preliminary results including the winner of the election and basic election statistics to the screen upon completion.

4.4 Produce Election Audit File

4.4.1 Description and Priority
The system will automatically produce an audit file containing detailed information about how the IR or OPL election progressed every time the program is run, allowing for the election to be tracked and verified as accurate by election officials. Due to the importance of election verification, this feature is of high priority.

4.4.2 Stimulus/Response Sequences
1. The voting system completes its calculation of the election.
2. The system compiles detailed information about the election into an audit file.
3. The system stores the generated audit file to the computer's file system, in the same directory in which the voting system's executable file is located.

4.4.3 Functional Requirements
REQ-1: An audit file with the name format 'election_audit_mm-dd-yyyy_hhmmss.txt' shall be produced that contains the information specified in Table 4.1. The audit file shall be human-readable and detailed enough for election officials to recreate and verify the election for accuracy and completeness (see UC_010 and UC_011). Example audit files are given in Appendix C.1 and C2

IR Audit File	OPL Audit File	Both Audit Files
Indication that the election was run via Instant Runoff	Indication that the election was run via Open Party List	Date and time when the election was conducted
Each ballot's ranking of the candidates	The number of open seats to be filled	The number of candidates and parties running for office.
The number of votes each candidate got in each round	The distribution of votes each candidate and party gets	Each candidate's name and party
The number of votes needed for a candidate to win by majority	The quota used for the first allocation of seats	The number of ballots cast
Which ballots voted for which candidate in each round	The number of seats each party has after first allocation	Indicate when ties were broken by random selection
Which candidate was eliminated after each round, if applicable	The number of remaining seats after first allocation	
Which candidate each of an eliminated candidate's ballots are transferred to	The remaining number of votes for each party after first allocation	

When a ballot is discarded.	The party order used to allocate remaining seats.	
The winner of the election.	The number of seats each party gets during remainder allocation.	
	The ranking of each candidate within their party.	
	The total number of seats awarded to each party, and which candidates are awarded seats.	
	Any outstanding empty seats, and whether another election must be called.	

Table 4.1: The information that shall be included in audit files. IR audit files shall include all information described in columns 1 and 3, and OPL audit files shall include all information in columns 2 and 3.

4.5 Produce a Media File

4.5.1 Description and Priority

When it comes to relaying election results to civilians the media plays a very important role. In order for the media to be able to give results and statistics to the mass populace though, they need to be given them first. Therefore, this feature will make sure that when elections are run a media file is created containing relevant information and statistics on the election. This will include election type, winner, total seats, winning percentage. This feature is only of medium priority.

4.5.2 Stimulus/Response Sequences

1. System runs proper election algorithm
2. System reads results of election and calculates media statistics
3. System generates media report with results and statistics
4. System places media file in the current directory.

4.5.3 Functional Requirements

REQ-1: System can read results and calculate the media statistics from them. Then a media file shall be created with the name ‘election_media_mm-dd-yyyy_hhmmss.txt’ containing the results and relevant media statistics. This file will be added to the current directory. Examples are given in Appendix C.4 and C.5.
REQ-2: It is easily readable and can properly be used by the media to inform civilians of results.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Voting System runs on Ubuntu, Windows, and MacOS. Voting System requires a system with at least 800 Hz CPU and 8 GB RAM. It requires that a C++ compiler and Terminal are installed to run. It is expected that under normal circumstances, 100,000 ballots can be counted in 8 minutes. However, performance depends on the number of ballots. Larger amounts of ballots slows the performance.

5.2 Safety Requirements

To ensure that the ballot result is determined and consistent, there will be no write in candidates for this system. That means users shouldn't edit the input file. And users cannot change the file structure outside of the program since the election files will come in the predetermined format. To maintain the system, the developer team added a Test Mode that allows testers and developers to test and debug the system. The developer team is open for recommendations and suggestions, and be able to improve our program in the next release. This program is expected to be run in accordance with any local laws.

5.3 Security Requirements

Voting System does not have any security requirements and thus any type of user can use it without any additional privileges.

5.4 Software Quality Attributes

Voting System provides the users with both simple and advanced features. Due to its well designed and easy to use interface it can be used by both experts and typical users. However, users must already have a basic knowledge of voting before using it. To maintain election result accuracy as the primary consideration, user input is limited to prevent misuse of the system.

5.5 Business Rules

Elections are very important and must remain secure. Therefore a major operating principle is that only election officials are allowed to use the Voting System. This is to ensure that only certified and trustworthy individuals have access to the ballots. All use of the Voting System is subject to the rules and regulations of the nation in which the election it is being used on is taking place. This system is designed to be used for two types of elections, Open Party List, and Instant Runoff. An Open Party List election is one where parties put up a slate of candidates. Voters then vote for their preferred party and seats are awarded to candidates of said parties proportionally. An Instant Runoff election is one where voters use ranked choice to vote for candidates. They rank their first, second, and third choices. The candidate with the least amount of votes is eliminated in an iterative process and their votes reshuffled based on the ranked choices. This process continues via rounds until a candidate possesses a majority and is declared a winner.

5.6 Input File Requirements

In order for the program to be run, the input file must meet certain specifications. It must be a CSV file that is in the same directory as the voting system program. Each row of this file must be separated by a new line. Only one file may be given to the program per election, so all data should be contained within the election file. No write in candidates will be accepted. The first line of the file will designate whether the file contains an OPL election or IR election. The second line tells the number of candidates. Each candidate's party is listed after the candidate's name in the manner shown in the example input files. Each candidate's party is listed after the candidate's name in the manner shown in the example input files.The program will assume no mistakes were made on the ballots.

5.6.1 OPL Election

The third line of an OPL election file enumerates the candidates and their party inside brackets. There is a comma between the candidate name and party, as well as one between each set of brackets. The fourth line is the number of seats, followed by the fifth line which is the total number of ballots. A "1" is placed in exactly one position relative to commas that designate which candidate is being voted for based on the order the candidates are listed on line 3.

The following is an example of a valid input file format:

```
OPL
6
[Pike,D],[Foster,D],[Deutsch,R],[Borg,R],[Jones,R],[Smith,I]
3
2
1,,,,,
,,,,1,
```

5.6.2 IR Election

The third line of an IR election file enumerates the candidates and their party. The name of the candidate is followed by the letter of their party in parenthesis, and followed by a comma before listing more candidates. The fourth line tells the total number of ballots in the file. Each ballot must have one of the candidates ranked as their top choice. This is done by marking a 1, the second ranked choice is marked with a 2, and so on. The numbers are placed in accordance with how the candidates are listed on line 3 and separated with commas. Only the top rank is required, so for example ,1,,, would mean a vote for the second candidate and no other rankings.

The following is an example of a valid input file format:

```
IR
```

```
4
Rosen (D), Kleinberg (R), Chou (I), Royce (L)
2
1,3,4,2
1,,2,
```

6. Use Cases

6.1 Start Program

Name	Start Program
ID	UC_001
Description	The program is run and counts the ballots using the proper election type.
Actors	Election officers, testers.
Organizational Benefits	Allows for the ballots cast in the election to be counted in an efficient manner.
Frequency of Use	Whenever an election needs to be conducted.
Triggers	A proper CSV file is selected
precondition	A properly formatted CSV File
postcondition	The program finishes counting the ballots and displays preliminary results.
Main Course	1. The user types “./voting.exe and presses enter.
Alternative Courses	None
Exceptions	EX1 The file fails to run 1. Contact IT

6.2 Input File

Name	Input file
ID	UC_002
Description	Voting system prompts and asks for the election name. Users can type the election name. Then the voting system prompts and asks for input file name, a user can type the input file name in the terminal so that the voting system can process the input file and calculate the result.
Actors	Election officers, testers.
Organizational Benefits	Increasing efficiency because users don’t need to type the ballots by hand.

Frequency of Use	Every time the program runs.
Triggers	Program runs the program
precondition	The input file should be CSV format and the program is started
postcondition	Program creates an ofstream object that is associated with the input file.
Main Course	1. System prompts for election name 2. User types election name 3. System prompts for input file name 4. User types the input file
Alternative Courses	None
Exceptions	EX1 User decided to close the program 1. Exit program. EX2 System can't find the input file. 1. Print out message to user "your input file can't be found, try again"

6.3 Read File

Name	Read file
ID	UC_003
Description	After the user types a valid file name and press enter, the system can read the input file and store the file's information.
Actors	System
Organizational Benefits	Increasing efficiency because users don't need to type the ballots by hand. System reads the input file and access ballot information.
Frequency of Use	Everytime a user inputs a valid input file name and runs the program.
Triggers	The user runs the program with a valid input file name.
precondition	The contents in the input file must have a certain format.
postcondition	Program stores ballot information into variables and data structures.
Main Course	1. System reads the input file content line by line 2. System stores information into variables.
Alternative Courses	None
Exceptions	EX1 User decided to close the program 1. Exit program. EX2 contents in the input file don't have a certain designed format. 1. Print out message to user "your input file don't have the correct format, contact IT department" 2. Exit program.

6.4 Confirm Information

Name	Confirm information.
ID	UC_004
Description	After the system reads and stores all the information from the input file, the user can confirm them by typing Yes or No in the terminal after viewing the brief info displayed by the system.
Actors	Election officers, testers.
Organizational Benefits	Increasing correctness and confidence because users can confirm the information before the algorithm runs.
Frequency of Use	Everytime user runs the program with a valid input file.
Triggers	The user inputs a valid input file and runs the program.
precondition	System has brief information on variables.
postcondition	The information we access is confirmed and already to run an algorithm to get the voting result.
Main Course	1. System prints out brief information in the terminal. 2. User types yes to confirm.
Alternative Courses	AC1 User types No when confirming the brief info 1. Print out message to user “double check the input file” 2. Exit program.
Exceptions	EX1 User decided to close the program 1. Exit program.

6.5 Run IR Election

Name	Run IR Election
ID	UC_005
Description	Counts the votes and calculates a winner using the “Instant Runoff” election style.
Actors	Election Official(s)
Organizational Benefits	Calculates the winner of the election quickly and easily.
Frequency of Use	Every time a file is input with the “Instant Runoff” election type.
Triggers	Election Official confirms that they want to calculate the result of an Instant Runoff election.
Precondition	The user has confirmed that they want to run an Instant Runoff election and polling data is available.
Postcondition	All votes have been accounted for and the system has election statistics.

Main Course	1. The following process is run to count the votes: a. Each ballot's top designated vote is assigned to its respective candidate. b. If one candidate has a majority of at least 50%, go to step 3. c. The votes of the candidate receiving the fewest votes are redistributed to their top marked eligible candidate. Return to step 1b. 2. The candidate with the majority wins the election. Go to UC_012.
Alternative Courses	AC1 A tie occurs when calculating the candidate with the fewest number of votes. 1. Go to UC_007. The selected candidate from this process is designated to have the fewest votes. 2. Go to Main Course 1c. AC2 A tie occurs when only two candidates remain. 1. Go to UC_007. The selected candidate from this process is designated as the winner of the election. AC3 No majority is reached after only two candidates remain. 1. The candidate with more votes is declared the winner. 2. If a tie has occurred, go to AC2.
Exceptions	EX1 A ballot does not have any remaining eligible designated votes. 1. The vote is not redistributed and will not be counted for any remaining calculation cycles. EX2 User interrupts the program 1. Calculation stops. No progress is saved.

6.6 Run OPL Election

Name	Run OPL Election
ID	UC_006
Description	Counts the votes and calculates a winner using the "Open Party List" election style.
Actors	Election Official(s)
Organizational Benefits	Calculates the winner of the election quickly and easily.
Frequency of Use	Every time a file is input with the "Open Party List" election type.
Triggers	Election Official confirms that they want to calculate the result of an Open Party List election.
Precondition	The user has confirmed that they want to run an Open Party List election and polling data is available.
Postcondition	All votes have been accounted for and the system has election statistics.
Main Course	1. The following process is run to count the votes: a. The "quota" is calculated by dividing the total number of votes by the number of seats available. b. Each ballot is counted, with a vote counting towards both a candidate and the candidate's party. Independents are grouped into one party.

	c. Each party then receives the number of seats equal to the quota being divided into their votes, to a whole number. Any remainder is saved for later. d. Seats are assigned within each party to the candidate with the highest number of individual votes that has not been chosen. e. Assign one seat to each party in the order of highest remainder to lowest until no seats remain. 2. Those chosen are the winners. Go to UC_012.
Alternative Courses	AC1 There is a tie between two parties' remainders or within a party. 1. Go to UC_007. The selected candidate from this process is designated as having the higher number of votes.
Exceptions	EX1 User interrupts the program. 1. Calculation stops. No progress is saved. EX2 User does not confirm they want to run an Open Party List election. 1. The system presents the user with their options.

6.7 Break Ties

Name	Break Ties
ID	UC_007
Description	When two or more candidates or parties (hereafter defined as entities) receive an equal number of votes, and a choice must be made between them, entities are chosen through random selection.
Actors	The System
Organizational Benefits	Allows ties to be broken in a fair and unbiased manner.
Frequency of Use	A tie in the system occurs with varying frequency depending on the number of ballots cast. Elections with many ballots encounter ties less often than those with few ballots.
Triggers	The system chooses to break a tie.
Precondition	There exists two or more entities that have received an equal number of votes. The relevant entities are known to the system. Only one entity among those tied must be chosen.
Postcondition	One of the tied entities is chosen.
Main Course	1. The system assigns a unique integer from 0 to (number of tied entities - 1) to each tied entity 2. The system uses an internally defined random number generator to randomly select one of the assigned integers (see EX1). 3. The system chooses the entity with which the randomly selected integer corresponds to (see EX2).
Alternative Courses	None
Exceptions	EX1 Random number generation fails 1. Return to Main Course step 1 and try again.

	EX2 Randomly selected integer does not have a corresponding entity 1. Return to Main Course step 1 and try again.

6.8 Generate IR Media Report

Name	Generate IR Media Report
ID	UC_008
Description	After the system runs the algorithm runoff, it generates a media report file that highlights the statistics, including election type, the winner, total seats, winning percentage.
Actors	The System.
Organizational Benefits	Provides an easier way to view the result and share it to the media.
Frequency of Use	Every time the system successfully runs the Instant Runoff algorithm and calculates the result.
Triggers	The system successfully runs the Instant Runoff algorithm and calculates the result.
precondition	Input file is in format and the algorithm runs successfully.
postcondition	A media report file for Instant Runoff will be generated following the format in Appendix C.3.
Main Course	1. System runs Instant Runoff algorithm 2. System generates a media report and places it in current directory
Alternative Courses	None
Exceptions	EX1 User decided to interrupt and close the program 1. Exit program.

6.9 Generate OPL Media Report

Name	Generate OPL Media Report
ID	UC_009
Description	After the system runs the algorithm OPL, it generates a media report file that highlights the statistics, including election type, winner, total seats, winning percentage.
Actors	The System.
Organizational Benefits	Provides an easier way to view the result and share it to the media.
Frequency of Use	Every time the system successfully runs the OPL algorithm and calculates the result.
Triggers	The system successfully runs the OPL algorithm and calculates the result.
precondition	Input file is in format and the algorithm runs successfully.

postcondition	A media report file for OPL will be generated following the format in Appendix C.4.
Main Course	1. System runs OPL algorithm 2. Generate a media report and place it in the current directory.
Alternative Courses	None
Exceptions	EX1 User decided to interrupt and close the program 1. Exit program.

6.10 Produce IR Audit File

Name	Produce IR Audit File
ID	UC_010
Description	When an IR election is held, an audit file is produced that lays out the step-by-step process by which the election was conducted, and includes all relevant statistics and calculations.
Actors	The system, Election Official(s)
Organizational Benefits	Allows election officials to verify the election by hand, ensuring that the election was conducted correctly and according to the rules of Instant Runoff.
Frequency of Use	An IR audit file is produced every time an IR election is run.
Triggers	The system completes UC_005.
Precondition	The system has completed UC_005 without error. All relevant data about the election is known to the system, including all information specified below in the Postcondition section.
Postcondition	A .txt file is produced that contains all relevant information as specified in Table 4.1 (see Appendix C.1 for an example file). The audit file is written to the program's directory The name of the audit file is of the form "election_audit_mm-dd-yyyy_hhmmss.txt"
Main Course	1. The system creates a new audit file with the proper name (see EX1, EX2). 2. All relevant information as defined in Table 4.1 is written to the audit file following the example in Appendix C.1.. 3. The system closes the audit file (see EX3).
Alternative Courses	None
Exceptions	EX1 System fails to open a .txt file 1. Write the message "WARNING: Failed to open audit file for writing. Trying again" to the terminal. 2. Attempt Main Course Step 1 again. EX2 There exists a file with the same name already in current directory 1. Append the string "_v2" to the end of the file name 2. Attempt Main Course Step 1 with the new name. 3. If error EX2 occurs again, increment the number in the appended string by 1 (e.g. "_v3", "_v4", . . .) and attempt Main Course Step 1 until a unique file name is

	found. EX3 The system fails to close the audit file 1. Write the message "WARNING: Failed to close audit file. Trying again" to the terminal. 2. Attempt Main Course Step 3 again.

6.11 Produce OPL Audit File

Name	Produce OPL Audit File
ID	UC_011
Description	When an OPL (Open Party List) election is held, an audit file is produced that lays out the step-by-step process by which the election was conducted, and includes all relevant statistics and calculations.
Actors	The system, Election Official(s)
Organizational Benefits	Allows election officials to verify the election by hand, ensuring that the election was conducted correctly and according to the rules of Open Party List voting.
Frequency of Use	An OPL audit file is produced every time an OPL election is run.
Triggers	The system completes UC_005.
Precondition	The system has completed UC_005 without error. All relevant data about the election is known to the system, including all information specified below in the Postcondition section.
Postcondition	A .txt file is produced that includes the relevant information specified in Table 4.1 (see Appendix C.2 for an example file). The audit file is written to the program's directory. The name of the audit file is of the form "election_audit_mm-dd-yyyy_hhmmss.txt"
Main Course	1. The system creates a new audit file with the name as defined in the Postcondition section (see EX1, EX2). 2. All relevant information as defined in Table 4.1 is written to the audit file following the example in Appendix C.2.. 3. The system closes the audit file (see EX3).
Alternative Courses	None
Exceptions	EX1 System fails to open the audit file 3. Write the message "WARNING: Failed to open audit file for writing. Trying again" to the terminal. 4. Attempt Main Course Step 1 again. EX2 There exists a file with the same name already in current directory 4. Append the string "_v2" to the end of the file name 5. Attempt Main Course Step 1 with the new name. 6. If error EX2 occurs again, increment the number in the appended string by 1 (e.g. "_v3", "_v4", . . .) and attempt Main Course Step 1 until a unique file name is found. EX3 The system fails to close the audit file 3. Write the message "WARNING: Failed to close audit file. Trying again" to

	the terminal. 4. Attempt Main Course Step 3 again.

6.12 Show Preliminary Results

Name	Show Preliminary Results
ID	UC_012
Description	Once the program is done running preliminary results will be displayed to the screen via the terminal. These include things such as winners, type of election, and number of seats. It will also include the number of ballots cast and the number of votes each candidate got.
Actors	System
Organizational Benefits	Allows for easy, quick, and readable statistics on the election that doesn't involve having to scan through a large audit file.
Frequency of Use	Everytime the program runs.
Triggers	The user runs the program and the program finishes running.
precondition	The input file is in CSV format and the program finishes running the election successfully.
postcondition	Program displays preliminary results to screen including winners, type of election, and number of seats, following the format in Appendix C.6. It will also include the number of ballots cast and the number of votes each candidate got.
Main Course	1. System accesses election data 2. System displays preliminary statistics to the screen
Alternative Courses	None
Exceptions	EX1 User decided to close the program 1. Exit the program EX2 System can't find all necessary data to calculate the statistics 1. Print out message to user "missing information on election" 2. Retry Step 1 of Main Course 3. If it fails again then exit program

6.13 Debug Mode

Name	Debug Mode
ID	UC_013
Description	Offers a non-deterministic (not randomized) option for breaking ties.
Actors	Testers, The System

Organizational Benefits	Allows testers to more efficiently test the voting system without having to account for randomization within the election process.
Frequency of Use	Whenever the program needs to be tested in a non-deterministic setting.
Triggers	A “-d” flag is indicated in the command line when starting the program.
Precondition	There exists two or more candidates or parties that have received an equal number of votes. The relevant candidates/parties are known to the system. Only one candidate/party among those tied must be chosen.
Postcondition	A candidate/party is chosen non-deterministically through the process outlined in the Main Course.
Main Course	1. The system determines whether the tie is between candidates or parties. 2. If the tie is between candidates, the tied candidate listed first in the list of candidates in the input file is selected. 3. If the tie is between parties, the tied party that has one of their candidates listed first in the list of candidates is selected.
Alternative Courses	None
Exceptions	None

Appendix A: Glossary

IR -> Instant Runoff: A voting algorithm that developed to ensure that the winning candidate has the majority support. If no candidate reaches 50% of total votes. The candidate with the least vote will be eliminated. And his votes will be distributed to other candidates. Keep eliminating until one candidate has equal or above 50% of total votes. Instant Runoff does not require a separate election, because there will also be a winner.

OPL -> Open Party List: A voting algorithm that proportionally allocates seats to parties by vote quota. Compared to a closed party list, the open party list system allows voters to select individuals rather than just parties.

Audit File: A file that contains each step of the election progress. The purpose of this file is to help users to verify the voting result.

Media File: A file that contains basic information for the voting result, including the winner, total ballots, number of candidates. The purpose of this file is to provide an intuitive result that can be shared through media.

Appendix B: Analysis Models

Class diagram, activity diagram , sequence diagram will be added in the further.

Appendix C: Output File Examples

C.1 Example IR Audit File:

election_audit_07-04-1776_115501.txt

```
****** OFFICIAL ELECTION AUDIT FILE ******
File generated on July 4, 1776 at 11:55:01 AM
Voting System : Instant Runoff (IR)


*** CANDIDATES ***
5 Candidates:
Alice Appleseed (Sensible)
Bob Burns (Sensible)
Charlie (Silly)
David Van Doe (Silly)
Isabella Indy (Independent)


*** BALLOTS ***
15 ballots cast
Ballots are shuffled and given IDs b1 to b15
Ballot's candidates listed in order of rank, first choice listed first
Candidate needs 8 or more votes to win

b1 : Appleseed,Burns,Charlie
b2 : Doe
b3 : Burns,Appleseed,Indy,Charlie,Doe
b4 : Indy,Doe
b5 : Appleseed,Charlie,Burns
b6 : Charlie,Appleseed,Doe,Indy
b7 : Indy,Appleseed,Doe,Burns,Charlie
b8 : Appleseed,Burns
b9 : Doe,Charlie
b10 : Charlie,Indy,Appleseed,Doe,Burns
b11 : Doe,Indy,Appleseed,Charlie,Burns
b12 : Appleseed,Burns,Charlie,Doe,Indy
b13 : Indy,Burns,Appleseed,Charlie
b14 : Charlie,Doe,Appleseed
b15 : Indy


*** ROUND 1 ***
Appleseed : 4 votes
      [b1,b5,b8,b12]
Burns : 1 vote
      [b3]
Charlie : 3 votes
```

```
      [b6,b10,b14]
Doe : 3 votes
      [b2,b9,b11]
Indy : 4 votes
      [b4,b7,b13,b15]

Total valid ballots : 15

No winner
Burns eliminated
1 vote transferred to Appleseed
      [b3]


*** ROUND 2 ***
Appleseed : 5 votes
      [b1,b3,b5,b8,b12]
Charlie : 3 votes
      [b6,b10,b14]
Doe : 3 votes
      [b2,b9,b11]
Indy : 4 votes
      [b4,b7,b13,b15]

Total valid ballots : 15

No winner
Charlie eliminated (Coin Flip)
1 vote transferred to Appleseed
      [b6]
1 vote transferred to Doe
      [b14]
1 vote transferred to Indy
      [b10]


*** ROUND 3 ***
Appleseed : 6 votes
      [b1,b3,b5,b6,b8,b12]
Doe : 3 votes
      [b2,b9,b11,b14]
Indy : 5 votes
      [b4,b7,b10,b13,b15]

Total valid ballots : 15

No winner
Doe eliminated
1 vote transferred to Appleseed
      [b14]
1 vote transferred to Indy
      [b11]
2 votes discarded
      [b2,b9]
```

```
*** ROUND 4 ***
Appleseed : 7 votes
      [b1,b3,b5,b6,b8,b12,b14]
Indy : 6 votes
      [b4,b7,b10,b11,b13,b15]

Total valid ballots : 13
Two candidates left, popularity wins

Appleseed wins
```

C.2 Example OPL Audit File:

election_audit_11-08-2020_010937.txt

```
****** OFFICIAL ELECTION AUDIT FILE ******
File generated on November 8, 2020 at 01:09:37 PM
Voting System : Open Party List (OPL)


*** ELECTION OVERVIEW ***
4 Parties, 6 Candidates
4 Seats to be filled

*** PARTIES AND CANDIDATES ***
Independent:
  Isabella Indy
Sensible:
  Alice Appleseed
  Bob Burns
Silly:
  Charlie
  David Van Doe
Slightly Silly:
  Erica Esther


*** BALLOTS ***
20 ballots cast
Ballots are shuffled and given IDs b1 to b20
Ballot's chosen candidate and party listed

b1 : Burns (Sensible)
b2 : Indy (Independent)
b3 : Burns (Sensible)
b4 : Charlie (Silly)
b5 : Doe (Silly)
b6 : Appleseed (Sensible)
b7 : Doe (Silly)
b8 : Appleseed (Sensible)
b9 : Indy (Independent)
b10 : Burns (Sensible)
```

```
b11 : Appleseed (Sensible)
b12 : Indy (Independent)
b13 : Charlie (Silly)
b14 : Charlie (Silly)
b15 : Burns (Sensible)
b16 : Burns (Sensible)
b17 : Esther (Slightly Silly)
b18 : Burns (Sensible)
b19 : Esther (Slightly Silly)
b20 : Doe (Silly)


*** VOTE DISTRIBUTION ***
Independent : 3 votes
  Indy : 3 votes

Sensible : 9 votes
  Appleseed : 3 votes
  Burns : 6 votes

Silly : 6 votes
  Charlie : 3 votes
  Doe : 3 votes

Slightly Silly : 2 votes
  Esther : 2 votes


*** FIRST ALLOCATION ***
QUOTA = number of seats / number of ballots cast
  A party needs 5 votes per seat awarded

Number of allocated seats for party
 = floor(number of votes party received / quota)

Independent wins 0 seats
Sensible wins 1 seat
Silly wins 1 seat
Slightly Silly wins 0 seats

2 seats filled


*** REMAINDER ALLOCATION ***
2 seats remaining

Remaining votes calculated by
  num votes party received - (quota * num seats allocated at first allocation)

Independent : 3 remaining votes
Sensible : 4 remaining votes
Silly : 1 remaining vote
Slightly Silly : 2 remaining votes

Remaining seats chosen in order of party with most remaining votes to least
Sensible -> Independent -> Slightly Silly -> Silly
Sensible wins 1 additional seat
Independent wins 1 additional seat


*** FINAL SEAT ALLOCATION ***
Sensible wins 2 seats
Independent wins 1 seat
Silly wins 1 seat
Slightly Silly wins 0 seats


*** CANDIDATE ALLOCATION ***
Candidates listed in order of votes received
Candidates at top of list for each party
  selected first for seats

Independent:
  Indy
Sensible:
  Burns
  Appleseed
Silly:
  Doe (Coin Flip)
  Charlie (Coin Flip)
Slightly Silly:
  Esther


*** RESULTS ***
Appleseed wins seat
Burns wins seat
Doe wins seat
Indy wins seat

There are 0 remaining seats to be filled
No subsequent election needed
```

C.3 Example IR Media File

election_media_10-03-2020_035505.txt

```
****** OFFICIAL ELECTION MEDIA FILE ******
File generated on November 3, 2020 at 3:55:05 AM
Voting System : Instant Runoff (IR)

*** UNITED STATES MAINE SENATE RACE ***

*** CANDIDATES ***
5 Candidates:
Alice Appleseed (Sensible)
Bob Burns (Sensible)
Charlie (Silly)
David Van Doe (Silly)
Isabella Indy (Independent)
```

```
*** RESULTS ***
Total Votes - 15,660 Votes

Alice Appleseed (Sensible) : (51%) - 7,987 Votes - ELECTED
Bob Burns (Sensible) : (30%) - 4,698 Votes
Charlie (Silly) : (19%) - 2,975 Votes
David Van Doe (Silly) : Eliminated Round 2
Isabella Indy (Independent) : Eliminated Round 1
```

C.4 Example OPL Media File

election_media_10-06-2020_115701.txt

```
****** OFFICIAL ELECTION MEDIA FILE ******
File generated on November 6, 2020 at 11:57:01 AM
Voting System : Open Party List(OPL)

*** UNITED KINGDOM GENERAL ELECTION ***

*** ELECTION OVERVIEW ***
4 Parties, 6 Candidates
4 Seats to be filled

*** PARTIES AND CANDIDATES ***
Independent:
  Isabella Indy
Sensible:
  Alice Appleseed
  Bob Burns
Silly:
  Charlie
  David Van Doe
Slightly Silly:
  Erica Esther


*** RESULTS ***
Total Votes - 18,807 Votes

Silly (51%) - 9592 Votes
Sensible (45%) - 8463 Votes
Independent (2%) - 376 Votes
Slightly Silly (2%) - 376 Votes

*** Winners ***
Silly:
  Charlie - ELECTED
  David Van Doe - ELECTED
Sensible:
  Alice Appleseed - ELECTED
  Bob Burns - ELECTED
```

C.5 Example Preliminary Results for IR

Printed to Terminal to Display on Screen

```
*** UNITED STATES MAINE SENATE RACE ***

Type of Election : Instant Runoff

*** CANDIDATES ***
5 Candidates:
Alice Appleseed (Sensible)
Bob Burns (Sensible)
Charlie (Silly)
David Van Doe (Silly)
Isabella Indy (Independent)

*** RESULTS ***
Total Votes - 15,660

Alice Appleseed (Sensible) : (51%) - 7,987 Votes - ELECTED
Bob Burns (Sensible) : (30%) - 4,698 Votes
Charlie (Silly) : (19%) 2,975 Votes
David Van Doe (Silly) : Eliminated Round 2
Isabella Indy (Independent) : Eliminated Round 1
```

C.6 Example Preliminary Results for OPL

Printed to Terminal to Display on Screen

```
*** UNITED KINGDOM GENERAL ELECTION ***

Type of Election : Open Party List

*** ELECTION OVERVIEW ***
4 Parties, 6 Candidates
4 Seats to be filled

*** PARTIES AND CANDIDATES ***
Independent:
  Isabella Indy
Sensible:
  Alice Appleseed
  Bob Burns
Silly:
  Charlie
  David Van Doe
Slightly Silly:
  Erica Esther


*** RESULTS ***
Total Votes - 18,807 Votes

Silly (51%) - 9592 Votes
```

```
Sensible (45%) - 8463 Votes
Independent (2%) - 376 Votes
Slightly Silly (2%) - 376 Votes

*** Winners ***
Silly:
  Charlie - ELECTED
  David Van Doe - ELECTED
Sensible:
  Alice Appleseed - ELECTED
  Bob Burns - ELECTED
```