Software Requirements Specification

for

Project Waterfall

Version 1.0 approved

Prepared by Amy Nguyen, Hoin Jang, Kaley Schiffler, William Henning

Team 13

February 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Audit use case	3/4/22	Missing audit file output use case	1.1

1. Introduction

1.1 Purpose

The purpose of this document is to characterize the features of an election-running system that supports two types of voting: Instant Runoff Voting (IRV) and Open Party Listing (OPL). It will explain the purpose and features of the software, the interfaces of the software, what the software will do, and the constraints under which it must operate. This document is intended for users of the software and potential developers.

1.2 Document Conventions

This document was created based on the IEEE template for System Requirement Specification Documents. The following use cases were modeled after a standard use case structured specification template.

1.3 Intended Audience and Reading Suggestions

For intended users: please refer to the following list as a guide to sections of this document that will be more pertinent.

- Election officials who will be using the software to run elections.
    - Section 1.4: Produce scope
    - Section 2: Overall Description
    - Section 5.4: Software Quality Attributes
    - Section 5.5: Business Rules
    - Appendix A: Glossary
- Testers who plan to test the voting features and better understand election results.
    - Section 2: Overall Description
    - Section 4: System features
- Programmers who are interested in adding, fixing, or changing the software in the future.
    - Section 2.2: Product Functions
    - Section 4: System features
- Media officials who want simple readable results from the election.
    - Section 1.4: Product Scope
    - Section 5.5: Business Rules
    - Appendix A: Glossary

1.4 Product Scope

An election-running system is being created to determine the winners of an election in different types of voting systems. The system will be able to handle Instant Runoff Voting (IRV) and Open Party List voting (OPL). The system will analyze the ballots to determine a winner and in the case of a tie, a coin will be flipped to determine the winner. An audit file containing all of the election information will be created in case election officials need to review the election. The system will output the winner and simplified information about the election for media officials to view.

1.5 References

IEEE Template for System Requirement Specification Documents:
https://goo.gl/nsUFwy

Use Case Documents (also provided in the SRS directory)
https://bit.ly/proj-waterfall-usecase-13

2. Overall Description

2.1 Product Perspective

This product will be able to perform elections with two different types of voting systems: Instant Runoff Voting (IRV) and Open Party Listing (OPL). The user will be able to choose which of the two systems to run. IRV is a system used in national elections in several countries. For example, members of the Australian House of Representatives, the president of India, the president of Ireland, etc., are elected through IRV. OPL is another system used in most European democracies and many newly democratized countries, including South Africa. Both voting systems are also used by several other political parties and private associations for various election purposes.

The output of this product will produce election results in a way that is clear and understandable to users. Information and data about the distribution of votes will be available for both voting systems.

2.2 Product Functions

First, users will be able to input a CSV file. The program will parse the header data for election type (IRV/OPL), number of votes, numbers of candidates, and candidate names. Depending on the header, the program will run the IRV system or the OPL system.

After running IRV or OPL, the program will output an audit file and display a simple version of the election results to the screen.

2.3 User Classes and Characteristics

1. Typical users, such as election officials who want to create data or statistical documentation on the number of votes cast by candidates.
2. Testers who plan to test the voting features and better understand the election.
3. Programmers who are interested in adding, fixing, or changing the software in the future.
4. Media officials who want simple readable results from the election.

2.4 Operating Environment

- Mac OS X
- Windows
- CSE Lab Machines (Keller Hall 1-250, VOLE)

2.5 Design and Implementation Constraints

The product will be developed using C/C++ and will need to run on a CSE lab machine. Voting information must be specifically formatted on a CSV file. The product must output an audit file.

2.6 User Documentation

A “README” file will be included specifying the necessary command line arguments to properly compile and run the program.

Example CSV file input for IRV:

Example File Generated for instant Runoff Voting:

```
IRV
4
6
Rosen (D), Kleinberg (R), Chou(I), Royce (L)
1,3,4,2
1,,2,
1,2,3,
3,2,1,4
,,1,2
,,,1
```

1st Line: IRV if instant runoff
2nd Line: Number of Candidates
3rd Line: Number of ballots in the file
4th Line: The candidates separated by commas (no space will follow the comma)

Example CSV file input for OPL:

Example File Generated for Open Party List voting:

```
OPL
6
3
9
[Pike,D],[Foster,D],[Deutsch,R],[Borg,R],[Jones,R],[Smith,I]
1,,,,,
1,,,,,
,1,,,,
,,,,1,
,,,,,1
,,,1,,
,,,1,,
1,,,,,
,1,,,,
```

1st Line: OPL for open party listing
2nd Line: Number of Candidates
3rd Line: Number of Seats
4th Line: Number of Ballots
5th Line: The candidates and their party in[]

2.7 Assumptions and Dependencies

This voting system program is developed in C++ and therefore requires C++ to be installed on the user's system. So this program needs an IDE that can run C++. Moreover, this program applies to Windows CSE Lab Machines and Mac OS X.

We assume the CSV input file will contain no errors and that the file will not be large enough to overflow the available memory.

3. External Interface Requirements

3.1 User Interfaces

When the appropriate voting system algorithm has been implemented and the results have been determined, an audit file will be created with the winners, losers, and other election information. This will be available for media officials to view. In addition, there will be a basic list of election winners displayed to the screen.

3.2 Hardware Interfaces

The program will work for computers that are capable of running cmake and C++. No other hardware interfaces are applicable to this project.

3.3 Software Interfaces

The software requires C/C++ to be installed on the system in order for the voting system to run. Additional information can be found on section 2.7 of this document.

3.4 Communications Interfaces

No communication interfaces are applicable with this project.

4. System Features

4.1 Run IRV

Name	Run IRV
ID	UC_001
Description	An election official or a tester wants to run IRV on a collection of ballot votes
Actors	Election officials, testers
Organizational Benefits	Produces the results of an IRV with given information
Frequency of Use	Every time an IRV election file is run with the program.
Triggers	The actor runs the program with a file.
Preconditions	1. Every ballot has at least one candidate selected 2. Every ballot has at least two candidates
Postconditions	1. If there are two losers, then a coin must be flipped in order to determine the loser 2. Results get outputted to the screen
Main Course	1. Check file for initial header data (IRV/OPL, number of votes, number of candidates, candidate names) 2. Load file data into the heap 3. Shuffle the ballots 4. Sort ballots into corresponding lists for each candidate based on their first choice 5. Log # of votes for each candidate 6. Distribute votes of the lowest candidate to their next choice

	7. Go to step 5 until theres is a candidate with the majority vote 8. Output audit file 9. Output the results to screen
Alternate Courses	AC1 file header reads "OPL" 1. Run OPL (UC_002) instead AC2 Shuffle is disabled (UC_004) 1. Do not shuffle ballots 2. Continue from MC4 AC5 If a candidate gets >50% of the ballot 1. Declare winner 2. Go to MC 8 AC6 Tie for lowest candidate 1. Flip a coin if enabled to determine lowest votes AC6 Ballot has no next choice 1. Remove ballot from the heap

4.2 Run OPL

Name	Run OPL
ID	UC_002
Description	An election official or a tester wants to run OPL on a collection of ballot votes
Actors	Election officials, testers
Organizational Benefits	Produces the results of an OPL with given information
Frequency of Use	Every time an OPL election file is run with the program.
Triggers	The actor runs the program with a file.
Preconditions	1. Every ballot has exactly one candidate selected
Postconditions	1. If there candidates are tied for the remainder votes, do a coin flip to determine which party gets the seat 2. Results get outputted to the screen
Main Course	1. Calculate quota for advancement

	2. Sort the party candidates by total votes 3. Advance the number of most popular candidates from each party equal to the floor of their total votes divided by the quota, saving the remainder. 4. For the remaining seats, the parties with the highest remainder have their candidates advanced. 5. Output audit file 6. Output the results to screen
Alternate Courses	AC2a. file header reads “IRV” 1. Set up a data structure to fill with IRV ballots. 2. Fill the data structure with ballots. 3. Run IRV (UC_001) AC4 Tie for Remaining Votes 1. Flip a coin to determine who gets the last seat.
Exceptions	EX. Fails to output winner to screen 1. System notifies that an error has occurred. EX. There are more seats than candidates available 1. System notifies that there are not enough candidates available

4.3 CSV Input

Name	CSV Input
ID	UC_003
Description	An election official or tester wants to input the election data into the program as a CSV file
Actors	Election officials, testers
Organizational Benefits	Allows for the input of data to run elections
Frequency of Use	Every time an election is run
Triggers	The actor enters the terminal command with a CSV file as an argument
Preconditions	1. A CSV file has been created accurately with a header that lists election type, number of candidates, number of ballots, candidate names, and number of seats.
Postconditions	1. Ballots are loaded into the program and ready for

	processing
Main Course	1. Access file and create file input stream. 2. Check header for election type. 3. File header has no specified type 4. Ask actor for election type
Alternate Courses	AC2a. file header reads “IRV” 1. Set up a data structure to fill with IRV ballots. 2. Fill the data structure with ballots. 3. Run IRV (UC_001) AC2b. file header reads “OPL” 1. Set up a data structure to fill with OPL ballots. 2. Fill the data structure with ballots. 3. Run OPL (UC_002)
Exceptions	1. Cannot access file and create file input stream(error,bug).

4.4 Toggle Ballot Shuffle

Name	Toggle Ballot Shuffle for IRV
ID	UC_004
Description	A tester needs to be able to disable ballot shuffling for testing purposes.
Actors	Testers
Organizational Benefits	Allows for consistent testing of IRV ballot data
Frequency of Use	Any time tester wants to test IRV data
Triggers	Runs election program with shuffle-off flag
Preconditions	1. IRV ballot is being used.
Postconditions	1. Shuffle is turned off for the following IRV run (UC_001)
Main Course	1. Disable shuffling functionality of the IRV ballot counting algorithm
Alternate Courses	
Exceptions	EX. System gives an error when actor tries to turn off shuffle for an OPL ballot file

4.5 View Election Results

Name	View election results
ID	UC_005
Description	Media personnel want to know the winner(s) and information about the election (e.g. type of election, number of seats).
Actors	Media officials
Organizational Benefits	The media knows the results of the election
Frequency of Use	Whenever election is completed
Triggers	IRV or OPL completed
Preconditions	All of the ballots have been collected and the winners have been determined.
Postconditions	Output file for simple, understandable election results
Main Course	1. File with tables are outputted showing the finals results, total votes, and percentage of votes for each candidate
Alternate Courses	
Exceptions	EX. Statistically incorrect results are obtained.

4.6 Generate Audit File

Name	Generate Audit File
ID	UC_006
Description	A file of election information and results for election officials.
Actors	Election Officials
Organizational Benefits	The election officials will be able to review the election results in case they expect an error or invalid election.
Frequency of Use	Whenever an election is completed

Triggers	IRV or OPL completed
Preconditions	All of the ballots have been collected and the winners have been determined.
Postconditions	Output file with all the necessary information to faithfully recreate the election.
Main Course	1. The system runs an election 2. Audit file is generated upon completion of the election
Alternate Courses	
Exceptions	EX. The system generates an audit file with inconsistent election results

4.7 Coin Flip

4.2.1 Description and Priority

High priority. In the case of a tie any time within the election, winners and losers must be decided by a fair coin flip. The coin flip itself will be a very simple function.

4.2.2 Stimulus/Response Sequences

A user will run the system as normal. If a tie has been indicated, the program will initiate a coin flip. To ensure fairness, the coin will be flipped several times and the last one will be the result. The results of the coin flip will be recorded and the program will move on with the resulting winner and loser. The user will not be required to perform any other actions outside of running the system and waiting for the results.

4.2.3 Functional Requirements

REQ-1: The coin flip must be random and fair.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The product must be able to run an election with 100,000 ballots in under 8 minutes on a CSE lab machine. Slow systems and long wait times are not desirable, hence a faster and efficient system is required.

5.2 Safety Requirements

No safety requirements are applicable with this product.

5.3 Security Requirements

No security requirements are applicable with this product.

5.4 Software Quality Attributes

This election-running system will provide users with accurate and fair election results with either an IRV or OPL voting system in a reasonable amount of time. However, users may have to familiarize themselves with running a program from a command prompt.

5.5 Business Rules

Only election officials will be able to run either an IRV or an OPL. They will also receive an election audit and a document of the results.

Any other persons, including the general public and media personnel, may receive a results document upon request of election officials but will not be permitted to use the product.

Testers of the product may turn off IRV ballot shuffling in order to test for correctness of test outputs.

6. Other Requirements

No other requirements known at this time, TBD.

Appendix A: Glossary

Instant Runoff Voting (IRV): All candidates are listed on the ballot and voters rank the candidates in order of their preference. If a candidate receives over 50% of the first choice votes, that candidate is declared elected. If no candidate receives a majority, then the candidate with the fewest votes is eliminated and their votes are transferred to the voters number two choice. The votes are recounted to see if a candidate receives a majority of the votes. The process is repeated until all the seats have been filled.

Open Party List Voting (OPL): Each political party has their own set of candidates. Voters are able to select one candidate which also indicates which party they are choosing. The number of seats each party will receive are determined by the percentage of votes their party received.

Comma Separated Values (CSV): Comma separated values. The file is a delimited text file and will be used as the only type of input file.