

# **Software Requirements Specification**

**for**

## **Project 1: Waterfall Methodology**

**Version 1.0 approved**

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

**CSCI 5801 – Team 7**

**February 2023**

# Table of Contents

<b>Table of Contents.....</b>	<b>2</b>
<b>Revision History.....</b>	<b>2</b>
<b>1. Introduction.....</b>	<b>3</b>
1.1 Purpose .....	3
1.2 Document Conventions .....	3
1.3 Intended Audience and Reading Suggestions .....	3
1.4 Product Scope .....	3
1.5 References .....	3
<b>2. Overall Description .....</b>	<b>4</b>
2.1 Product Perspective .....	4
2.2 Product Functions .....	4
2.3 User Classes and Characteristics .....	5
2.4 Operating Environment .....	5
2.5 Design and Implementation Constraints .....	6
2.6 User Documentation .....	6
2.7 Assumptions and Dependencies .....	6
<b>3. External Interface Requirements .....</b>	<b>6</b>
3.1 User Interfaces .....	6
3.2 Hardware Interfaces .....	8
3.3 Software Interfaces .....	8
3.4 Communications Interfaces .....	8
<b>4. System Features .....</b>	<b>8</b>
<b>5. Other Nonfunctional Requirements .....</b>	<b>8</b>
5.1 Performance Requirements .....	8
5.2 Safety Requirements .....	8
5.3 Security Requirements .....	9
5.4 Software Quality Attributes .....	9
5.5 Business Rules .....	9
<b>6. Other Requirements .....</b>	<b>9</b>
<b>Appendix A: Glossary .....</b>	<b>9</b>
<b>Appendix B: Analysis Models .....</b>	<b>9</b>

## Revision History

Name	Date	Reason For Changes	Version
Liam, Marcus, Lucas, Justin	1/30/2023	SRS Document Created & Started	1.0
Liam, Marcus, Lucas, Justin	2/16/2023	SRS Document Completed	1.0

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to provide a detailed description of the voting system. It will explain the purpose and features of the voting system, what the software will do, and the constraints under which the software must operate. This document is intended for the users of our system, which include election officials, programmers, testers, audit officials, and potential future developers.

## 1.2 Document Conventions

This document was created based on the IEEE template for System Requirement Specification Documents.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for election officials, audit officials, programmers, testers, and future developers. Since they have different goals, here is what they can expect from the document.

- Election officials and audit officials will want to be able to refer to this document if they ever need to clarify how the system was developed or how to use the voting system.
- Programmers, testers, and future developers will constantly refer back to this document to reference how the system was built and to know what requirements our team designed the software with.

The recommended sequence for reading this document would be to start with the Introduction section (section 1) where the reader can get all of the relevant starting knowledge needed to read the rest of the document. This includes, who this document is for and how they can best utilize it, the purpose of this document, any required notation conventions used, and a general scope of the software and its design. Following the introduction, the Overall Description section (section 2) will give the reader a deep dive into the product scope from the introduction providing an in-depth look into the functionality of the product and the environment in which it was developed. After the overall description, viewing the External Interface section (section 3) will give the reader a better understanding of how the system should look. Lastly, the System Features section (section 4) and the Other Nonfunctional Requirements section (section 5) will give the reader a supplemental in-depth look into the use cases for the system detailing and other requirements used to create the system which will substantiate the functionality of the software.

## 1.4 Product Scope

The voting system encapsulates two different voting methods: Instant Runoff and Closed Party List Voting. Based on the IR and CPL methods outlined in the project requirement specifications, the software will produce election results accurately and efficiently. It will also print the results of the election to the screen, and create an audit file with election information for auditors of the election.

## 1.5 References

In creating this document the team referenced the *Project1\_Waterfall\_VotingSRS\_Spring2023.docx* document for relevant information on voting methods and requirements. The team also utilized the *gephi\_srs\_document.pdf* document and the *SRSEExample-webapp.doc* document to use as an example for modeling this SRS. Each of these files were accessed through the course canvas page.

## 2. Overall Description

### 2.1 Product Perspective

The voting system is a new, self-contained product that determines the results of an Instant Runoff Election or a Closed Party Election. The system has been created to provide efficient and accurate election results from a ballot file and creates an audit file for transparency. It should be used after all votes have been cast and after those ballots are compiled into a single CSV file. The diagram below provides a basic overview of the voting system.

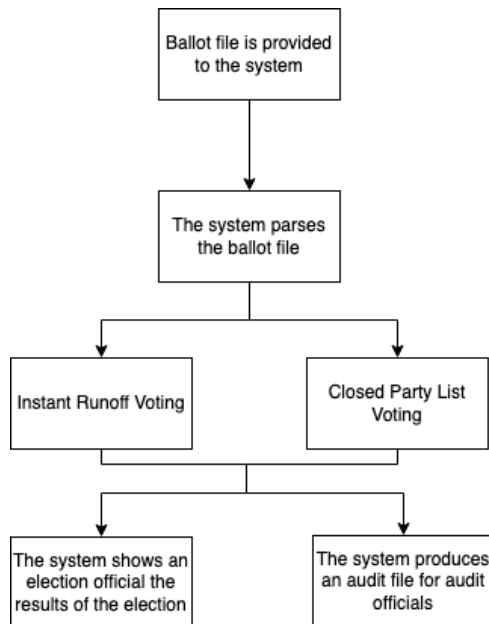


Figure 2.1: Voting System Overview Diagram

### 2.2 Product Functions

The major functions the product must perform or let the user perform are listed below.

- Starting the System – The user will start the system to be able to process the ballot file and determine the election winner.
- Ballot File Input & Identification – There are two ways for the ballot file to be inputted into the system:
  - 1: While the user starts the system, they will provide the name of the ballot file as a command line argument along with the command to start the system
  - 2: If no ballot file name was entered as a command line argument, the user starts the system with a command and the user is then prompted to enter the name of the ballot file
- Determining the Election Type – The type of election is determined by the first line of the input ballot file, so the system will read the first line of this ballot file and determine whether the election is an Instant Runoff Election (IR) or a Closed Party Election (CPL).
- Creating the Audit File – An audit file is automatically created when the program is run which creates transparency in the election results.
- Naming the Audit File – Audit files are created with a certain naming convention. The names begin with the type of election (IR or CPL), then they have an underscore, and they end with the date in a mmddyyyy format.
  - For example:

- IR\_02152023
- CPL\_02152023
- Ballot File Processing – The system will process the rest of the ballot file (after the first line) to gain information about the candidates and/or parties, and the number of votes to determine a winner
- Calculating a Winner in an Instant Runoff Election – The system operates as a series of runoff elections until one candidate gets a majority of the votes (over 50% of the votes). If there is not a majority, then popularity wins after all votes have been handed out.
- Calculating a Winner in a Closed Party List Election – Ballots have a ranked list of candidates for each party. Voters cast their vote for a party as a whole and the votes are proportionally distributed to the parties. The number of seats won by a party determines how many candidates from that party are elected in their ranked order.
- Calculating the Winner in a Tie – In both IR and CPL elections, ties can occur between candidates and parties. The system will perform a fair coin toss to determine the winner.
- Display the Winner(s) to the Screen – The system will display the winner, and other information about the election such as the type of election, number of seats won for a party in a CPL election, vote percentages, and other information relevant to the election.
- System Shutdown During Execution – The system can be stopped while it is running with a graceful reset to the main menu. Before the system stops, it will ask the actor to confirm if they would like to stop the system.

## 2.3 User Classes and Characteristics

The most important users of the system are programmers and testers. They ensure the system functions correctly and provides accurate election information. Without accurate results, nobody will be able to use the system. Programmers and testers may provide the system with many ballot files in both Instant Runoff Elections and Closed Party List Elections to ensure accuracy and efficiency is maintained throughout the system. These users have access to the full system as they are the ones who have made it.

A group of users that will use the system most often and for this reason are seen as very important users are election officials. They conduct elections to ensure voting is handled properly, and they are to provide the system with a ballot file each time they want to count the votes from an election. Election officials use the system to provide it with a ballot file, but should not be able to access or use the system in any other way.

Audit officials check to make sure the system is running properly and producing accurate election results after the system has processed a ballot file for an election. The system provides an audit file after an election official has conducted a vote count, and auditors look at this file to see how the votes were counted. These users are important to the system to ensure transparency is provided in elections. Audit officials should be able to access and read the audit file, but should not interact with the system in any other way.

Media personnel report the election results to the public. This group of users are not as important as programmers, testers, election officials, or audit officials, and they should not be able to directly interact with the system.

## 2.4 Operating Environment

The software is written in C/C++ for development and will be compiled with gcc (Ubuntu 9.4.0-lubntul~20.04.1) 9.4.0. It is recommended that all users run the system on CSE Lab machines, which use Ubuntu (Linux) 20.04.5.

## 2.5 Design and Implementation Constraints

Developers of the voting system will expect that the system works accurately and efficiently on CSE Lab machines. The system will be written in C/C++ as stated in section 2.4, and developers can also expect that the users of the system will be proficient in English. A timing requirement for the system is that it should be able to run 100,000 ballots in under four minutes. Users should not be able to write to any file produced by the system. Audit officials should be able to read the audit files produced by the system, and only election officials should have access to view the results of the election. No other users other than programmers and testers should have access to the system. Lastly, the customer's organization will be responsible for maintaining the system after it is developed.

## 2.6 User Documentation

The voting system will have a README file that will contain detailed instructions on how to use the system. It will provide instructions for an election official on how to run the system, as well as instructions for an audit official to view the audit file the system produces.

## 2.7 Assumptions and Dependencies

It is assumed that the provided CSV ballot file will have no mistakes or errors in the ballots. It is also assumed that the most up-to-date CSE Lab machines will be used to run the voting system. For Instant Runoff Elections, it is assumed that if there is a ballot in the file, it will have at least one of the candidates ranked. Election officials should have proficient knowledge of directory navigation in Linux and should know how to run programs from the command line with command line arguments. Lastly, it is assumed that only authorized users have access to the system which includes programmers, testers, election officials, and audit officials.

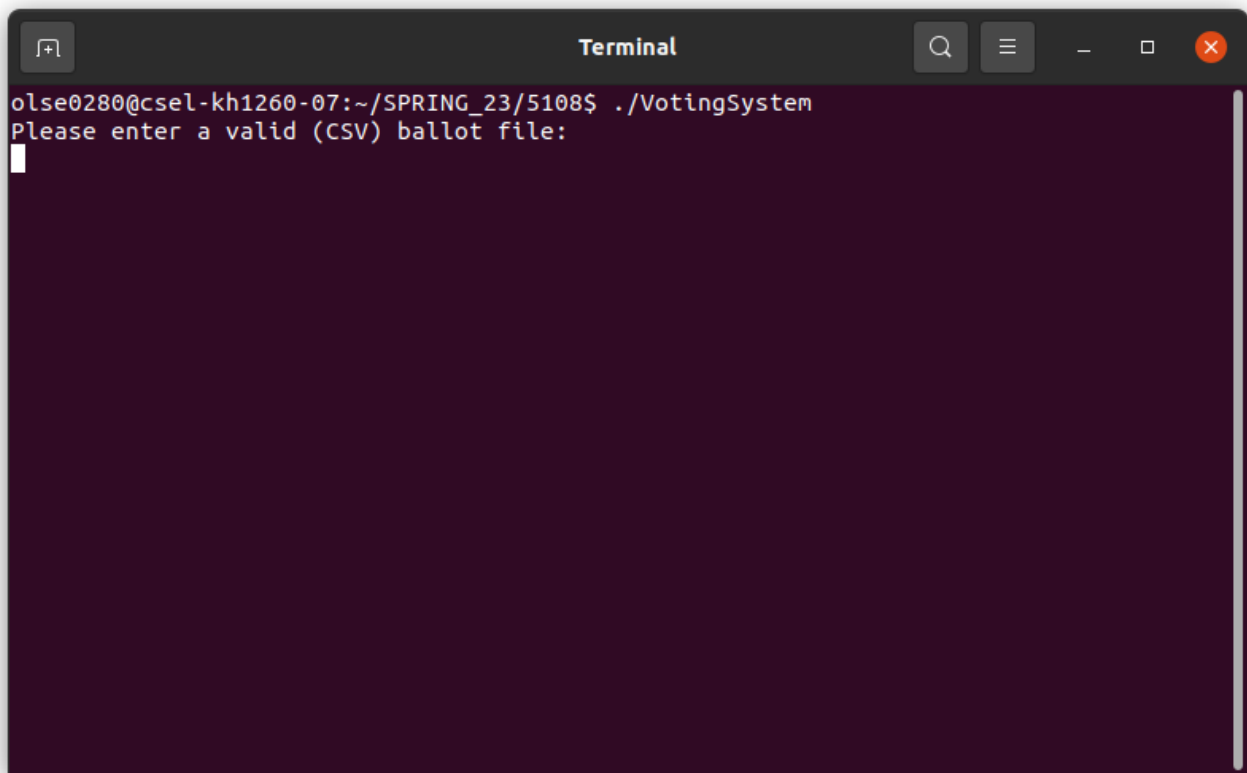
# 3. External Interface Requirements

## 3.1 User Interfaces

To begin, the user can start the voting system in two ways. First, the user can run the system from the command line in the terminal without any command line arguments. After the system has started without any command line arguments, the system will provide a simple prompt to the user to enter a valid ballot file. Figure 3.1.1 below shows an example of this prompt. The second way to run the system is with the ballot file as a command line argument. Figure 3.1.2 shows an example of starting the system, in this example *VotingSystem*, with a ballot file *BallotFileName.csv*.

After the system starts and has been provided with a valid ballot file, it will process the ballot file to determine the results of the election. Once the results have been processed and determined, the system will output those results to the screen for an election official to see.

To stop the system while it is running, a keyboard shortcut can be used. This shortcut is control+c (^c). This will stop the system from running and will gracefully return to the main menu.

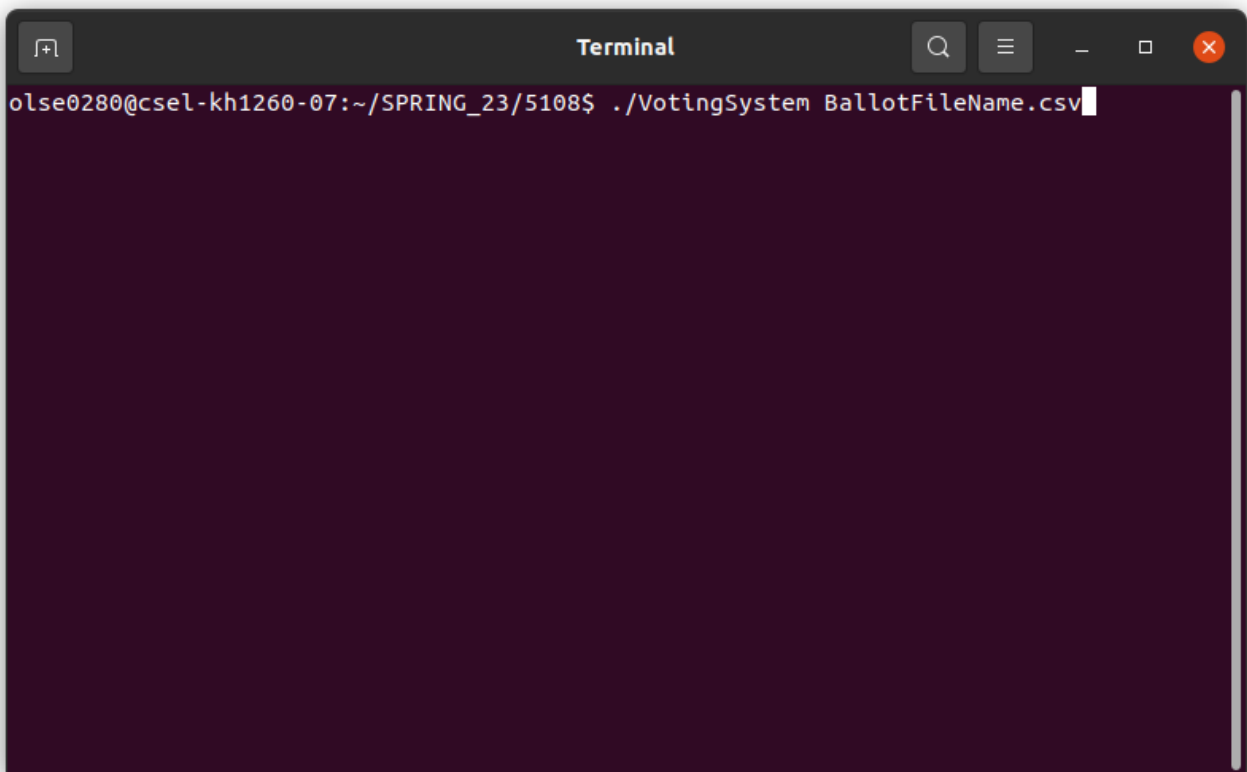


A terminal window titled "Terminal" with a dark background. The prompt is `olse0280@cse1-kh1260-07:~/SPRING_23/5108$`. The command `./VotingSystem` has been executed, and the output is `Please enter a valid (CSV) ballot file:`. A white cursor is positioned on the line following the prompt.

```
olse0280@cse1-kh1260-07:~/SPRING_23/5108$ ./VotingSystem
Please enter a valid (CSV) ballot file:

```

Figure 3.1.1: Enter a Valid Ballot File Prompt



A terminal window titled "Terminal" with a dark background. The prompt is `olse0280@cse1-kh1260-07:~/SPRING_23/5108$`. The command `./VotingSystem BallotFileName.csv` has been entered, and a white cursor is at the end of the command.

```
olse0280@cse1-kh1260-07:~/SPRING_23/5108$ ./VotingSystem BallotFileName.csv

```

Figure 3.1.2: Run with Ballot File as Argument

## 3.2 Hardware Interfaces

The Voting System's minimum requirements are an 8-core Intel Core i7 @ 2.5 Ghz and at least 8 GB of RAM. A system with these specifications should be able to support an election to count 100,000 votes in under 4 minutes. Lower-end Linux machines are likely to support the program's functionality; however, the performance threshold of 100,000 votes in under 4 minutes is not guaranteed.

## 3.3 Software Interfaces

The voting system has been compiled for use on Linux systems (Ubuntu 20.04). A command terminal is necessary to execute the voting system. The system must be able to store CSV files with reading privileges and create .txt files with writing privileges.

## 3.4 Communications Interfaces

No connection is required to operate the program. Depending on how the ballot file is received by the system, a connection may be required, but receiving this file is not a concern of the voting system itself.

# 4. System Features

Please refer to the file *UseCases\_Team7.pdf* for the use cases that are associated with the voting system's features.

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

The program is likely to work on a variety of low-end Linux machines; however, its execution and performance will have only been officially verified on the CSE Labs machines. The runtime requirement for the voting system is to process 100,000 in under four minutes.

## 5.2 Safety Requirements

Ballot files stored in the program's operating directory are the most important pieces of data relevant to its operation and security. When executing, the program will need to open ballot files for analysis, and it will never open these files in any mode other than read mode. Additionally, when the program needs to create a file (such as the audit file) it will make sure to not overwrite or modify any existing files in the current directory.



## 5.3 Security Requirements

There are no security measures in place for the program to operate; however, it is intended for only election officials to use such that confidential election information is not disclosed to the public. Ballot data interpreted by the program will not be disclosed by the program. Instead, the ballot data will be compiled into a summary in addition to the election process stored in an audit file to not unnecessarily disclose individual votes. Election officials may then proceed to publicize the results of the election as uncompromisingly determined by the system. Similar to the safety requirements, ballot file data will only be opened with read permissions to avoid changing original ballot data. Any other security requirements will be handled by the voting center where the election takes place.

## 5.4 Software Quality Attributes

The program is designed to be easy to start up and execute for those with some knowledge of terminal operations while still providing robust election computation abilities. Methods used by the program to declare election winners shall be finely documented within an audit file corresponding to each election to assure full transparency and security in the results of an election.

## 5.5 Business Rules

The system shall follow specific guidelines regarding the implementation of the IR and CPL voting processes set by the project specification and FairVote.org. These rules are in place to keep the election process fair and transparent while also implementing the valid voting systems requested by the project description. Additionally, while it is not enforced in the code, the program and its output is intended only for certified election officials to use and view. No other parties shall be allowed to use this program and its output to maintain a secure, fair, and unbiased election result.

# 6. Other Requirements

There are no further requirements not already specified in this document.

## Appendix A: Glossary

SRS: Software Requirements Specification

IR: Instant Runoff Voting

CPL: Closed Party List Voting

## Appendix B: Analysis Models

Figure 2.1: Voting System Overview Diagram

Figure 3.1.1: Enter a Valid Ballot File Prompt

Figure 3.1.2: Run with Ballot File as Argument