Software Requirements Specification

for

Voting System

Version 1.0.1 approved

Prepared by Chork Hieng (hieng001)
Ziyoda Mamatkulova (mamat007)
Andrei Anicescu (anice002)
Mohamed Ali(ali00429)

Team 7

February 2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Voting System	9 Feb, 2024	Initial release	1.0
Voting System	12 Feb. 2024	Style updates	1.0.1

1. Introduction

1.1 Purpose

This document describes the features and requirements of the voting systems used by election officials for processing voting ballots in a OPL or CPL election type. The documentation includes functional and non-functional requirements, and systems behavior.

1.2 Document Conventions

This document is created based on IEEE standards requirements specification.

1.3 Intended Audience and Reading Suggestions

- This system is intended for election officials who have the responsibility of processing the voting ballots and vote count for the election.
- Testers who want to ensure that the system behaves as expected.

1.4 Product Scope

This system is intended to be used by election officials for processing votes and providing unbiased results for the election. The system serves as the fast-forwarding software in order to ease the data processing. The user(s) can interact with the system through GUI interface and command line by inputting the file name for each election.

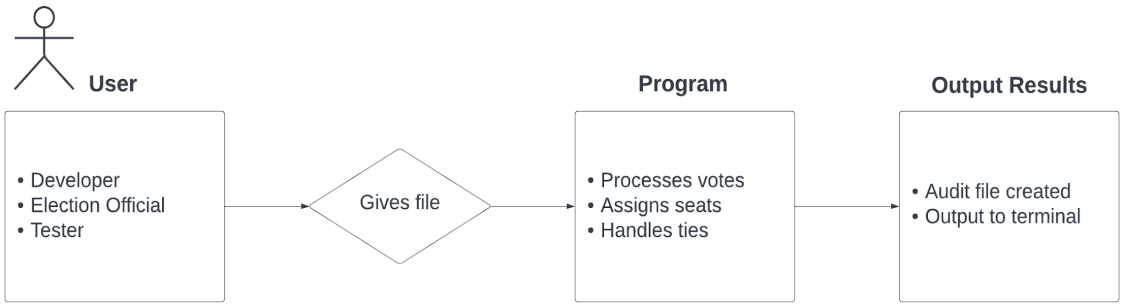
1.5 References

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

2. Overall Description

2.1 Product Perspective

This voting system was created to process ballot counts for OPL and CPL elections. It is designed to accurately assign seats to candidates based on popularity, and to fairly handle ties. This product is self-contained, and the first version. There is only one entry point with the system, which is to provide it with a file.



2.2 Product Functions

- The user (election officials, tester) will be given a prompt to type in a file name
- The program checks if a file with such a name exists in the same directory, if it does, it will read it
- The program determines if the CPL or OPL algorithm will be used
- The program will handle any ties
- The program will display results to the terminal
- The program will create an audit file

2.3 User Classes and Characteristics

The election official and tester should be allowed to use this program. All users receive the same prompt, and the same output (audit file, results on terminal). Users should know how to use a computer (know to type), and know to spell the name of the ballot file correctly. Members of the media can observe the results of the election after all votes have been processed.

2.4 Operating Environment

The program must be able to run on a University of Minnesota - Twin Cities campus CSE machine. This means it must run on the Linux operating system. The program will be written on C++, and the necessary compilers should be installed before running.

2.5 Design and Implementation Constraints

- The programming language must be OOP
- The program must run 100,000 ballots in less than 4 minutes
- The program only allowed to read one file at a time

2.6 User Documentation

A comprehensive guide covering the step-by-step process of running the software on CSE lab machines will be provided in a Readme.txt that will be in plain text format. This file will be in the same folder as the voting system. That way it will be accessible and readable on any computing platform without the need for specialized software.

2.7 Assumptions and Dependencies

The input ballots are assumed to be error-free and formatted properly. It is assumed all standard C++ library functions used in the software are supported across target platforms. The program is assumed to be run only once per election to ensure fairness.

3. External Interface Requirements

3.1 User Interfaces

The users will interact with the product from a terminal or command line interface. They must correctly type the name of the file they want to run when they are prompted.

3.2 Hardware Interfaces

The system must be fully compatible with the hardware of the CSE lab machines, to make sure it can run effectively without any compatibility issues.

3.3 Software Interfaces

The system will be compatible with the operating systems in CSE lab machines such as Linux. The software will share output results as standard text files after processing election results from the given ballots file.

3.4 Communications Interfaces

The voting system will operate offline. The software will exchange data by reading ballots and writing the results.

4. System Features

4.1 One Program to Handle Everything (one entry point)

4.1.1 Description and Priority

The election counting program will handle the counting of votes, the allocation of seats, and the handling of ties. It will provide an audit file at the end, and print results on the terminal. This is a high priority, as the program has only one entry point.

4.1.2 Stimulus/Response Sequences

The user can start the program and provide a file name as a command line argument, or they can start the program and the program will prompt the user to type a file name through the terminal. The file must exist in the same directory as the program, else the program will have the user re-enter the file name.

4.1.3 Functional Requirements

Name	One program to handle everything
ID	UC_001
Actors	System
Organizational Benefits	Simplifies election processing, and ensures there is no bias.
Frequency of Use	Once per election.
Triggers	Program is run by the user.
Preconditions	File exists in the same folder as the program.
Postconditions	Program runs and produces an audit file and outputs to the terminal.
Main Course	Program reads the ballot file, process results.
Alternate Courses	N/A
Exceptions	N/A

4.2 Filename Prompt

4.2.1 Description and Priority

File can be given to the system through a prompt, or as a command line argument when the program is first started. The file must exist in the same directory as the program. In either of the cases, if the filename is not spelled correctly, the program will prompt the user to retype the filename on the terminal. This is a high priority because the system cannot guess the ballot file's name.

4.2.2 Stimulus/Response Sequences

The system checks whether the file name is correct before processing the file. If the user(s) provides the wrong file name or the file name does not exist, the system should prompt the user to retype the file name or select the correct file.

4.2.3 Functional Requirements

REQ-1: The program is run with the filename as an argument in the command line

Name	File name prompt in command line
ID	UC_002
Actors	User (election official, tester), system
Organizational Benefits	Correct election file is processed.
Frequency of Use	Once per election, run until the correct filename is provided.
Triggers	The file name given by the user exists in the same folder as the program.
Preconditions	Program can run and accept filenames as command-line arguments.
Postconditions	Confirm file exists.
Main Course	Program is started, it looks for the file with the given name in its current directory.
Alternate Courses	The program outputs to the terminal that it couldn't find the file, and prompts the user to type in the correct file name.
Exceptions	N/A

REQ-2: Program (is running, and) asks the user to type in a filename.

Name	File name prompt in terminal
ID	UC_003
Actors	User (election official, tester), system
Organizational Benefits	The correct election file is processed.
Frequency of Use	Once per election, run until the correct filename is provided.
Triggers	The file name given by the user exists in the same folder as the program.
Preconditions	The program is running.
Postconditions	Read permissions enabled for the ballot file.
Main Course	The system is running and asks the user for a filename. If the file exists, the program starts reading it.
Alternate Courses	Continuously ask the user for the filename, output that the requested file was not found in the current folder.
Exceptions	N/A

4.3 Extract Information from a File

4.3.1 Description and Priority

When the program is run, you need to extract all information needed to run the voting algorithm directly from the file (e.g. number of votes, number of candidates, etc). This feature is a high priority.

4.3.2 Stimulus/Response Sequences

The system reads everything in the file.

4.3.3 Functional Requirements

Name	Extract information from a file
ID	UC_004
Actors	System
Organizational Benefits	Unbiased and efficient vote counting and tie management.
Frequency of Use	Extract the file once per election. There may be multiple elections per year.
Triggers	The file name given by the user exists in the same folder as the program.
Preconditions	The file must exist in the folder.
Postconditions	Read permissions enabled for the ballot file.
Main Course	The system starts reading the file and keeps track of seat numbers and voting types used in the election.
Alternate Courses	N/A
Exceptions	N/A

4.4 Process File Header

4.4.1 Description and Priority

The program will read the file, and determine the election type, number of seats, and number of candidates. This is a high priority, as it determines which algorithm will run.

4.4.2 Stimulus/Response Sequences

When the existence of the file is confirmed and the file is read, the system will check the header of the file. The header specifies either the CPL (Closed Party List) or OPL (Open Party List) voting type.

4.4.3 Functional Requirements

Name	Process file header
ID	UC_005
Actors	System
Organizational Benefits	Proper calculations are done for seat allocation.
Frequency of Use	Once per election.
Triggers	The program can read the given file.
Preconditions	The file exists and is correctly formatted.
Postconditions	Choose an algorithm to run.
Main Course	The program reads the first few lines of the file, gathering three main pieces of information: election type, number of seats, and number of candidates.
Alternate Courses	N/A
Exceptions	N/A

4.5 CPL (Closed Party Listing)

4.5.1 Description and Priority

If “CPL” is the voting type used, then the program will count the votes accordingly. For CPL, each party arranges the order of their candidates. A voter can only cast a vote for the party. If a party wins one seat, then the first listed person of that party gets the seat. If the party wins two seats, then the second person is given the seat, and so on. This is a high-priority item, as the system must accurately give seats based on party popularity.

4.5.2 Stimulus/Response Sequences

At this stage, it’s assumed that the file is correctly formatted and the data required exists in the file. After reading through the entire file and counting the votes, the system gives this data to the CPL algorithms to calculate the seat assignments based on party popularity.

4.5.3 Functional Requirements

Name	CPL
ID	UC_006
Actors	System
Organizational Benefits	Appropriately assign candidates based on party popularity.
Frequency of Use	Once per election.
Triggers	The program reads that the election type is CPL.
Preconditions	The file is correctly formatted and the file’s header is read.
Postconditions	The header is read, and the CPL algorithm is given the ballot file.
Main Course	The program confirms the header is read, chooses the CPL algorithm, which processes the votes correctly and assigns seats.
Alternate Courses	N/A
Exceptions	N/A

4.6 OPL (Open Party Listing)

4.6.1 Description and Priority

If “OPL” is the voting type used, then the program will count the votes accordingly. For OPL, any candidate can be voted for. This vote counts for the candidate and the party. After votes are counted, the order of candidates in each party is sorted by most votes to lowest votes. Those with the most votes win seats, and the algorithm goes down the line of candidates with most votes until it runs out of seats. If a tie occurs, the algorithm will handle that fairly. This task is of high priority, as each candidate has their own votes with the party.

4.6.2 Stimulus/Response Sequences

At this stage, it’s assumed that the file is correctly formatted and the data required exists in the file. After reading through the entire file and counting the votes, the system gives this data to the OPL algorithms to calculate the seat assignments based on candidate and party popularity.

4.6.3 Functional Requirements

Name	OPL
ID	UC_007
Actors	System
Organizational Benefits	Appropriately assign candidates based on candidate popularity.
Frequency of Use	Once per election.
Triggers	The program reads that the election type is OPL.
Preconditions	The file is correctly formatted and the file’s header is read.
Postconditions	The header is read, and the OPL algorithm is given the ballot file.
Main Course	The program confirms the header is read, chooses the OPL algorithm, which processes the votes correctly and assigns seats.
Alternate Courses	N/A
Exceptions	N/A

4.7 Fair Process for Determining a Tie

4.7.1 Description and Priority

If a tie occurs, the program must randomly select a winner. This is a high priority because true randomness allows for fair elections.

4.7.2 Stimulus/Response Sequences

If there are two or more candidates with the same amount of votes for one seat, the system will randomly select a winning candidate.

4.7.3 Functional Requirements

Name	Fair tie
ID	UC_008
Actors	System
Organizational Benefits	A fair tie breakdown will make the system valuable in a fair election.
Frequency of Use	Once per election.
Triggers	Two or more candidates have the same number of remaining votes when there are one or two seats left.
Preconditions	Seats have been fairly assigned based on vote count popularity.
Postconditions	The system ensures that the chance of any candidate winning is 50-50 if a two-candidate tie, or ⅓ -⅓ -⅓ if there are three candidates, and so on.
Main Course	The program runs the random selection process thousands of times until the chances of all candidates are nearly the same.
Alternate Courses	N/A
Exceptions	N/A

4.8 Produce an Audit File

4.8.1 Description and Priority

After processing is done, an audit file should be created and saved in the same directory as the program. The audit file will contain the type of election (OPL, CPL), number of parties, number of ballots counted, number of seats, candidate's names and party affiliation. The amount of votes each candidate or party receives will also be shown in a table.

4.8.2 Stimulus/Response Sequences

The audit file can only be produced after the results have been determined by the program.

4.8.3 Functional Requirements

Name	Produce audit file
ID	UC_009
Actors	System
Organizational Benefits	An audit file shows the results of the election and other needed data.
Frequency of Use	Once per election.
Triggers	The program has counted and processed election votes.
Preconditions	All voting has been counted, and all seats have been assigned.
Postconditions	The program starts to create and write in a text file.
Main Course	The program writes the results of the election in a readable format in a text file.
Alternate Courses	N/A
Exceptions	N/A

4.9 Display to Terminal

4.9.1 Description and Priority

After all the files are processed and vote counts are determined, the result will be displayed to the terminal.

4.9.2 Stimulus/Response Sequences

The system determines whether all files are processed (including the chosen algorithm for each file, vote counts for each party, handling tie cases). The system then produces results for the election.

4.9.3 Functional Requirements

Name	Display to terminal
ID	UC_010
Actors	System
Organizational Benefits	Lets users easily see election results.
Frequency of Use	Once per election.
Triggers	The program has allocated all seats. It’s ready to announce results.
Preconditions	The program is done by computing the election results, and all seats have been assigned to a candidate.
Postconditions	The program has finished all its tasks.
Main Course	The program confirms the header is read, processing the algorithm correctly based on the file's header.
Alternate Courses	N/A
Exceptions	N/A

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The program must run 100,000 ballots in less than 4 minutes and correctly produce the results. The program should also be able to run on CSE Lab machines as mentioned above in section 2.4. The test files must be easily testable under various conditions. The system must be able to handle an increase in ballot numbers without lower performance.

5.2 Safety Requirements

The users will be screened through external procedures before being able to use the program to ensure data integrity.

5.3 Security Requirements

The voting system will be available only offline. Users are authorized to use the program by external policies. The input files will not be able to be modified in order to preserve data integrity.

5.4 Software Quality Attributes

The system will perform its intended functionalities under specified conditions without failure. The software will be designed for easy updates and maintenance such as being open to extension for write-in candidates. The system will adhere to OOP principles.

5.5. Business Rules

The system shall provide an interface for all users to view election results, ensuring transparency. However, only election officials can input election data.

Appendix A: Glossary

Audit file - text-based file to inspect with results

Command line - text-based interface where commands can be typed. These commands interact with the operating system of the computer

CPL - Closed Party Listing

OOP - Object-Oriented Programming

OPL - Open Party Listing