Software Requirements Specification

for

VoteEasy

Version 2.2 approved

Prepared by Jashwin Acharya (achar061), Steve Petzold (petzo017), Vincent Hoang (hoang317), and Carlos Chasi-Mejia (chasi009)

CSCI 5801

23 September 2023

Table of Contents

Table of Contents **ii**
Revision History **ii**
1. Introduction **1**
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 1
1.5 References 2
2. Overall Description **2**
2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Classes and Characteristics 2
2.4 Operating Environment 3
2.5 Design and Implementation Constraints 3
2.6 User Documentation 3
2.7 Assumptions and Dependencies 3
3. External Interface Requirements **3**
3.1 User Interfaces 3
3.2 Hardware Interfaces 3
3.3 Software Interfaces 4
3.4 Communications Interfaces 4
4. System Features **4**
4.1 Command Line Interface 4
4.2 Parse CSV File Header 4
4.3 Parse Ballots and Candidate Information 5
4.4 Produce Audit File 5
4.5 Majority wins when using IR Voting Protocol 5
4.6 Popularity wins when using IR Voting Protocol 6
4.7 Implement Open Party List Voting 6
4.8 Breaking a Tie 6
4.9 Display Winner and Election Information 7
5.0 Help Option 7
5. Other Nonfunctional Requirements **8**
5.1 Performance Requirements 8
5.2 Safety Requirements 8
5.3 Security Requirements 8
5.4 Software Quality Attributes 8
5.5 Business Rules 9
6. Other Requirements **9**
Appendix A: Glossary **9**
Appendix B: Analysis Models **9**
Appendix C: To Be Determined List **9**

Revision History

Name	Date	Reason For Changes	Version
VoteEasy	23 September 2023	Defined Section 1 details	1.0
VoteEasy	24 September 2023	Defined Section 2 details	1.1
VoteEasy	24 September 2023	Defined Section 5.4 & 5.5	1.2
VoteEasy	24 September 2023	Defined Section 5.1 and 5.3	1.3
VoteEasy	24 September 2023	Defined Section 3	1.4
VoteEasy	1st October 2023	Defined subsections 4.1, 4.2 and 4.3	1.5
VoteEasy	3rd October 2023	Improved wording for features 4.1, 4.2 and 4.3	1.6
VoteEasy	3rd October 2023	Defined information for Section 6	1.7
VoteEasy	3rd October 2023	Defined subsections 4.4 and 4.5	1.8
VoteEasy	3rd October 2023	Defined subsections 4.6 and 4.7	1.9
VoteEasy	4th October 2023	Defined subsections 4.8 and 4.9	2.0
VoteEasy	4th October 2023	Defined Sections 6	2.1
VotEasy	4th October 2023	Modifactions to sections 4,5,6	2.2

1. Introduction

1.1 Purpose

The purpose of this document is to provide a high-level design of ***VoteEasy***: a new Voting System that supports two types of voting protocols: Instant Runoff and Open Party List Voting. This document intends to explain the significance and features of the product, along with details about the software interface, design constraints, and the operating environment. This is Version 1.0 of the product.

1.2 Document Conventions

The document has been prepared according to the IEEE Software Requirements Specification Template.

1.3 Intended Audience and Reading Suggestions

This document has been designed for the following users:

- Software developers looking to understand the significance of the product and planning to work on improving existing functionality.
- Testers tasked with testing different software components to ensure a robust user experience.
- Election Officials looking to understand what features the product provides and providing feedback on what features can be improved to enhance user experience.

The rest of the SRS contains details of the specific features the product will provide, what constraints it will operate under, and how users can interact with the product. The right order of reading the SRS is as follows:

- Section 1 which introduces the product and describes its intended users.
- Section 2 describes the overall functionality and design constraints the product will operate under.
- Section 3 describes the user interface for using the product as well as how the product interfaces with external systems (if applicable).
- Section 4 provides detailed descriptions of each functional requirement that the product must provide to its users.
- Section 5 provides details of non-functional requirements for our overall system.

1.4 Product Scope

VoteEasy aims to provide election officials with the ability to automate ballot calculations during election season. Our system supports two types of voting protocols, namely - Instant Runoff (IR) Voting or Open Party List (OPL) Voting. The system provides a command line interface (CLI) that allows for an official to input a CSV file containing information about the voting protocol to be used and candidate details, along with the number of votes each candidate received. Our system performs the votes calculations, determines the winner, and provides an audit file providing a step-by-step progression of the election process.

Organizational benefits include saving valuable time by automating the ballot calculation process instead of spending time and money on manual calculations. The audit file generated by the system can also be used to debug any calculation errors and provide a transparent record of how the election progressed, which is necessary when sharing details of the election result with media personnel.

1.5 References

IEEE Software Requirements Specification Template
https://canvas.umn.edu/courses/391384/files/37845415?module_item_id=10914534

Use Case Template
https://canvas.umn.edu/courses/391384/files/37845475?module_item_id=10914542

2. Overall Description

2.1 Product Perspective

VoteEasy is a new tool that is designed to help election officials correctly calculate the result of an election given only an input file. The product is designed to be very simple to use and is expected to correctly determine the winner 100% of the time.

2.2 Product Functions

- A CLI interface allows the user to input a file name containing election information.
- Ability to parse the CSV file to retrieve voting protocol, candidate, and ballot information.
- Produce an audit file containing a step-by-step record of the election process.
- Handling the “Majority Wins” scenario for the Instant Runoff Voting Protocol.
- Handling the “Popularity Wins” scenario for the Instant Runoff Voting protocol.
- Implementing ballot calculations when using Open Party Listing as the voting protocol.
- Resolving ties whenever they occur.
- Displaying Winner Information.

2.3 User Classes and Characteristics

- Election Officials - This is the most important user group since they are the ones who have to declare a winner once VoteEasy determines the winner using ballot calculations.
- Software developers and Testers - Both user classes have the same level of importance since Software Engineers are responsible for building the product and improving existing functionality based off of the testing results that QA Teams and other Test teams provide them. Both user classes are responsible for building a robust system that correctly calculates the votes 100% of the time.

2.4 Operating Environment

- Windows 10/11
- Mac OS
- Linux

2.5 Design and Implementation Constraints

The product requires the latest version of Java SDK and JRE to run the voting program. Other than that, there are no other requirements or constraints that should prevent programmers, testers, or election officials from running the software.

2.6 User Documentation

The product comes with a README file which contains general instructions on how to run the Java program file(s) and how to run associated JUnit tests on a local machine.

2.7 Assumptions and Dependencies

The product does not use or rely on third-party software for functionality. We assume that the user has the latest version of Java SDK and JRE installed on their system prior to running the application. This is a completely new product that does not utilize any existing functionality from a previous project.

3. External Interface Requirements

3.1 User Interfaces

A Command Line Interface will be defined with a text prompt that allows users to provide input in the form of a CSV (Comma-Separated Values) file containing voting protocol and candidate details. The interface will provide clear and informative error messages in case of invalid inputs, errors during processing, or system issues. Error messages will guide users on how to correct their input or handle the situation. A “help” option will be provided to users to understand what file types are accepted by the system. Once a winner is determined by the program, we will display winner information such as Candidate Name, Party Affiliation, Percentage of Votes won, and details of other candidates and their votes.

3.2 Hardware Interfaces

Regarding Hardware interfaces, we are assuming the software will be run on the CSELabs machines that are running x86 and x64 bit architectures. These computers are also running Linux which should be capable of running our program without any issues. We are also assuming the computers will have proper peripherals including a mouse, keyboard, and monitor that will serve as the main tools for the user to complete their needed tasks.

3.3 Software Interfaces

For software interfaces, we want to make sure that we have the most up-to-date version of Java that can be run on the CSE labs machines as per the requirements. Notably, our application does not integrate any databases. Instead, files will be stored within the program's current directory on the user's computer for seamless access and management.

3.4 Communications Interfaces

We assume the user has received the file via email, or another form of document sharing. However, our application does not utilize any network server communications protocols when performing ballot calculations and determining the winner of the election.

4. System Features

4.1 Command Line Interface

4.1.1 Description and Priority

The Command Line Interface should facilitate the user to enter the name of the CSV file that contains the voting protocol and ballot information. This is a high-priority item since it is a fundamental feature of the product that is necessary for eventually performing vote calculations and determining a winner. This feature will be used multiple times during the year at normal election times and special elections.

Benefit: 9, Risk: 1, Cost: 1, Penalty: 9

*Refer to **Use Case ID #1** on Page 1 of the UseCases_Team08.pdf file for further details on Organizational Benefits, Triggers, Frequency of Use, Preconditions, Postconditions, Main Course, and Alternate Course.*

4.2 Parse CSV File Header

4.2.1 Description and Priority

Parse the CSV file header to determine the type of voting protocol to be used for ballot calculations. This is a high-priority feature since it helps determine how the ballot calculations should be performed to determine a winner.

Benefit: 9, Risk: 1, Cost: 1, Penalty: 9

*Refer to **Use Case ID #2** on Pages 1-2 of the UseCases_Team08.pdf file for further details on Organizational Benefits, Triggers, Frequency of Use, Preconditions, Postconditions, Main Course, and Alternate Course.*

4.3 Parse Ballots and Candidate Information

4.3.1 Description and Priority

Parse each line of the CSV file after the header is processed to determine the number of candidates and how many ballots each received. This is a high priority feature since we need to determine candidate information, calculate how many ballots they received, and optionally, determine the number of seats available (if we use Open Party List only) so that our voting protocol can correctly judge the winner of the election.

Benefit: 9, Risk: 1, Cost: 1, Penalty: 9

Refer to ***Use Case ID #3*** *on Pages 2-3 of the UseCases_Team08.pdf file for further details on Organizational Benefits, Triggers, Frequency of Use, Preconditions, Postconditions, Main Course, and Alternate Course.*

4.4 Produce Audit File

4.4.1 Description and Priority

The system produces an audit file that shows the progress of the election and contains all election information such as voting protocol name, number of candidates, ballot information, number of seats, winner information etc. This is a high-priority feature because the audit file provides a transparent record of how the voting progressed and can be used to detect and fix any errors that might have occurred in the vote calculation process. The results of the election can be shared with media personnel too.

Benefit: 9, Risk: 1, Cost: 1, Penalty: 9

Refer to ***Use Case ID #4*** *on Pages 3-4 of the UseCases_Team08.pdf file for further details on Organizational Benefits, Triggers, Frequency of Use, Preconditions, Postconditions, Main Course, and Alternate Course.*

4.5 Majority wins when using IR Voting Protocol

4.5.1 Description and Priority

The IR voting protocol must determine a winner if they receive a majority (greater than 50% of the total votes). This is a high-priority feature because the calculations are automated so the system should determine the candidate that has more than 50% of the total votes has won the election and the audit file should display the correct winner information for the IR voting protocol. This would also reduce manual labor.

Benefit: 9, Risk: 1, Cost: 1, Penalty: 9

Refer to ***Use Case ID #5*** *on Pages 5-6 of UseCases_Team08.pdf file for further details on Organizational Benefits, Triggers, Frequency of Use, Preconditions, Postconditions, Main Course, and Alternate Course.*

4.6 Popularity wins when using IR Voting Protocol

4.6.1 Description and Priority

The IR voting protocol must determine a winner if a clear majority is not achieved in the first round of ballot calculations. This is a high-priority feature since it will be used by Election Officials to declare a winner for the election and the system must be able to handle a scenario where a clear majority is not achieved in the first round of ballot calculations. This is an essential feature for Programmers to implement and for Testers to test thoroughly to ensure that votes are correctly calculated and distributed 100% of the time.

Benefit: 9, Risk: 1, Cost: 1, Penalty: 9

*Refer to **Use Case ID #6** on Pages 6-7 of UseCases_Team08.pdf file for further details on Organizational Benefits, Triggers, Frequency of Use, Preconditions, Postconditions, Main Course, and Alternate Course.*

4.7 Implement Open Party List Voting Protocol

4.7.1 Description and Priority

The OPL voting protocol will determine a winner based on a calculated quota. The winner is determined by checking which party has the highest percentage of seats. This is a high priority feature since Election Officials will be using the system to determine a winner and automating ballot calculations helps reduce manual labor.

Benefit: 9, Risk: 1, Cost: 1, Penalty: 9

*Refer to **Use Case ID #7** on pages 8-9 of the UseCases_Team08.pdf file for further details on Organizational Benefits, Triggers, Frequency of Use, Preconditions, Postconditions, Main Course, and Alternate Course.*

4.8 Breaking a Tie

4.8.1 Description and Priority

Regardless of the voting protocol being used, in the event of a tie between 2 or more candidates at the end, a winner is determined by a fair coin toss. This is a high-priority feature because the calculations are automated, so the system should produce a fair winner even in the event of a tie after all the ballots calculations have been performed.

Benefit: 9, Risk: 1, Cost: 1, Penalty: 9

*Refer to **Use Case ID #8** on pages 9-10 of the UseCases_Team08.pdf file for further details on Organizational Benefits, Triggers, Frequency of Use, Preconditions, Postconditions, Main Course, and Alternate Course.*

4.9 Display Winner and Election Information

4.9.1 Description and Priority

Once ballot calculation has been processed, and a winner has been produced, the system will display on screen election information such as the winner, type of election, number of seats, number of ballots cast, as well as a statistical breakdown of how many votes each candidate received. This is a high-priority feature because it allows users to test the system to make sure a correct outcome has been achieved and offers a comprehensible overview of the election.

Benefit: 9, Risk: 1, Cost: 1, Penalty: 9

*Refer to **Use Case ID #9** on pages 10-11 of the UseCases_Team08.pdf file for further details on Organizational Benefits, Triggers, Frequency of Use, Preconditions, Postconditions, Main Course, and Alternate Course.*

4.10 Help Option

4.10.1 Description and Priority

The CLI provides a text prompt where a user can enter “help” (not case sensitive) in order to receive instructions on how to use the system and details about what file type(s) are supported.

Benefit: 7, Risk: 1, Cost: 1, Penalty: 3

4.10.2 Stimulus/Response Sequences

The User enters “help” (not case sensitive) as the text prompt.

4.10.3 Functional Requirements

1. The only requirement is that the CLI should’ve been developed and should take text input.
2. The User is provided with information about supported file types (in our case, just CSV is valid) when entering the “help” option.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The voting system program written in Java is required to be executable and run on CSE lab machines. We will be providing all source files and class files that can be executed from the command line or using an IDE such as Eclipse. This is to ensure our product is graded without any technical issues.

- To run the program, the election input file has to be in the same directory as the project files. This is to ensure that our product is able to detect the file and easily parse its information to determine a winner.
- Performance depends on the file size, but it has to be able to run 100,000 ballots in under 8 minutes. We require fast results to accurately determine the winner of an election.
- Only one program is being written to handle both election types. Users only have to provide a CSV file, and will not be required to enter what voting protocol is to be used. This is to ensure that our program is easy to use for our different user classes.

5.2 Safety Requirements

- It’s possible that the input CSV file gets corrupted once it is received from an electronic source. In such an event, the user must retrieve a new copy of the file from the original sender of the file.
- The audit file can produce a file write error when election results are being written into it once a winner is determined. In this situation, it is ideal to re-run the program from scratch.
- In order to prevent any kind of data loss, do not try to force exit out of the program while information is being processed from the CSV file or written into the audit file.
- There’s a significantly low chance that the IDE being used for development could crash due to issues outside of our control, so please restart the IDE whenever that happens.

5.3 Security Requirements

The voting system program has the precondition that the software is only run by programmers, testers, and electoral officials. Additionally, to ensure that the election file is not tampered with and causes election fraud, the input file is read-only.

5.4 Software Quality Attributes

- The software product will correctly count the number of votes 100% for each candidate every time it is run.
- The software provides users with an easy way to use the software by only requiring the user to input a filename to count the votes.
- The software design must follow the Waterfall model of development since we want to be able to maintain this system and evolve it when necessary without increasing tech debt.
- Each software requirement has been designed in a way to allow it to be tested using either unit tests or manual integration tests.

5.5 Business Rules

The only users who can run this software product are programmers, testers, and election officials. The aforementioned users have access to every feature of the product.

6. Other Requirements

Appendix A: Glossary

For the purpose of readability, the following are the full descriptions of acronyms mentioned in this document:

1. IR - Instant Runoff
2. OPL - Open Party List
3. CLI - Command Line Interface

Appendix B: Analysis Models

TBD

Appendix C: To Be Determined List

1. UML diagram to be added at a later date