Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Team 3 (Hailin Archer - deak0007, Bryan Baker - bake1358, Colin Kluegel - klue0037, Josh Spitzer-Resnick - spitz123)

CSCI 5801 - Software Engineering I, Spring 2020

University of Minnesota

February 21, 2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Baker	2/16/2020	1.1, 1.2, 1.3, 2.2, 2.6, 3.3, 5.1, 5.5	0.5
Archer	2/16/2020	2.1, 2.5, 3.2, 5.4, Appendix B	0.4
Baker	2/15/2020	UC_1, UC_5	0.3
Kluegel	2/15/2020	UC_2, UC_6	0.2

Archer	2/14/2020	UC_4, UC_8	0.1
Archer	2/13/2020	Cover Page	0.0

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the *Voting System (VS) Version 1.0*. 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.

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

The intended audience includes:

- Election officials who want to use the VS to run an election with supplied ballot information.
- Election officials and system testers who want to test and calibrate the VS.
- Programmers who are interested in working on the project by further developing it or fixing existing bugs.

1.4 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.>

The software system being developed is a voting system to be used in local elections. The system will be designed to automate the counting of ballots to simplify the running of elections. The main feature of the software will be to run two types of elections, a plurality voting election and a single transferable voting (STV) election.

In addition to its primary purpose of running an election the software will need to provide some additional features. The software needs to display detailed information about the election results, that is, it should display the number of ballots, the number of seats, the number of candidates and the winner /winners of the election. For STV elections the software will create a detailed report that will act as an audit for the election. The report will be saved as a text file and show details about how ballots were assigned to candidates as the election progressed.

The software will also require diagnostic modes. One of these diagnostic modes will be a calibration mode that will turn off the ballot shuffling feature (the ballot shuffling feature is required for STV elections). The software will also include a debug mode that can be used for developer testing purposes.

To aid the user of the voting system a help window will be provided that will give the user information about how to run the program.

1.5 References

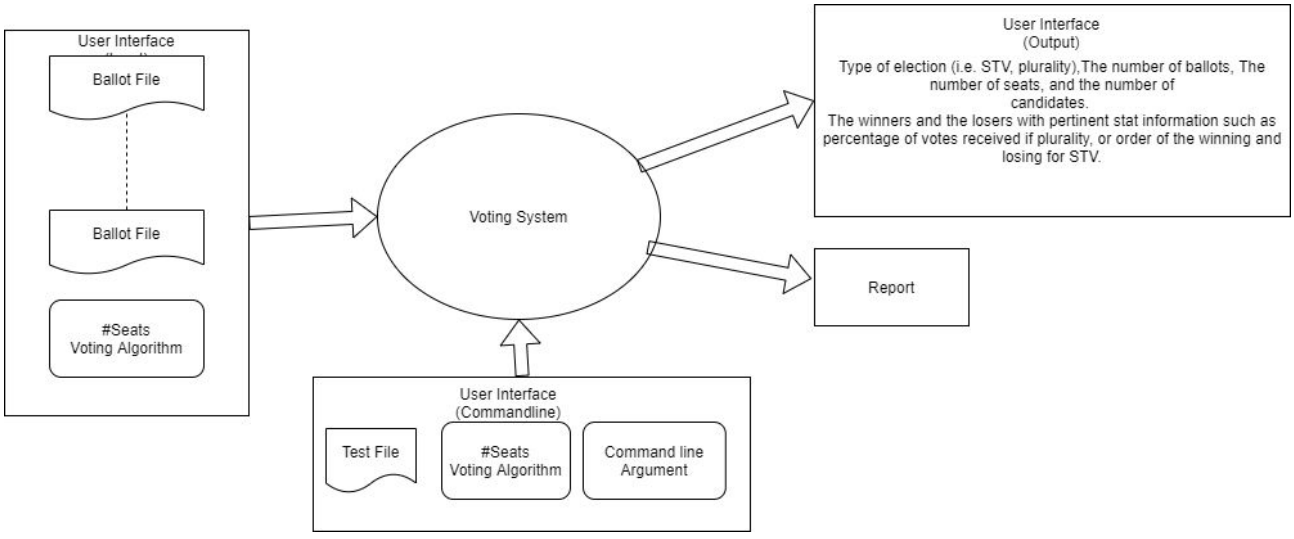
<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference,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

2. Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

Voting system is a new system capable of performing both plurality voting and an STV system using the Droop quota. It is a stand alone program.



2.2 Product Functions

The system shall be able to:

- Accept user inputs including:
    - The number of election seats to fill
    - The type of election to run
    - The file location for the election ballots

- Import election ballots
    - Provide the ability to shuffle the election ballots
- Run an election
    - Run a plurality election type
    - Run an STV election type
    - Provide the ability to break ties in an election
- Provide election results
    - Provide elected and non-elected candidate lists
    - Calculate election statistics
    - Provide election audit
- Provide the ability to calibrate or test the system under repeatable circumstances
    - Provide the ability to turn off the election ballot shuffle
- Provide a help window for the user

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

There are three primary classes of users for the voting system

Election Official
- **Definition**: The election official is in charge of setting up and running the election
- **Functionality Required**: The election official needs to be able to set up the election by specifying the election type (STV/plurality), the number of seats and inputting the ballot files. After the election is run the election official will need to be able to see the results of the election.
- **Security Required**: ?

Tester
- **Definition**: A tester is a user who checks the software to make sure that it is properly calibrated and is outputting the correct results.
- **Functionality Required:** A tester needs to be able to set the software into calibration mode by disabling the ballot shuffling functionality. Once they have entered the software into calibration mode they need to be able to run an election.
-**Security Required:** ?

Developer
- **Definition**: A developer is a user who writes and maintains the code for the voting system
- **Functionality Required**: A developer needs to be able to understand the code base and understand how to use the developer/debug mode features
- **Security Required**:

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

Voting system shall be written in C++. It shall run on cse labs computer. Testers shall be able to build the program with a single makefile. This system shall be delivered in 4 weeks.

2.6 User Documentation

An on-line help window will be provided for the user in the VS. This window will provide information on how to operate within the system.
<Q - Should the help window be delivered within the voting system or open a browser and deliver via HTML?>
<Q - Do the users require documentation to start the program? If so what format?>
<Q - Do the users require documentation on how to turn off the shuffle ballot feature? If so what format?>

2.7 Assumptions and Dependencies

<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 around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

The voting system must be written in either Java or C++ and must be capable of running on a CSE labs machine. The ballot files need to be CSV files saved as a Windows Comma Separated file where each row is separated by a newline.

3. External Interface Requirements

3.1 User Interfaces

<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.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

Voting system is a stand alone program designed to run on a personal computer.
<Q- how do we determine computer hardware requirement for this system?>

3.3 Software Interfaces

The following software shall be used:

- The VS shall be programmed in C/C++ version 7.4.0 as provided on CSE lab machines running Ubuntu 18.04.4 LTS using the Linux 4.15.0-76-generic x86_64 kernel.
- User input shall be through a standard keyboard either on site or through a proxy (for example SSH).
- The VS shall read Windows standard comma separated (CSV) files.
- Audit reports shall be written to a standard text file.
- All user interfaces including input requests and help systems will be provided in a text based format.

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:
REQ-2:

4.2 System Feature 2 (and so on)

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system should operate given the following function and performance constraints:

- Training for the user interface should take less than 15 minutes.
- The election needs to be processed in under 5 minutes.

- Run on a CSE lab machine from the command prompt.
- Must process up to 100,000 ballots.

5.2 Safety Requirements

<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.>

<Q - Are there any safety requirements for this voting system?>
<Q - Could loss of faith in election be a possible harm that could happen from use of this product if the software fails?>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 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.>

Voting system provides the users with both simple and advanced features. Due to its well designed and easy to use interface it can be used by both experts and typical users. Voting system provides a help window where users can get instruction on how to use the system.
However, users must already have a basic knowledge of linux before using it.

5.5 Business Rules

Ballot rules:

- Plurality ballot
    - The first line of the file will provide the names of the candidates and each line after that will be a voter's ballot.
    - Each ballot will have only one candidate indicated as their choice. This will be indicated by the number "1".
    - An example of a ballot file provided for a plurality election is:
      A,B,C,D,E,F
      1,,,,,

,,1,,,
,,,,,1

- STV ballot
    - The first line of the file will provide the names of the candidates and each line after that will be a voter's ballot.
    - Each ballot must have at least half of the candidates ranked.
    - Each ballot will have numbers listed for each candidate in order of preference. A "1" is the first choice, a "2" is the second, etc. A ballot does not have to rank all candidates.
    - An example of a ballot file provided for an STV election is:
      A,B,C,D,E,F
      1,,2,,3,
      3,2,1,4,6,5
      1,2,,,3,4
      4,1,,2,,3

Plurality election rules:

- Each ballot will be assigned to a candidate and a count maintained for each candidate.
- The candidate(s) with the greatest number of votes will be declared the winner(s).
- If there are ties in an election the winner will be chosen from the pool of tied candidates.
- If there are seats to fill but the remaining candidates have no votes, winners will be chosen randomly.

STV election rules:

- The STV election will be conducted using the Droop Quota algorithm as carried out using the following steps:
    a. Shuffle the ballots.
    b. Calculate the Droop Quota:

$$\left\lfloor \left(\frac{NumberOfBallots}{NumberOfSeats + 1} \right) \right\rfloor + 1$$

    c. Distribute all ballots
        - One-at-a-time to each candidate ordered by first vote received.
        - Any candidate who reaches the quota defined above is declared a winner and the ballots are removed from the election. The candidate is added to the winner list in the order elected.
        - Any ballot with a vote for a winning candidate will go toward the next candidate on that ballot's list.
    d. Redistribute ballots from losing candidates
        - Candidates with the fewest number of votes are removed from the voting pool and placed on the losing list.
        - In the case of a tie for the losing position, the candidate who was last to receive their first ballot is eliminated.
        - If, upon redistribution, a candidate receives the quota they will be added to the winning list.
    e. Continue until all ballots have been processed

        - There will be two lists upon completion: a winning list and a losing list.
        - These lists will be ordered based on when they were elected and when they were placed on the non-elected list.
        - The candidates placed on the non-elected list last will be higher up on the election list itself.
        - Both lists will be provided to the user.
    f. Provide an election audit report
        - Show the ballots that were assigned to a candidate as the election progressed
        - All election data must be included in the report at the top of the report. This includes: type of election, number of seats, number of candidates, winners, losers, etc.
        - This report should be output to a text file.

Election officials may:

- Input the number of seats to fill in an election
- Input the file location for election ballots
- Run a plurality type election
- Run an STV type election
- Review election results
- Perform election audits
- Ask for help through a help window
- Turn off the ballot shuffle option to test for system calibration

Developers / Testers may:

- Perform all activities described for election officials
- Use command line arguments to facilitate testing and development.

6. Other Requirements

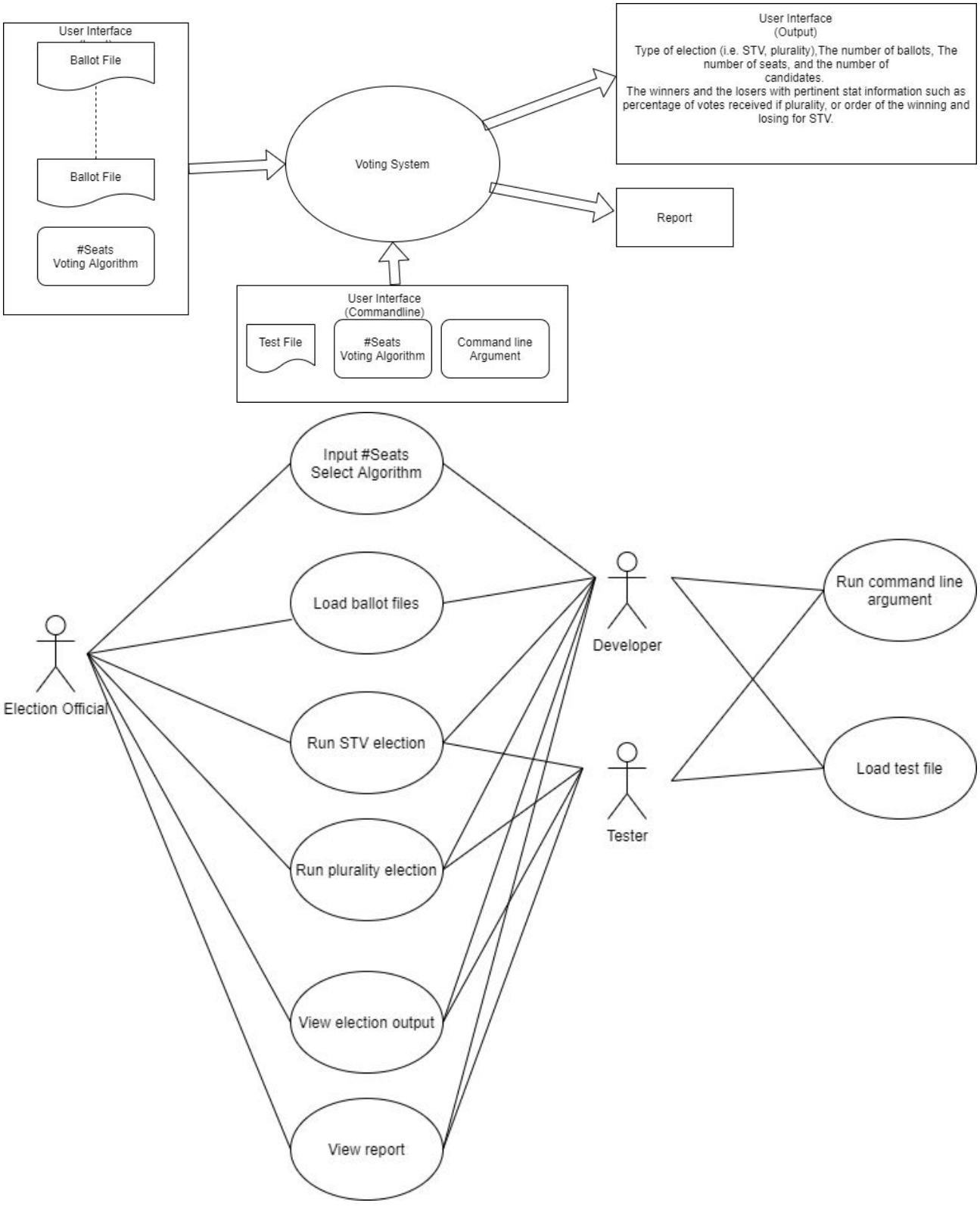
<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

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.>


User Interface
Ballot File
Ballot File
#Seats
Voting Algorithm
Voting System
User Interface
(Output)
Type of election (i.e. STV, plurality),The number of ballots, The number of seats, and the number of candidates.
The winners and the losers with pertinent stat information such as percentage of votes received if plurality, or order of the winning and losing for STV.
Report
User Interface
(Commandline)
Test File
#Seats
Voting Algorithm
Command line
Argument
Input #Seats
Select Algorithm
Load ballot files
Run STV election
Run plurality election
View election output
View report
Election Official
Developer
Tester
Run command line
argument
Load test file

Appendix C: To Be Determined List

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