Software Requirements Specification

for

Voting System

Version 2.0 approved

Prepared by Carlos Alvarenga (alvar357), Justin Koo (kooxx078), Michael McLaughlin (mclau361), Xiaochen Zhang (zhan4487)

Team 17

10/22/2018

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
First Draft	10/08/2018	Initial writeup	1.0
Revised Draft	10/22/2018	Revise document based on grader's feedback	2.0

1. Introduction

1.1 Purpose

This document serves as the SRS for the VS - version 2.0. This SRS covers the entire scope of the VS-version 2.0, including any and all existing and future subsystems. The SRS describes only the single subsystem (i.e. VS) that exists on its own. The actual voting will be done separately from the VS that is specified within the SRS and will be designed and implemented. Ballots will be cast online and a comma delimited text file will be provided to the VS program.

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

1.2 Document Conventions

As a convention used throughout this SRS, distinct user roles that act within the system will be italicized. For example the *user* denotes the system-level entity whereas the term user denotes the entity as used in plain English. Higher-level requirements are assumed to be inherited by detailed requirements.

<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.>

1.3 Intended Audience and Reading Suggestions

This document is intended for developers as well as project managers, users, testers, documentation writers and other staff with a base level knowledge of technical concepts. As an example, it is sufficient for the audience to know at a high level what a database is and why they are used, but not necessary to know the full details of the internal workings of one.

This document is organized into 6 sections. Section 1 (this section) gives a brief introduction to the VS product as well as conventions used within the SRS itself. Section 2 provides an overall high-level description of the VS. Section 3 outlines the various external interfaces that the VS must interact with. Section 4 describes the set of functional requirements in the form of use cases that are pertinent to the VS. Section 5 describes the set of nonfunctional requirements that are pertinent to the VS. Finally, Section 6 describes requirements that belong neither in Section 4 nor Section 5.

It is recommended that completely new audiences to the VS product read through this entire SRS document. Audiences that already have a high level understanding of the VS including its purpose, scope, and core modules may wish to proceed to Sections 4, 5 and 6 for a more detailed outline of the various requirements.

<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.>

1.4 Product Scope

The VS product is a program will be written in the Java programming language and will consist of Java source file(s) and class file(s) that implement said product.

The purpose of the VS program is to grant *election officials* the capability to perform IRV and OPLV to determine election winners in either voting system. Based on all the ballots received and gathered, this product will streamline the *election officials'* process of tallying up the votes to find a winner when conduction IRV or OPLV. Doing so will allow the *elected officials* to reap organizational, logistical benefits by providing them an efficient, unbiased voting system to perform IRV and OPLV elections and determine the winning candidate, according to the public's expressed interest.

<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.>

1.5 References

The description of the VS product, as well as any requirement specifications within this SRS, were initially specified by the customers found here:

https://ay17.moodle.umn.edu/pluginfile.php/2780770/mod_assign/introattachment/0/Project1_Waterfall_VotingSRS_Fall2018V3_92518.pdf?forcedownload=1

<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

The product specified is a new, self-contained program that is not built on top of any pre-existing system. However, it may later be integrated into a larger system once it has been designed, implemented and thoroughly tested.

<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.>

2.2 Product Functions

In summary, the 4 basic modules along with their relevant functions of the VS are:

1. The user runs the program
2. The user is prompted for the name of the file that contains the ballots and voting information
3. The program uses the inputted file to tally up votes and determine the election winner
4. The program outputs the winning candidate and pertinent election information on the CLI on the terminal window or a GUI, as well as an audit file in the same directory that the program resides in.
    a. The audit file contains election information at the time (e.g. Type of Voting, number of candidates, candidates, number of Ballots, calculations, how many votes a candidate had, etc). The audit should show how the election progressed so that it could be used to replicate the election itself.
    b. The output to the screen should be the winner(s) and basic information about the election such as the type of election and number of seats. The candidates and their election stats, such as the number of votes he/she received, should also be shown on the screen.

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

2.3 User Classes and Characteristics

The users of the VS will be classified within the product as 1 of 3 classes described in detail below. Classes are differentiated by various factors but are primarily separated according to the needs and functions of a certain class.

1. The *programmers* are the users responsible for the design and implementation the VS program. They are responsible integrity and correctness of the VS. They are the ones creating the VS product that can be used by the *testers* and *elected officials* according to their needs. Consequently, they are the ones creating and writing the program files and are responsible for the program's correctness.
2. The *testers* are the users that can run the VS program and test it for its correctness. They are knowledgeable enough to understand the program so they have technical expertise, education level and experience equal to or greater than that of *programmers*. They have information that the *programmer* does not possess, but instead prompt from the program users, such as sample voting ballot files designed to test the product. Moreover, they have access the programs output, such as the election results and the audit file. Finally, *testers* have access to the program files to be able to analyze and test it.
3. The *elected officials* are the primary users who will use this product to conduct IRV and OPLV elections. They have direct access to the actual voting ballot files and can supply them to the

program as needed to determine the winner. They also have access to the program's output, such as the election results and the audit file. However, they do not necessarily have access to the actual program files since they're not required to have the same technical expertise as the programmer or tester. Instead, they are solely interested in running the VS program to correctly perform IRV and OPLV elections.

<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.>

2.4 Operating Environment

The VS program must be able to be run and executed in CSE Linux lab machines with a current version of Java installed. Java programs will be run either at the CLI or through Eclipse.

<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

The VS program must be able to be executed in CSE lab machines.
No pre-processing of the voting ballot files will be done on the *testers'* or *elected officials'* end to accommodate the VS program's needs or flow of logic. In other words, the ballot files will have a certain structure that will not be altered by *election officials* and *testers*.

<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).>

2.6 User Documentation

TBD. No other documentation has been created at this time. Supplementary documentation will be created after completion of the SRS.

<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.>

2.7 Assumptions and Dependencies

- Product users have the proper credentials to be using it. For example, *elected officials* are qualified to be using the VS program to perform an election
- The voting ballot files do not contain errors
- Each ballot will have at least 1 rank and no person will have more than 1 ranking. The ranking numbers will not have issues
- Ballot files sizes will be small enough so they will be able to be loaded into the local machine's main memory.
- The VS program will only need to be able to execute IRV and OPLV elections.
- The IRV ballot file will have the following structure (see reference document link in section 1.5):

  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

  1st Line: IR for instant runoff voting
  2nd Line: Number of Candidates
  3rd Line: The candidates separated by commas
  4th Line: Number of ballots in the file

- The OLV ballot file will have the following structure (see reference document link in section 1.5):

  OPL
  6
  [Pike,D], [Foster,D],[Deutsch,R], [Borg,R], [Jones,R],[Smith,I]
  3
  9
  1,,,,,
  1,,,,,
  ,1,,,,
  ,,,,1,
  ,,,,,1
  ,,,1,,
  ,,,1,,
  1,,,,,
  ,1,,,,

  1st Line: OPL for open party listing voting
  2nd Line: Number of Candidates

3rd Line: The candidates and their party are denoted in [ ]. The name and party are separated by a comma.
4th Line: Number of seats
5th Line: Number of ballots

* It can be assumed that there are no errors in the ballots. Only a single 1 will be placed in the position of the given candidate selected.

<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).>

3. External Interface Requirements

3.1 User Interfaces

TBD. Users will be interacting with the program through the CLI. However, a GUI may be added in the later stages of the product development.

<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

A CSE Linux lab machine is the hardware needed to run the VS program

<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.>

3.3 Software Interfaces

The VS program needs to be run in a Linux environment with a current working version of Java installed. Java programs will be run either at the CLI in a Linux terminal or through Eclipse software.

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components.

Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

TBD. No communication interfaces have been specified at this time. The VS program will exist within its own environment without interaction with another server or cloud-based system.

<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

Features of the VS program in this section are organized by use case. Since the VS program has a relatively linear sequence of steps executed by users, the use cases are listed in chronological order.

<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 Users Run The Instant Runoff Voting Program

4.1.1 Description and Priority

User(s) run the program by providing ballot files to the voting program to determine the election winner of an IRV election. High 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

Initial step-by-step Description (“Triggers” and “Main Course” section of UC_001 in attached use case document):

1. The user navigates to the directory in which the program resides in
2. The user attempts to compile and run the program through a CLI argument “javac filename.java” or a GUI that hasn’t been designed yet

3. The local machine confirms that the program files exist to compile and run them (see EX1)
4. The local machine compiles the program files (see EX2)
5. The local machine runs the program files (See AC1)
6. The program outputs the winner on the CLI or GUI and generates an audit file in the same working directory that the program resides in

<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

AC-UC_001-1: The local machine can't run the program files or it doesn't terminate

1. The user makes sure that the installed software is capable of running the program
2. Return to main course 1

EX-UC_001-1: The local machine determines that the program files aren't in the same directory or don't exist

1. The local machine notifies the user that the program files can't be found
2. The user verifies that the program files are downloaded and placed in the appropriate directory where they can be run
3. Return to main course 1

EX2-UC_001-2: The local machine can't compile the program files

1. The user makes sure that the installed software is capable of running the program
2. Return to main course 1

<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.>

4.2 Users Run The Open Party Listing Voting Program

4.2.1 Description and Priority

User(s) run the program by providing ballot files to the voting program to determine the election winner of an OPLV election. High Priority.

4.2.2 Stimulus/Response Sequences

Initial step-by-step Description ("Triggers" and "Main Course" section of UC_002 in attached use case document):

1. The user navigates to the directory in which the program resides in
2. The user attempts to compile and run the program through a CLI argument "javac filename.java" or a GUI that hasn't been designed yet
3. The local machine confirms that the program files exist to compile and run them (see EX1)
4. The local machine compiles the program files (see EX2)
5. The local machine runs the program files (See AC1)
6. The program outputs the winner on the CLI or GUI and generates an audit file in the same working directory that the program resides in

4.2.3 Functional Requirements

AC-UC_002-1: The local machine can't run the program files or it doesn't terminate

1. The user makes sure that the installed software is capable of running the program
2. Return to main course 1

EX-UC_002-1: The local machine determines that the program files aren't in the same directory or don't exist

1. The local machine notifies the user that the program files can't be found
2. The user verifies that the program files are downloaded and placed in the appropriate directory where they can be run
3. Return to main course 1

EX2-UC_002-2: The local machine can't compile the program files

1. The user makes sure that the installed software is capable of running the program
2. Return to main course 1

4.3 Users Specify Input File

4.3.1 Description and Priority

Users specify the name of the file containing ballot records. High priority.

4.3.2 Stimulus/Response Sequences

Initial step-by-step Description ("Triggers" and "Main Course" section of UC_003 in attached use case document):

1. The user runs the program via the CLI.
2. The user specifies the name of a file that contains the desired ballot records.
3. The program confirms the specified file exists in the same directory as the program and it has permission to read the contents (see EX1).

4. The program reads and parses input from the specified file.

4.3.3 Functional Requirements

REQ-UC_003-1: The specified file exists in the same directory as the program

REQ-UC_003-2: The program has permission to read the specified file

EX-UC_003-1: The program determines that either the specified file does not exist in the same directory as the program or the program does not have permission to read the contents.

1. The program notifies the user of the specific error
2. Return user to Main Course step 2

4.4 Users See Results

4.4.1 Description and Priority

Users can see the information relating to the election as well as the winner(s) on the CLI. High priority.

4.4.2 Stimulus/Response Sequences

Initial step-by-step Description ("Triggers" and "Main Course" section of UC_004 in attached use case document):

1. The program finishes the ballot tallying phase with no fatal errors (see EX1).
2. Election information and its winner(s) are displayed to the CLI.

4.4.3 Functional Requirements

REQ-UC_004-1: The ballot tallying phase encountered no fatal errors.

EX-UC_004-1: The program encounters fatal errors during the ballot tallying phase.

1. The program notifies the user of the error.
2. Return to UC_003.

4.5 Users View Audit

4.5.1 Description and Priority

Users can open and view the audit file generated by the program. High Priority.

4.5.2 Stimulus/Response Sequences

Initial step-by-step Description ("Triggers" and "Main Course" section of UC_005 in attached use case document):

1. The user navigates to the directory in which the program resides.
2. The user attempts to open the audit file generated (see AC1).
3. The local machine confirms that the user has permission to read the contents of the audit file (see EX2).
4. The contents of the audit file are visible to the user.

4.5.3 Functional Requirements

REQ-UC_005-1: The user is examining the directory in which the program resides.

REQ-UC_005-2: The audit file exists in the directory in which the program resides.

REQ-UC_005-3: The user has permission to read the contents of the audit file.

AC-UC_005-1: The user determines that the audit file does not exist in the directory in which the program resides.

EX-UC_005-1: The local machine determines that the user does not have permission to read the contents of the audit file.

1. The local machine notifies the user of the permission restriction.
2. The user somehow obtains the necessary permissions.
3. Return to Main Course step 2.
4. Return to UC_003

4.6 Testers Run The Voting Program

4.6.1 Description and Priority

Tester types in CLI arguments to run the program or to run a testing script to compile and run the voting program. High Priority.

4.6.2 Stimulus/Response Sequences

Initial step-by-step Description ("Triggers" and "Main Course" section of UC_006 in attached use case document):

1. The user navigates to the directory in which the program resides
2. The user attempts to compile and run the program through the CLI argument "javac filename.java" or runs a testing script that'll do this job
3. The local machine confirms that program files exist to compile and run them (see EX1)
4. The local machine compiles the program files (see EX2)
5. The local machine runs the program files so that user can verify the program's correctness (See AC1)
6. The program outputs the winner on the CLI and generates an audit file in the same working directory

4.6.3 Functional Requirements

AC-UC_006-1: The local machine can't run the program files or the program doesn't terminate

EX-UC_006-1: The local machine determines that the program files aren't in the same directory

1. The local machine notifies the user that the program files can't be found
2. The user verifies that the program files are downloaded and placed in the same working directory

3. Return to main course 1

EX2-UC_006-2: The local machine can't compile the program files

1. User verifies that the local machine is running the most updated version of Java
    a. If the correct version is being used, the program is deemed to be incorrect
    b. If not, update Java to its current versions and return to main course 1

4.7 Election Officials Run The Voting Program

4.7.1 Description and Priority

Election official(s) run the program by providing ballot files to the voting program to determine the election winner of an IRV or OPLV election. High Priority.

4.7.2 Stimulus/Response Sequences

Initial step-by-step Description ("Triggers" and "Main Course" section of UC_007 in attached use case document):

1. The user navigates to the directory in which the program resides in
2. The user attempts to compile and run the program through a CLI argument "javac filename.java" or a GUI that hasn't been designed yet
3. The local machine confirms that the program files exist to compile and run them (see EX1)
4. The local machine compiles the program files (see EX2)
5. The local machine runs the program files (See AC1)
6. The program outputs the winner on the CLI or GUI and generates an audit file in the same working directory that the program resides in

4.7.3 Functional Requirements

AC-UC_007-1: The local machine can't run the program files or it doesn't terminate

1. The user makes sure that the installed software is capable of running the program
2. Return to main course 1

EX-UC_007-1: The local machine determines that the program files aren't in the same directory or don't exist

1. The local machine notifies the user that the program files can't be found
2. The user verifies that the program files are downloaded and placed in the appropriate directory where they can be run
3. Return to main course 1

EX2-UC_007-2: The local machine can't compile the program files

1. The user makes sure that the installed software is capable of running the program
2. Return to main course 1

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The VS program should be able to process a ballot file with 100,000 ballots in under 8 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.>

5.2 Safety Requirements

TBD. No safety requirements have been specified at this time

<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.>

5.3 Security Requirements

TBD. No security requirements have been specified at this time

<Specify any requirements regarding security or privacy issues surrounding the 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

1. Availability: The VS program should provide the users with both simple and advanced features. There will be a well designed and easy-to-use interface such that both experts and typical users should be able to use it.
2. Correctness: The VS program should be able to input/output correct files and implement correct voting procedure.
3. Maintainability: The voting result should maintain correctly based on input files.
4. Usability: The VS program should be able to handle the size of users' input file as long as it can be loaded into main memory.

<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.>

5.5 Business Rules

Users have user level rights and can perform the following functions:

1. Enter a filename when prompted.
2. Specify missing information not included in the file.
3. Enter the name of the audit file that will be produced.
4. See winner(s) and information about the election on the CLI.
5. Open and view the audit file produced.

Testers have tester level rights and can perform the following functions:

1. Run the program to generate election winner and the contents of the audit file are visible to the user after all input information satisfies the precondition.

<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.>

6. Other Requirements

TBD. No other requirements relating to database, internationalization or legalities have been specified at this time

<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

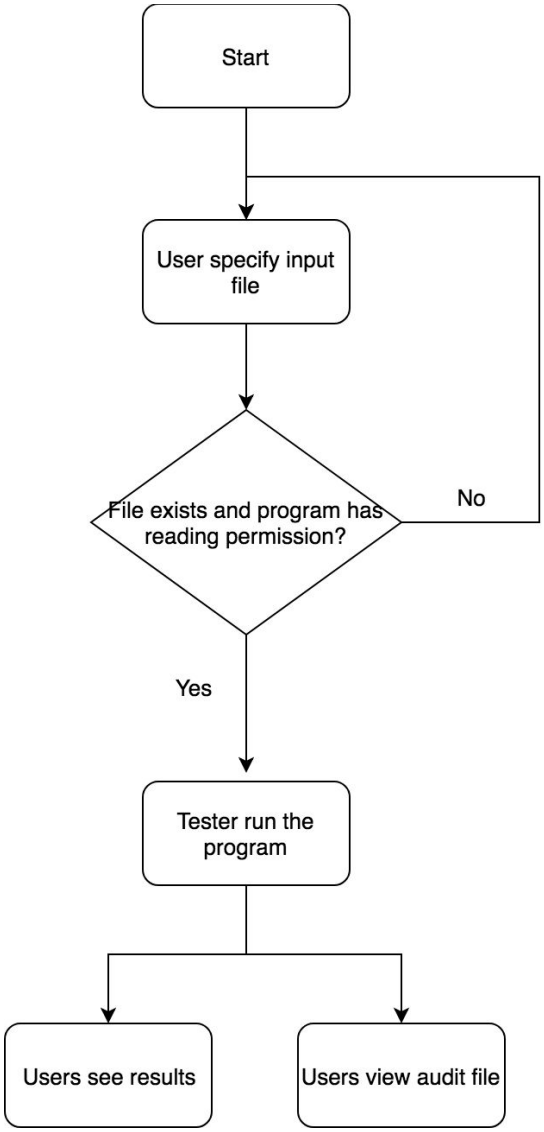
Acronym	**Long Form**	**Definition**
SRS	Software Requirements Specification	A description of a software system to be developed
IRV	Instant Runoff Voting	A plurality/majority voting system also known as the “majority preferential voting”. One of the voting systems that needs to be implemented as part of the VS and whose requirements are specified within this SRS.
OPLV	Open Party List Voting	A proportional representative type of voting

		system that also needs to be implemented as part of the VS and whose requirements are specified within this SRS.
VS	Voting System	The voting system program that must be created and is capable of performing IRV and OPLV. This is the product being described and specified within this SRS
GUI	Graphical User Interface	An interface that includes graphical elements for users to interact with
CSE	College of Science & Engineering	One of the colleges of the University of Minnesota in Minneapolis, Minnesota
TBD	To Be Determined	An acronym for “to be determined”
CLI	Command Line Interface	A form of sending commands to the computer via successive lines of text

<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

Activity Diagram:



<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

2.6 User Documentation
3.1 User Interfaces
3.4 Communication Interfaces
5.2 Safety Requirements
5.3 Security Requirements
6.* Other Requirements

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