Software Requirements Specification

for

Voting System

Version 2.0 approved

Prepared by <Dahir Ali, Julian Heyman, Matt Graba, Muhsin Mohamed>

<CSCI 5801>

Spring 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

First Pass	2/10/2023	Establish preliminary SRS	1.0
Final Pass	2/16/2023	Finished Version of SRS	2.0

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the voting system. It will explain the purposes and features of this system, what the system will do and the constraints it must operate under. The document is intended for users of the software and developers.

1.2 Document Conventions

This document was created based on IEEE template for System requirement specification documents.

1.3 Intended Audience and Reading Suggestions

This document is intended for different type of readers such as:

- Developers who are fixing bugs and to continue further development of the project

- Users such as election officials who want to understand how to operate the system and that the system follows all the requirements that the user needs

- Documentation writers who need to create the SRS document and create changes if needed

The SRS contains the requirement specifications of the system and is organized into different Sections describing the system. We suggest that the reader reads the most pertinent sections to them. However, to fully understand the election system we suggest they read the whole document.

1.4 Product Scope

The voting system is a software that will allow users to input a delimited text file containing information about the election. The software will then read in the file and produce an audit file containing the election information at the time. The software should also show how the election progressed so that the audit can replicate the election itself. The software should also be able to handle both Instant Runoff Voting and Closed party listing voting. The software should then display to the screen the winner and information about the election.

1.5 References

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

UseCases_Team#24 (Should be in the same GitHub repository)

2. Overall Description

2.1 Product Perspective

The voting system software was developed for election officials who are interested in a voting system that is accurate, fast and secure. The software can run up to 100,000 ballots in under 4 minutes. The software can handle both Closed party listing and instant runoff voting and can produce an audit file providing information about parties, candidates, ballots, ranks and even more. The software reads in a text file and displays on screen the winner of the election.

2.2 Product Functions

- Identify File
- Read File
- Process File
- Prompt User for extra Information
- Handle on ballot number
- Run Instant Runoff Election
- Run Closed Party listing Election
- Display results
- Handle Popularity Voting
- Tie breaker
- Produce Audit File

2.3 User Classes and Characteristics

- Developers who are fixing bugs and to continue further development of the project

- Election officials who use the software to determine the winner of an election

- Testers who create and run unit tests to make sure that the software has no bugs

2.4 Operating Environment

- Ubuntu 20.04

2.5 Design and Implementation Constraints

At current implementation the program can only handle an election if all ballots are within a single file, this program will not work if a single election's votes are spread out across multiple files. Furthermore outside of the audit file this program does not log or save results anywhere, if a user would like to see the information of an election and they don't have the audit file they will have to rerun the election.

2.6 User Documentation

Users will be required to know how to manage and input command line arguments into their operating system to be able to run the software and respond to future prompts throughout the operation of the program. A useful guide to understand how command line arguments are operated can be found in the following link.

https://www.bleepingcomputer.com/tutorials/understanding-command-line-arguments-and-how-to-use-them/

The user is not required to obtain any knowledge of technical components within this software through any means of external tutorials or documentation. The system will handle the technological components pertaining to the processing and evaluation of the election. For any information the user will need to provide, they will be prompted accordingly on the command line.

2.7 Assumptions and Dependencies

The biggest assumption is that the ballot file needs to be readable, i.e. its user permissions are at the very least set to read. In the case that they aren't users will need to chmod to give permissions or files cannot be processed.

3. External Interface Requirements

3.1 User Interfaces

For current implementation the program interacts only with the command line interface both for inputs and outputs, excluding the audit file it will produce.

3.2 Hardware Interfaces

The minimum hardware requirements for this software is an Intel i7 @2.5 GHz or equivalent CPU. The program will also require at least 32 GB of RAM or unified memory equivalent.

3.3 Software Interfaces

The Voting System program requires that Java/JDK be installed on the operating system. The latest version of Java (Version 19) would be preferable to prevent any errors caused by outdated versions. By utilizing command line functions and the Java software, files consisting of a CSV format will contain the data items coming into the system.

3.4 Communications Interfaces

The Voting System program requires an internet connection so that it can be updated as a whole in the future or if it needs an overhaul on specific components.

4. System Features

This section describes the voting system software's features and explains how they are used and what result they will produce.

4.1 File processing and analysis

4.1.1 Description and Priority

A user of the program can pass in a file to run a voting election against the program will iterate the file and analyze the votes to determine a winner based off the selected voting system.

Priority: High

4.1.2 Stimulus/Response Sequences

To stimulate the file processing the user simply has to start the program, they can either pass the file in on start or wait to be prompted for a file name. After the program validates the file, the user will be prompted if they would like to begin reading the contents of the file so that the information it contains can be handled. After the contents are successfully read, the system will then process the information into separate data variables that will be used in separate system features in the process of analyzing and evaluating the election.

4.1.3 Functional Requirements

REQ-1:	Identify File
REQ-2:	Read File
REQ-3:	Process File

Name	**Identify File**
ID	UC_001_identify_f
Description	The CSV file that contains the details of the election will be identified based on its location within the directory.
Actors	Any actors that would benefit from the information within the file (e.g., testers, programmers, election officials, etc.).
Organizational Benefits	Enable the program to be functional by being able to successfully locate and identify the CSV file.
Frequency of Use	The file will only be identified once upon the startup of the program.
Triggers	An actor runs the program.
Preconditions	● The program is started via a command line argument ● The file exists ● The file is a CSV file ● The file is located within the correct directory
Postconditions	● The file is located and correctly identified so it is ready to be read ● The type of election is identified
Main Course	1. A user runs the program. 2. The user is prompted for a file name by the system 3. The directory is searched for a file (See EX1) 4. A file is located within the directory (See AC1, AC2)
Alternate Courses	AC1) CSV File is located

	1. See use case 'Read File' AC2) Non-CSV file is located 1. The system informs the user that the file located is not a CSV file and that the file must be CSV to continue 2. The program exits
Exceptions	EX1 Directory does not contain a file. 1. System informs the user that the directory does not contain a file 2. The program exits

Name	**Read File**
ID	UC_002_read_f
Description	Acquires the information from the file by reading each line and storing that information in a database to be used in future use cases.
Actors	Programmers, testers, election officials (Anyone that benefits from any item located within the file).
Organizational Benefits	Thoroughly reads and records the contents of the file by maintaining election integrity so that accurate information can be stored for future use.
Frequency of Use	File is read one time after the program is run in which all data will be stored in external memory locations so that the file won't need to be read again.
Triggers	Users are prompted if they would like to read the file.
Preconditions	An actor has run the program and the file has already been previously identified and processed. The file should be located in the correct directory for it to be identified.
Postconditions	The contents of the file are accurately read and stored as data into separate variables for future use.
Main Course	1. The system prompts the user if they would like to read the contents within the file while displaying specific inputs for the prompt. 2. The user inputs whether they would like to read the file or leave the program. 3. If the user selects they would like to read the file, the program

	reads the file and stores all of the file contents into separate variables. 4. If the user selects they would not like to read the file, the system exits prompt (AC1) 5. If the user inputs incorrectly, the system will display the requirements for the prompt and prompt the user again (EX1) 6. User is then prompted if they would like to continue or exit the program
Alternate Courses	AC1) User inputs they do not want to read file 1. The system exits the prompt and asks the user if they would like to close program 2. The user inputs whether to continue or close the program 3. The system acts accordingly on user input
Exceptions	EX1) User inputs incorrectly into prompt 1. The system responds to the user with incorrect input and repeats the correct inputs that must be entered specifically 2. The user then enters one of the correct inputs otherwise step will continuously repeat until the user enters correctly.

Name	**Process File**
ID	UC_003_process_f
Description	Acquires the information that is read from the file so that it may be processed accordingly based on the election type as well as the ballot information within the file.
Actors	Any actor that benefits from the contents of the file (e.g., programmers, testers, election officials, etc.)
Organizational Benefits	Ensures the integrity of the election by accurately processing correct information from the file and storing the data accordingly.
Frequency of Use	The file will be processed once after the program is initially run and the file is identified and read.
Triggers	Upon the completion of the file being read, it will simultaneously be processed.
Preconditions	● The file is correctly identified

	● The file is correctly read
Postconditions	● The type of election contained within the file is determined ● The details of the election are obtained ● Data is processed and recorded to allow the handling of the election in future use cases
Main Course	1. The header is skipped over so that the type of election can be processed 2. Type of the election is processed (See AC1, AC2) 3. The number of candidates/parties are processed (See AC1, AC2) 4. The names of candidates/parties are processed (See AC1, AC2) 5. The number of ballots/seats are processed (See AC1, AC2) 6. All details of the file are processed into data variables that will be transferred to an audit file and used to continue evaluating the election.
Alternate Courses	AC1) Process IR File 1. Line 1 of the file reads 'IR' (See use case 'read file'). The type of election is processed as an Instant Runoff 2. Line 2 of the file reads the number of candidates (See use case 'read file'). The number of candidates is processed. 3. Line 3 of the file reads the names of the candidates. The names of the candidates are processed. 4. Line 4 of the file reads the number of ballots. The number of ballots is processed. AC2) Process CPL File 1. Line 1 of the file reads 'CPL' (See use case 'read file'). The type of election is processed as a Closed Party Listing. 2. Line 2 of the file reads the number of parties (See use case 'read file'). The number of parties is processed. 3. Line 3 of the file reads the names of the parties. The names of the parties are processed. 4. Line 4 of the file reads the number of seats. The number of seats is processed. 5. Line 5 of the file reads the number of ballots. The number of ballots is processed.
Exceptions	Information is missing from the file: 1. see use case Prompt User For extra Information

4.2 Handling type of election system

4.2.1 Description and Priority

Program should be able to run both a CPL and IR election as well as handle any ties

Priority: High

4.2.2 Stimulus/Response Sequences

The user simply has to wait for the system to finish running the election and determine the winner and it will print the audit file.

4.2.3 Functional Requirements

REQ-1:	Handle IR
REQ-2:	Handle CPL

Name	**Handle IR**
ID	UC_004_handle_ir
Description	Program is incorporated functionality that allows an Instant Runoff Ballot election. Utilizing information from the file, an accurate IR election will be conducted versus its contrary CPL.
Actors	Typically every user: Programmers, testers, election officials (Anyone that benefits from the election results)
Organizational Benefits	Information from the file will be correctly utilized and distributed
Frequency of Use	When a test is being run or election, not super frequent
Triggers	Actor starts the program
Preconditions	● File has been read in and processed for header information ● All required information has been acquired ● IR election has been determined
Postconditions	● Election results have been determined
Main Course	1. IR election has been determined 2. Ballots are processed using the given IR algorithm (see glossary for explanation) 3. The winner is determined
Alternate Courses	After all possible candidates have been eliminated there is still a tie of 2 or more people:

	1. See use case tie breaker During elimination, 2 candidates are tied for the least number of votes: 1. See use case tie breaker
Exceptions	Since file is assumed to be correct no exceptions should arise

Name	**Handle CPL**
ID	UC_005_handle_cpl
Description	Program has incorporated functionality that allows a Closed Party List Ballot election. Utilizing information from the file, an accurate CPL election will be conducted versus its contrary IR.
Actors	Typically every user: Programmers, testers, election officials (Anyone that benefits from the election results)
Organizational Benefits	Allows an actor to process a CPL ballot file without having to do it manually or design their own solution
Frequency of Use	When a test is being run or election for CPL
Triggers	● Program has determined CPL will be run either from actor or from file
Preconditions	● File header has been processed ● Required information has been acquired
Postconditions	Election winners have been determined
Main Course	1. CPL election has been determined 2. Ballots are processed using the given CPL algorithm (see glossary for explanation) 3. Winner is determined
Alternate Courses	A party receives more seats than they have space for: 1. See use case tie breaker During second allocation of seats there is a tie in parties: 1. See use case tie breaker
Exceptions	Since file is assumed to be correct no exceptions should arise

4.3 Evaluating election results

4.3.1 Description and Priority

Audit file is created showing the progression of the election and also shows election information such as type of voting, number of ballots, candidates and more. It should also show the winner

Priority: Medium

4.3.2 Stimulus/Response Sequences

The user simply has to wait for the system to finish running the election and determine the winner and it will print the audit file.

4.3.3 Functional Requirements

REQ-1:	Display Results
REQ-2:	Produce Audit File

Name	**Display Results**
ID	UC_006_display_r
Description	System will display to the user the results and details of the elections such as the winners as well as the number of ballots cast, the winners and the stats for everyone such as the number of votes received, the percentage of votes received.
Actors	Typically any user: Programmers, testers, election officials (Anyone that benefits from the election results).
Organizational Benefits	Provides election results in a detailed and well-formed structure that will make analyzing the results and details of the election simple and easy and find each item involved.
Frequency of Use	After the program has been run and all the details and information have been calculated and processed.
Triggers	After the program completes.
Preconditions	● The file has been identified in the correct directory ● The file has been read

	• The program has been completed by accurately processing the information contained within the file
Postconditions	Every detail within the election that would benefit the user is organized and accurately displayed to the screen
Main Course	1. The program is run 2. CSV file is successfully identified, read, and processed (See EX1) 3. A GUI is created to display the election results 4. The system accesses details of the election (See AC1, AC2) 5. The details of the election are organized into a text document that will be opened and displayed on the screen upon completion of the program
Alternate Courses	AC1) Details for an IR election 1. The winner of the IR election is added to text document 2. The number of votes received for the IR election is added to text document 3. The percentage of votes received for the IR election is added to text document 4. User is prompted for any other details of the election they would like to be displayed 5. The remaining additions that the user would like are added to the text document AC2) Details for a CPL election 1. The winner of the CPL election is added to text document 2. The number of votes received for the CPL election is added to text document 3. The percentage of votes received for the CPL election is added to text document 4. User is prompted for any other details of the election they would like to be displayed 5. The remaining additions that the user would like are added to the text document
Exceptions	EX1) Error in handling the CSV file 1. The system responds that the results cannot be displayed due to an error in handling the CSV file. 2. Review use cases UC_001-3

Name	**Produce Audit File**

ID	UC_007_produce_audit_file
Description	Produces an audit file with election information at the time
Actors	Election officials who need to ensure the voting system is working properly
Organizational Benefits	The audit file displays information about the election at the time and the winner. It should show how the election progresses so that the audit could replicate the election itself
Frequency of Use	This would occur after the election is ran and the winner is determined
Triggers	System has processed file and determines winner
Preconditions	System has processed file and determines winner
Postconditions	Audit file contains election information to determine accuracy of election
Main Course	Producing audit file in CPL: 1.System processes file 2.System determines winner 3.System produces audit file containing the following election information: - Type of Voting - Number of Candidates - Names of Candidates - Number of Ballots - Number of votes received by each candidate - Winner(s) of the election - Progression of the election, including the order in which ballots were received and assigned to candidates.
Alternate Courses	Producing audit file in IR: 1. System processes file 2. System determines winner 2. System produces audit file containing the following election information: - Type of Voting - Number of Candidates - Names of Candidates - Number of Ballots - Number of votes received by each candidate - Winner(s) of the election - Progression of the election, including the order in which

	candidates were removed and ballots were redistributed.
Exceptions	File is assumed to be correct so no exceptions

4.4 Exception Handling (No Ballot, Ties, No Majority)

4.4.1 Description and Priority

The program should be able to accommodate situations where there is not enough information given to it from the user such as when there is no ballot number. It should also be able to handle situations such as when two parties/candidates tie in the number of votes or in situations such as when there is no clear majority.

Priority: Low

4.4.2 Stimulus/Response Sequences

If the program does not receive ample information to conduct and simulate the elections. If any situation occurs where there is no clear winner it should defer to REQ-2 or REQ-3 depending on the situation.

4.4.3 Functional Requirements

REQ-1: Handle No Ballot Number
REQ-2: Handle Popularity Voting
REQ-3: Tie Breaker
REQ-4: Prompt User for Extra Information

Name	**Handle No Ballot Number**
ID	UC_008_handle_no_ballot_number
Description	While some information prevents us from running the voting system, in the case that the ballot number is unknown we can still run the program and count the number of ballots as we do the first pass
Actors	Any actor using the system who doesn't know the ballot number
Organizational Benefits	It's possible a ballot might not have the number of ballots nor the program runner knows the number, having this feature helps us process files even when the numbers unknown

Frequency of Use	Very rarely as you first need the number not on the ballot as well as the test runner not knowing
Triggers	After being prompted for missing information the user has selected to run the test without the provided info
Preconditions	● File has been processed ● User has been prompted at least once for missing information ● User has selected to run the file this alternative way
Postconditions	● Number of ballots determined
Main Course	1. User selects to run vote with first pass counting 2. Bitflag is turned on that adds a counter to the ballot processing 3. Voting system runs as normal except on the first pass where the counter is used 4. Number of ballots variable is set
Alternate Courses	No alternate courses
Exceptions	No exceptions

Name	**Handle Popularity Voting**
ID	UC_009_handle_popularity_v
Description	Program should be able to facilitate the handling of votes based on their popularity if no clear majority
Actors	Election Official
Organizational Benefits	Let's the system decide a winner in the case of an Instant Runoff election where there is no clear majority.
Frequency of Use	This would only happen within an IR election and it will only happen in cases where there is no clear candidate with majority of votes
Triggers	An actor runs a program
Preconditions	No winner has been determined after IR election has been run once
Postconditions	Election winner has been determined

Main Course	1. IR election has been determined 2. IR election has been processed with data from CSV 3. No clear majority (No winner) 4. Program checks popularity of each candidate 5. Program decides winner based on popularity
Alternate Courses	No clear majority and two most popular candidates are tied: 1. See Use Case Tie Breaker
Exceptions	In the case of the program failing to run, it should notify the user and return to Step 1.

Name	**Tie Breaker**
ID	UC_010_tie_breaker
Actors	Election Official
Organizational Benefits	Creates a tiebreaker in the rare case there is a tie. This can help avoid confusion and pick a definite winner in a situation where you can't have two winners.
Frequency of Use	This would only happen in the result of tie in the election between two parties/candidates, so it wouldn't be a common occurrence
Triggers	Users complete voting
Preconditions	System finishes tallying votes
Postconditions	Winner is randomly selected
Main Course	During IR, 2 or more candidates have tied after all possible eliminations 1. Users finish voting 2. System tallies votes 3. System detects two or more parties/candidates have the same amount of votes 4. System does a virtual coin flip by selecting a number at random and a random number for each tied option and determines who was closest 5. Winner is randomly selected through coin toss
Alternate Courses	Tie breaker is being used to eliminate a candidate during IR: 1. Follow main course step 3

	2. Follow main course step 4 3. Candidate to eliminate has been fairly and randomly selected In CPL A party has received more seats than they have candidates for: 1. For how many seats are available and up to how many seats a party has they are randomly selected a number (For example if there are 3 seats available but the party only has 2 candidates left the program would randomly select 2 numbers, if they have 4 candidates left then 3 numbers are selected) 2. N closest numbers are taken where N is the number of seats up for grabs 3. Winners are determined and assigned seats In CPL during allocation of extra seats there is a tie between parties for a seat: 1. Perform main course tie breaker as normal 2. Winner is randomly selected and assigned an extra seat to their total
Exceptions	1. One of the parties that tied decides to concede the election.

Name	**Prompt User For Extra Information**
ID	UC_011_ prompt_user_extra_info
Description	When an actor passes in a file there is certain information needed before running either vote, for IR we need the number of candidates, the candidates separated by a comma, and the number of ballots in the file and for CPL we need the number of parties, the parties, for each party a ranking of their candidates, the number of seats and the number of ballots. Both voting types also needed to be identified by the file. In the case that certain information is missing the user needs to be prompted for said information.
Actors	Any actor that runs this test (election official, tester, etc)
Organizational Benefits	If information is missing from the ballot, outside of the number our program has no way of identifying the missing information so it would not be able to proceed with the election, as such its important to have this function when information is missing

Frequency of Use	Since files are assumed to have no errors the frequency should be never, realistically if there is an error in the file we would want to report it to the file generators to fix, as such the frequency should be low
Triggers	Information missing from the file
Preconditions	● The file has been processed ● The file has been flagged for missing certain information ● The missing information is known
Postconditions	● Missing information has been provided and logged into the program
Main Course	1. The system identifies which vote is being run 2. The system identifies which info is missing 3. The system prompts the user for missing info 4. A user inputs any missing info
Alternate Courses	Info is still missing after first pass: 1. Go to Main Course 2
Exceptions	User cannot provide information about ballot numbers: See handle_no_ballot_number User cannot provide information about info other than ballot numbers: Without the required information the program cannot run, in the case that they cant provide information the actor will be prompted to either stop the program or restart

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The software has a runtime constraint in which an election needs to run 100,000 ballots in under 4 minutes. To ensure this, various system features should be designed to operate in their most optimal version.

5.2 Safety Requirements

There are no safety requirements needed in the software.

5.3 Security Requirements

There are no security requirements needed in the software as security such as ensuring one vote per person is handled at voting centers.

5.4 Software Quality Attributes

The software needs to be adaptable so that it can be a part of an integrated online voting system. The software interface should be easy to use so that customers with limited technical knowledge should be able use it. It should also be flexible enough so that developers and users have the option to adjust and make changes to the system if needed.

5.5 Business Rules

The software will be run multiple times during the year at normal elections and special elections by users such as the election officials, testers of the software, programmers, etc. Election officials will only benefit from the results of the program and are not given the functionality to edit or change the software. Testers of the software are only given access to the software functions and not the hardcore components of the program as they will communicate the errors resulting in the software to the programmers. In such circumstances where the software has experienced any problem resulting in a program error, a programmer contains a role that will offer them the ability to fix these errors and edit the components of the program.

6. Other Requirements

No other requirements as of now.

Appendix A: Glossary

Instant runoff*: All candidates are listed on the ballot, rather than voting for one candidate voters rank all the candidates in order of preference, at least 1 candidate is needed to be ranked to be a valid ballot. During the first pass all ballots are counted and a voter's first rank choice is counter. After all ballots are counted the winner is checked by seeing if any candidate has 50%+ more of the votes, if one does the election is called and they are the winner. If a winner cant be determined the candidate with the lowest votes is determined and eliminated, anyone who had them ranked as their first choice has their second choice vote counted. the process repeats until a winner is determined or a tie.*

Closed Party Listing*: Each party puts up a list of candidates equal to the number of seats up for grabs. During voting each voter votes for the party, not a specific person. Parties than receive seats proportional to the amount of votes they got. So if there are 10 seats up and a party gets 20% of the votes they get 2 seats. to distribute any remaining seats you take the remainder, so if a party got 23000 votes, with 10 seats open, their remainder would be 3000. the party with the highest amount of remaining votes gets the final seat, a coin toss is used for ties*

***Bug:** An unexpected error in software*

Runtime: *Amount of time it takes to run a program*

CPU: *Central processing unit - a processor that executes instructions making a computer program*

RAM: Random access memory - short term memory where the processor stores data temporarily

Appendix B: Analysis Models

No diagrams

Appendix C: To Be Determined List

No TBD