**Software Requirements Specification**

**for**

**Voting System**

**Version 1.0 approved**

**Prepared by Adam Wall, Christopher Hanson,**

**Gabriel Jardio, Patrick Perez**

**<organization>**

**<date created>**

***Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.***

***Software Requirements Specification for Voting System Page***

**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 1

**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 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

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 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

***Software Requirements Specification for Droop Voting System Page***

**Introduction**

**Purpose**

The purpose of this document is to present a detailed description of the software Voting System. It will explain the purpose and features of the software, the interfaces of the software, what the software will do and the constraints under which it must operate. This document is intended for users of the software and also potential developers. This SRS describes only a part of the overall election system, where actual voting is done separate from this software.

**Document Conventions**

This document will be written following the IEEE SRS template.

*<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>*

**Intended Audience and Reading Suggestions**

This document is intended for software developers in the future who will update this software along with testers of the software. Election officials can also look over this document if they have any concerns about the software that was created and the requirements used in design. It is best to read through this document mostly in order. The images for the user interfaces might not be as important as the section that talks about functionality for software developers.

*<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>*

**Product Scope**

*<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>*

Voting system is a software tool that is capable of determining the winners and losers of an election. The software will take in ballots and will be able to employ two voting algorithms, STV or plurality to best determine the winners and losers. The software will be able to provide the users with stat information to analyze about the election after it has run.

**References**

*<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>*

* Project 1 Waterfall Voting SRS Reference Sheet
* Software Requirements Specification Document Template
* Use Case Structured Specification Template Example

**Overall Description**

**Product Perspective**

This system will be for election officials to fairly get a count of an election without doing a physical count. This will decrease the amount of time it takes to count and decide an election. This will be a stand-alone software that may be updated in the future with menu options or implementation with other election software tools.

**Product Functions**

* Run an election system to determine the winners and losers of an election
  + Two different techniques for determining the winner can be select
    - STV voting
      * A Droop
    - Plurality voting
      * Votes are counted off of CSV files and the candidate(s) with the highest number of votes wins
      * Ties will be determined by a random selection between the candidates that tie
  + The system will then give results of the election
    - The type of election
    - Who won and who lost
    - Number of Ballots
    - Number of Seats
    - Candidates
    - Winners and Losers
  + Audit will be created to show how calculations were done

**User Classes and Characteristics**

* Voting Official - Most important and only intended user of this software. Has access to all functions of the software (collecting ballots, running voting algorithms, etc.) Should be technically experienced in running software.

**Operating Environment**

This program must run on a CSElabs machine. This is a linux based operating system with a Dell PC machine being used.

**Design and Implementation Constraints**

This software must run on a CSElabs machine so any hardware or interface constraints will come from this machine. No other applications or tools will need to be used in conjunction with running the software. The software will be written using C++ and the database that will be used is still to be determined when the algorithm is created. No security standards need to be used but the code created must be properly commented and designed so that it can easily be changed in the future.

**User Documentation**

All help and information about how to run the software will be given within the software itself (help screen). This screen will provide textual knowledge of how the system is run and how to run the system as an election official. This software will also need documentation of an election (CSV “,” delimited files) to run.

*<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>*

**Assumptions and Dependencies**

This software will be run independently with no additional software being needed. No dependencies or assumptions will affect the design of the system.

*<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues*

**External Interface Requirements**

**User Interfaces**

Questions will be asked on command module - specific inputs such as file names - prompt the user for what file you need

Assume the file is in the same folder???

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

**Hardware Interfaces**

The hardware interface for this system is out of the scope of the project. Software must work on CSElabs.

**Software Interfaces**

Once the program has been started, it will only be interacting with the user and the data. Data coming into the software includes csv files of votes for a candidate with a line for each voter and commas separating each vote.

**Communications Interfaces**

No communication with the internet is done in this software.

**System Features**

Features

* File selection
* Droop voting system
* Plurality voting system
* Help window
* Audit

**File Selection**

4.1.1 Description and Priority

This is an input from the user to select which file they would like to perform the vote counting algorithm on. This feature has a high importance as it is needed for being able to identify what file is being used for counting. This approach can be done with many different methods but it must be easy for the user to understand how to select a file. In the case of this software, a file is selected by an input of the file address by the user.

4.1.2 Stimulus/Response Sequences

This process is done after the voting type has been selected. The software will ask the user for the file address of the .csv file. If the file is not found, it will prompt the user to ask again for an address or to return to the main screen.

4.1.3 Functional Requirements

REQ-1: User inputs a file address for the software to find and begin counting votes of the file

REQ-2: The software will notify the user if a wrong file has been selected (such as not a CSV file or a CSV file without the proper

REQ-3: The software allows the user to return to the main screen if the file entered was incorrect

**Plurality Voting System**

4.1.1 Description and Priority

Plurality voting is a specific voting type where each vote is counted as one vote and the candidate(s) with the highest number of votes wins the election. This process is selected before a file is picked where the user is asked if CTV voting or plurality voting will be chosen. After all other required items have been determined, the plurality voting algorithm will determine the winners. This process isn't as high priority as STV voting but is still required to fulfill the requirements for this system.

4.1.2 Stimulus/Response Sequences

This process is done right at the start after the software has begun. The software will prompt the user with a question on which voting system they want to use. Once Plurality voting has been chosen, the voting system of plurality will be completed.

4.1.3 Functional Requirements

REQ-1: Users must be able to select which voting system they would like - plurality voting being one option

REQ-2: The software must allow the user to run plurality voting which is done with an algorithm that counts every vote and has the winners with the most votes displayed to the user

**STV Voting**

*<Don’t really say “System Feature 1.” State the feature name in just a few words.>*

4.1.1 Description and Priority

STV (Droop) voting is a specific voting type where each vote is counted until a “droop” number has been reached, once this number has been reached a winner and a loser is selected and the process starts over until all candidates have been placed in the winner or loser category. This process is selected before a file is picked where the user is asked if CTV voting or plurality voting will be chosen. After all other required items have been determined, the STV voting algorithm will determine the winners. This process is of high priority.

4.1.2 Stimulus/Response Sequences

This process is done right at the start after the software has begun. The software will prompt the user with a question on which voting system they want to use. Once STV voting has been chosen, the voting system will begin. Once completed, a list of winners and losers will be provided along with a method on how it was determined.

4.1.3 Functional Requirements

REQ-1: Users must be able to select which voting system they would like - STV voting being one option

REQ-2: The software must allow the user to run an STV voting system. This process will show winners and losers of the election based on the STV algorithm

REQ-3: This process is documented so that users understand how exactly a winner was determined

**Other Nonfunctional Requirements**

**Performance Requirements**

The calculation of winners and losers must be done in under 5 minutes.

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

***Software Requirements Specification for Droop Voting System Page***

**Safety Requirements**

All data used in testing must not be deleted. All data will be properly counted so that the correct winners and losers of an election can be chosen.

*<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>*

**Security Requirements**

This Software will not require any security requirements. All privacy and log in possibilities are out of the scope of this project.

**Software Quality Attributes**

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

**Business Rules**

*<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>*

**Other Requirements**

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

**Appendix A: Glossary**

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: To Be Determined List**

***Software Requirements Specification for Droop Voting System Page***

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*

Questions:

For the actual system, will a full GUI need to be created?

Do Cin statements apply for UI

Will the file always be in the same folder?

What does the help window imply?

How do we save the report and what do we save it as?

Do we need diagrams for use cases or just textually

Answers to questions

No security or log in - make it this way in the future if needed

File Names: No set file naming convention - assume plurality - file(s) will align

Shuffle can be anything but must be random - continuously randomize

We get to pick what the screen would look like - no specific order to hte information needed

Can use text based and not necessary a GUI

Audit file must occur

Election can be stopped before the algorithm starts

Audit file should be able to see every vote from the system

Command line interface - don’t provide the election official a way to turn off the

Assume program will be run from terminal -- terminal text interface is good

For c++ - the make files are also needed

Help crreen should be giving info to user on the UI

No internet access

Always audit

User errors - handle user interface by user

Always assume we will have the rightn umber of candidates