Software Requirements Specification

for

Voting System

Version 1.0

Prepared by Team 24

Michael Vang (vang2891), Matin Horri (horri031), Shivali Mukherji (mukhe105), Wenjing Jiang (jian0508)

University of Minnesota, Twin Cities

Feburary 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Document created	1/22/23		1.0a
Public	2/16/23	Initial release	1.0

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of a voting system capable of handling two types of elections: an instant run-off and a closed-party voting system. The document will explain the purpose and features of the software, what the software will do, and the constraints under which it must be operated. This document is intended for users of the software and potential developers.

1.2 Document Conventions

The document was created in guidelines with the IEEE template for System Requirements Specification Documents.

1.3 Intended Audience and Reading Suggestions

- Election officials who want to use the software to effectively process ballots.
- Testers who will test the software
- Programmers who are interested in expanding the software and adapting new voting systems

1.4 Product Scope

The voting system is a tool to effectively process many ballot files during an election and determine a candidate as a result. A CSV file will be entered containing the type of election,

candidates, and individual ballots, then it will process by the software and an audit file will be produced.

1.5 References

IEEE Template for System Requirement Specification Documents:

https://goo.gl/nsUFwy

Voting System Github:

https://github.umn.edu/umn-csci-5801-01-S23/repo-Team24

2. Overall Description

2.1 Product Perspective

The software was developed with election officials in mind for an efficient process of ballots pertaining to a certain voting system. The software is designed to process and analyze CSV files provided by the user with the ability to provide additional information.

The software is tasked to handle two voting systems that occur: the Instant Runoff Voting (IRV) system and the Closed Party Voting System (CPV). The way to inform the software of the type is through the first line of the CSV file. Once the software successfully processes the CSV file, the results of the voting system will be displayed and an audit file in the txt format of all the steps taken during the software will be produced, and the software is developed to run on Linux.

2.2 Product Functions

Terminal Interface:

- Opening of a CSV format file either through the command line or through the terminal interface.
- Prompt the user for additional information
- Show the results of the election process of the voting system to the terminal user. This is automatically done after an election file has been processed.

Voting System

A voting system will be determined from the first line of the CSV format file. The following voting systems are supported along with their abbreviations in the CSV file. The software will handle the uniqueness of the voting system as well. For example, redistribution of ballots in instant runoff or seat distribution in closed party listing. See section 2.6 for more information on the two voting systems supported here.

Instant Runoff (IR):

If an instant runoff has been declared, the system will parse the next 3 lines to gather the following information, sequentially: number of candidates, candidates separated by commas, and number of ballots in the file. An example CSV file generated for Instant Runoff voting:

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

Closed Party Listing (CPL):

If a closed party listing has been declared, the system will parse the first 2 lines to gather the number of parties and name of candidates. Immediately following will list the candidates in ranked order. After all the candidate's lines, it will be the number of seats. Then, the number of ballots. An example CSV file generated for Closed Party Listing voting:

```
CPL
6
Democratic, Republican, New Wave, Reform, Green, Independent
Foster, Volz, Pike
Green, Xu, Wang
Jacks, Rosen
McClure, Berg
Zheng, Melvin
Peters
3
9
1,,,,,
1,,,,,
,1,,,,
,,,,1,
,,,,,1
,,,1,,
,,,1,,
1,,,,,
,1,,,,
```

Tie Breaker

In the case a situational tie has occurred during an election, ie. a tie between seats, the software will do a fair coin flip as a tie-breaker.

Audit File

Every time the software successfully processes a CSV file, an audit TXT file will be generated that includes the documentation of the software's activity with the CSV file. See section 4 of use cases pertaining to the creation of an audit file for a more detailed of what the audit file is documenting.

2.3 User Classes and Characteristics

- Election officials who want to use the software to efficiently and effectively conduct ballots for a type of voting system (instant runoff and closed party).
- Programmers who are interested in further implementing additional voting systems for the system.
- Testers who are looking to debug the system.

2.4 Operating Environment

The program will run on the CSE LABs that are running Ubuntu 20.04.5 LTS. The program is coded in C++ and will be compiled by the GCC compiler 9.4.0. Makefiles are provided and can be performed through the terminal via the /make command.

2.5 Design and Implementation Constraints

The Voting System is developed in C++. The interaction with the Voting System is through the terminal and can only open a formatted CSV file. It can only support two election types: the instant run-off and the closed-party listing. In the case, a situational tie has occurred during an election, ie. a tie between seats, the software will do a fair coin flip as a tie-breaker. As for the audit file, it will be produced as a .txt file and will only have readable permission. The system can only read a single CSV file at a time and cannot process two files simultaneously.

‘

2.6 User Documentation

There is a quick start guide available on the website:

https://github.umn.edu/umn-csci-5801-01-S23/repo-Team24

For a description of closed party listing:

https://www.washoecounty.gov/voters/old-site/elections/closedprimarystate.php

For a description of instant runoff:

https://vote.minneapolismn.gov/ranked-choice-voting/details/

2.7 Assumptions and Dependencies

To work with the voting system and count the votes, a correctly formatted CSV file must be opened. The system does not open any other file. After the vote is counted, the system will produce an audit file in a .txt file with only readable permissions that displays the election results. The software is developed in C++ compiled by GCC version 9.4.0. The system can only work with one CSV file at a time, and can not work with more than one file at once.

3. External Interface Requirements

3.1 User Interfaces

The command-line user interface is used in the voting system. It requires users to type appropriate instructions into the command line. Keep in mind that report_election and write_file are automatically performed after an election CSV file has been processed. The commands are there for

1. Welcome information

   When the software is launched, show the welcome information which includes the name, version of the software, and other information for initializing: The following is an example of how the welcome screen will look:

   ```
   Voting System Software v 1.0.0

   To view all the commands, type in "help"
   The following command will return details of how to use the command "cmd": type in "man <cmd>".
   To load in a CSV file for process with or without .csv, type in "read_file <filename>"
   ```

2. Help

   "**help**" command returns all commands available. "**help** *keyword*" returns the commands containing "keyword".

3. Manual of commands

   "**man** cmd" command returns the details of how to use the command "cmd".

4. Read csv file

   "**read_file** filename" command reads a csv format file.

5. Report election

   "**report_election**" command returns the information of the election including the type of election, number of seats, the candidates, the total votes and the winner. Use option "-attribute" to report the specified information.

6. Write audit file

   "**write_file**" command writes a txt format audit file.

3.2 Hardware Interfaces

The suggested hardware requirements of the voting system can run on a 2GHz CPU and a 8GHz RAM. To give a sense, the hardware will need to be able to handle running a terminal window, file I/O such as reading and writing, and the ability to process terminal text.

3.3 Software Interfaces

Voting system requires GCC version 9.4 to be installed on the system. C++ standard library is used to design the voting system, and Readline library allows users to edit the command lines as they are typed in.

3.4 Communications Interfaces

The voting system will require an internet connection to install, update, and additional add-ons. HTTPS communication standards will be used.

4. System Features

All system features are listed in use cases. All use cases are provided through the GitHub repository which is accessible in section 1.5 or through this link here: https://github.umn.edu/umn-csci-5801-01-S23/repo-Team24/tree/main/SRS/

All use cases will be in one pdf or can be accessed individually. Use cases will describe the features in detail.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The voting system will only process 1 CSV file. In addition, the system must be capable of processing at least 100,000 ballots in under 4 minutes.

5.2 Safety Requirements

Assume that the system does not have any issues with safety, and its performance is faultless.

5.3 Security Requirements

Assume security is top-notch, and there are no issues with people's personal information getting leaked. Security such as ensuring one vote for one person will be handled at the voting centers

5.4 Software Quality Attributes

The voting system's usability will be simple for election officials to navigate due to the terminal’s limited commands and simple usage.

As for developers that are looking to integrate additional voting systems, the voting system will use a strategy design pattern to easily interchangeable between voting systems and seamlessly integrate new voting systems into the software.

5.5 Business Rules

There are no business rules.