Software Requirements Specification

for

Voting System

Version 1.0 approved

Prepared by Daniel Rockcastle (rockc004), John Caspers (caspe158), Michael Birk (birkx031), and Mukesh Pragaram Choudhary (choud108)

CSCI 5801 Team 6

October 8, 2018

Table of Contents

Table of Contents ii
Revision History iii
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 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 3
3.4 Communications Interfaces 3
4. System Features 3
4.1 Election Official Requests Audit File 3
4.2 Tester Requests Audit File - Instant Runoff 4
4.3 Tester Requests Audit File - Open Party 4
4.4 User Views Election Results 5
4.5 User Views Audit File 5
5. Other Nonfunctional Requirements 5
5.1 Performance Requirements 5
5.2 Safety Requirements 6
5.3 Security Requirements 6
5.4 Software Quality Attributes 6
5.5 Business Rules 6
6. Other Requirements 6
Appendix A: To Be Determined List 6

Revision History

Name	Date	Reason For Changes	Version
Michael Birk	10/05/18	Adding use cases.	1.0
John Caspers	10/05/18	Adding use cases.	1.0
Mukesh Pragaram Choudhary	10/05/18	Adding use cases.	1.0
Daniel Rockcastle	10/05/18	Adding use cases.	1.0
Mukesh Pragaram Choudhary	10/05/18	Completing sections 5 and 6.	1.0
Michael Birk	10/07/18	Completing section 4.	1.0
John Caspers	10/07/18	Completing section 3.	1.0
Daniel Rockcastle	10/07/18	Completing sections 1 and 2.	1.0
Daniel Rockcastle	10/08/18	Updating formatting and combining SRS and use case documents.	1.0

1. Introduction

1.1 Purpose

This document describes the Voting System software, revision 1.0. The purpose of this software is to calculate election results based on a given set of ballots, with functionality for both instant runoff and party list voting.

1.2 Document Conventions

This document is based on the IEEE's Software Requirements Specification template.

1.3 Intended Audience and Reading Suggestions

This document is written for developers, testers, documentation writers, and project managers who will make the Voting System software as well as election officials who will verify that this specification meets the desired requirements. This document is divided into sections, each containing specific information pertaining to different parties involved in the software development:

1. Overall description is intended for both election officials and the development team so that they can effectively understand high-level system details.
2. External interface requirements are intended for developers and testers so that they can understand what systems the software needs to interface with and how this should be done; election officials should also be aware of the software's external dependencies.
3. System features should be understood by the development team so that the intended features are developed, development work is properly distributed, and meaningful tests are written; election officials should also ensure these features align with the desired functionality of the software.
4. Other nonfunctional requirements should also be understood by the development team to ensure robustness and efficiency.

1.4 Product Scope

The Voting System software will automate the vote counting process, providing election officials with a fast and simple way to calculate election results. The software also generates an audit describing the details of the results that can be viewed by an official later.

1.5 References

IEEE Software Requirements Document template:
https://ay17.moodle.umn.edu/mod/resource/view.php?id=1347603

Use case specification template/example:
https://ay17.moodle.umn.edu/mod/resource/view.php?id=1359508

Voting System overview:
https://ay17.moodle.umn.edu/mod/assign/view.php?id=1603952

Voting System Use Cases document; written by Daniel Rockcastle, John Caspers, Michael Birk, and Mukesh Pragaram Choudhary; version 1.0; released 10/08/2018: appended to end of document.

2. Overall Description

2.1 Product Perspective

The Voting System is new software and is a standalone program, intended to automate the vote counting process for election officials.

2.2 Product Functions

The Voting System will perform the following key functions:

- Read an input file of ballots and calculate election results based on the input.
- Display the results of a calculated election on the screen.
- Generate an audit file containing the results and details of an election.

2.3 User Classes and Characteristics

The Voting System requires a user to run it. There are three user classes:

2.3.1 Election Officials

Election officials are the primary users of the software. They will use the software to calculate the results of normal and special elections, as necessary. Of the intended user classes, election officials will likely have the lowest amount of technical knowledge and will only be required to provide the software with an input file and interpret the output the software provides.

2.3.2 Testers:

Testers will only use the software to verify development work and ensure functionality, which will occur many times before the software's release. They must have a knowledge of both the technical, backend aspects of the software and how the results should appear to the end-user. Testers will be required to use the software in as many ways as possible to ensure correctness and robustness.

2.3.3 Programmers:

Programmers will only use the software during development to verify intended functionality. This will occur continuously during the coding/development process. They should have an extensive knowledge of the software's back-end and be familiar with the necessary outputs.

2.4 Operating Environment

The software will operate on Dell OptiPlex 9020 machines running Ubuntu version 18.04. In general, the software only needs version 8 or later of the Java Runtime Environment to run.

2.5 Design and Implementation Constraints

The results of an election should be able to be calculated within eight minutes given 100,000 ballots.

2.6 User Documentation

User documentation will be provided to election officials alongside this software.

2.7 Assumptions and Dependencies

- Write-in candidates are not allowed.
- Only one input file will be provided to calculate an election.
- The most up-to-date CSE Labs machines are being used to run the software.
- The input file will be in the same directory as the program executable.
- There are no security or safety requirements.

3. External Interface Requirements

3.1 User Interfaces

The user can either be prompted with a GUI or use the command line to run the program. After the program is run, an audit file will be produced in a specific folder the user can view. The main program and election file input will exist within the same directory.

3.2 Hardware Interfaces

The program must be able to run on the most up-to-date University of Minnesota CSE lab machines. The computers are Dell OptiPlex 9020 with Intel Core i7 @ 3.6 GHz. These machines should be able to handle 100,000 ballots in under 8 minutes.

3.3 Software Interfaces

The program must be compatible with the latest version of Java. This program must work on a variety of operating systems including Linux and Windows.

3.4 Communications Interfaces

This will be an offline system so any online functions such as email, web browser, and database server connections are ignored. There are no safety or security requirements except ensuring that one vote by a person is handled at the voting center.

4. System Features

4.1 Election Official Requests Audit File

4.1.1 Description and Priority

Description: Testers produce an audit file which contains the information about the election. Election official requests this file from the testers to verify.

Priority: High. This is a high priority feature since it is a required feature of the system. The election official will want to see the audit file for the election.

4.1.2 Stimulus/Response Sequences

1. System produces an audit file using the file given and saves in the same directory.
2. Election Official requests the file in the command prompt.

4.1.3 Functional Requirements

REQ-1: Ballot fed into program follows format of either open-party or instant-runoff.
REQ-2: Ballot filename exists.
REQ-3: Output file name is unique, ensuring not to overwrite already made audit.

4.2 Tester Requests Audit File - Instant Runoff

4.2.1 Description and Priority

Description: A tester inputs an election ballot that follows instant-runoff. They then request an audit file that they can view to confirm the results were correct.

Priority: High. This is a high priority feature since it is a required feature of the system. The tester will want to see the audit file for the election.

4.2.2 Stimulus/Response Sequences

1. Tester opens program by starting it through command line.
2. Program prompts for an input file.
3. Tester writes name of an input file that matches the instant-runoff ballot file pattern.
4. System reads in input file.
5. System calculates winner of election by counting votes, recording winner and process in audit file.

4.2.3 Functional Requirements

REQ-1: Ballot fed into program follows format of open-party.
REQ-2: Ballot filename exists.
REQ-3: Output file name is unique, ensuring not to overwrite already made audits.

4.3 Tester Requests Audit File - Open Party

4.3.1 Description and Priority

Description: A tester inputs an election ballot that follows open-party. They then request an audit file that they can view to confirm the results were correct.

Priority: High. This is a high priority feature since it is a required feature of the system. The tester will want to see the audit file for the election.

4.3.2 Stimulus/Response Sequences

1. Tester opens program by starting it through command line.
2. Program prompts for an input file.
3. Tester writes name of an input file that matches the open-party ballot file pattern.
4. System reads in input file.
5. System calculates winner of election by counting votes, recording winner and process in audit file.

4.3.3 Functional Requirements

REQ-1: Ballot fed into program follows format of open-party.

REQ-2: Ballot filename exists.
REQ-3: Output file name is unique, ensuring not to overwrite already made audits.

4.4 User Views Election Results

4.4.1 Description and Priority

Description: After a user runs the software, calculated results of the election are displayed on the command line.

Priority: High. This is a high priority feature since it is a required feature of the system. The main function of the system is determining election winners, and as such this is required.

4.4.2 Stimulus/Response Sequences

1. Software calculates election results based on a provided input file.
2. Software displays calculated results on the command line for the user to view.

4.4.3 Functional Requirements

REQ-1: Ballot fed into program follows format of open-party or instant-runoff.
REQ-2: Ballot filename exists.

4.5 User Views Audit File

4.5.1 Description and Priority

Description: A user provides a file with the election information and run it. This will produce an audit file which the user can then view.

Priority: High. This is a high priority feature since it is a required feature of the system. The user running the election will want to see the audit of how the election was determined.

4.5.2 Stimulus/Response Sequences

1. User opens runs program through command line.
2. User is prompted for an election file for input.
3. User inputs election file.
4. Program processes input file and audit is outputted into the correct folder.
5. User can then go into and select audit file to view it.

4.5.3 Functional Requirements

REQ-1: Ballot fed into program follows format of open-party.
REQ-2: Ballot filename exists.
REQ-3: Output file name is unique, ensuring not to overwrite already made audits.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system requires most up-to-date CSE lab machine. The election program should be able to run 100,000 ballots in under 8 minutes.

5.2 Safety Requirements

No safety requirements.

5.3 Security Requirements

No security requirements because not connected to the internet. Although, security such as ensuring one vote for one person is handled at the voting centers.

5.4 Software Quality Attributes

The system provides various quality attributes like re-usability since the program will be run multiple times during the year at normal election times and special elections. Other attributes include accessibility, effectiveness, correctness, accuracy, reliability, and timeliness. In general, the system provides FURPS (Functionality, Usability, Reliability, Performance and supportability) model of software quality attributes.

5.5 Business Rules

Operating principles of the system include:

- Continual improvement in the system is essential to our success.
- Various election officials and testers check the accuracy of the program by checking the audit file produced by the system. This involvement of the Election Officials and testers is essential to improving the system.

6. Other Requirements

The system should be usable multiple times over the year at normal election times and special elections.

Appendix A: To Be Determined List

1. User documentation for Voting System.

Use Cases for Voting System

Daniel Rockcastle, John Caspers, Michael Birk, and Mukesh Pragaram Choudhary
October 8, 2018

Name	Election official requests audit file
ID	UC_001
Description	Testers produce an audit file which contains the information about the election. Election official requests this file from the testers to verify.
Users	Election officials
Organizational Benefits	Verify the election results for accuracy
Frequency of Use	Every time election official wants to verify the election results.
Triggers	Election official writes in the command line a file name of a file produced by the system.
Preconditions	System has already produced an audit file.
Postconditions	Election official can view the complete audit file.
Main Course	1. System produces an audit file using the file given and saves in the same directory. (EX1) 2. Election Official requests the file in the command prompt
Alternate Courses	AC1: System determines there was a tie. 1. System flips a coin to determine winner. 2. System outputs winner on screen and prints to audit file that a coin was flipped, how it was flipped, and the result.
Exceptions	EX1 Software is given invalid input file. 1. Software informs user that the given input file is invalid. 2. Software prompts user for an input file name again. 3. Return user to Main Course step 1.

Name:	Tester requests audit file - instant-runoff
ID:	UC_002
Description:	A tester inputs an election ballot that follows instant-runoff. They then request an audit file that they can view to confirm the results were correct.
Actors:	Testers
Organizational Benefits:	An audit file is the only way to confirm that election results are correct. In something like an election, accuracy of results is extremely important.
Frequency of Use:	Testers will use this functionality when the program is first being written and will also use it whenever any changes are made to the program.
Triggers:	A tester writes in the command line a file name of a file that follows instant-runoff voting.
Preconditions:	A tester has started the program and is on the opening screen that requests a file name.
Postconditions:	A unique audit file is outputted in the current directory. This audit file is ensured unique by the naming condition of Audit-systemClock.
Main Course:	1. Tester opens program by starting it through command line. 2. Program prompts for an input file. 3. Tester writes name of an input file that matches the instant-runoff ballot file pattern. 4. System reads in input file. 5. System calculates winner of election by counting votes, recording winner and process in audit file.
Alternate Courses:	AC1: System determines there was a tie. 3. System flips a coin to determine winner. 4. System outputs winner on screen and prints to audit file that a coin was flipped, how it was flipped, and the result.
Exceptions:	EX1: Input file does not exist. 1. System notifies tester that their input file does not exist. 2. System prompts tester for another input file. EX2: Output file already exists and thus is not unique. 1. System notifies tester that output file already exists. 2. System prompts tester if they want to overwrite old audit file. 3. If no, system restarts from Main Course 1.

Name:	Tester requests audit file - open-party
ID:	UC_003
Description:	A tester inputs an election ballot that follows open-party. They then request an audit file that they can view to confirm the results were correct.
Actors:	Testers
Organizational Benefits:	An audit file is the only way to confirm that election results are correct. In something like an election, accuracy of results is extremely important.
Frequency of Use:	Testers will use this functionality when the program is first being written and will also use it whenever any changes are made to the program.
Triggers:	A tester writes in the command line a file name of a file that follows open-party voting.
Preconditions:	A tester has started the program and is on the opening screen that requests a file name.
Postconditions:	A unique audit file is outputted in the current directory. This audit file is ensured unique by the naming condition of Audit-systemClock.
Main Course:	1. Tester opens program by starting it through command line. 2. Program prompts for an input file. 3. Tester writes name of an input file that matches the open-party ballot file pattern. 4. System reads in input file. 5. System calculates winner of election by counting votes, recording winner and process in audit file.
Alternate Courses:	AC1: System determines there was a tie. 1. System flips a coin to determine winner. 2. System outputs winner on screen and prints to audit file that a coin was flipped, how it was flipped, and the result.
Exceptions:	EX1: Input file does not exist. 1. System notifies tester that their input file does not exist. 2. System prompts tester for another input file. EX2: Output file already exists and thus is not unique. 1. System notifies tester that output file already exists. 2. System prompts tester if they want to overwrite old audit file. 3. If no, system restarts from Main Course 1.

Name	User views election results.
ID	UC_004
Description	After a user runs the software, calculated results of the election are displayed on the command line.
Actors	Election officials, testers, and programmers.
Organizational Benefits	Provides simple and easy-to-read election results for officials.
Frequency of Use	Election officials: every time an official wants to calculate the results of an election. Testers: every time a tester wants to confirm the software produces expected results in a specific case. Programmer: every time a programmer runs the software to confirm software behaves as desired.
Triggers	The user runs the software for a specific input file.
Preconditions	Input file is valid.
Postconditions	Results are displayed correctly.
Main Course	1. Software calculates election results based on a provided input file (see EX1, see AC1). 2. Software displays calculated results on the command line for the user to view.
Alternate Courses	AC1 Election is a tie. 1. Software informs user that the election resulted in a tie. 2. Software randomly chooses a winner. 3. Return user to Main Course step 2.
Exceptions	EX1 Software is given invalid input file. 1. Software informs user that the given input file is invalid. 2. Software prompts user for an input file name again. 3. Return user to Main Course step 1.

Name	User views audit file
ID	UC_005
Description	A user provides a file with the election information and run it. This will produce an audit file which the user can then view.
Actors	Programmers, Testers, Election Officials
Organizational Benefits	Audit file provides election information and shows how election progressed
Frequency of Use	Audit file can be viewed every time an election file is run
Triggers	User inputs a file for program and runs it.
Preconditions	Election file inputted has no errors and program produces audit file in correct folder
Postconditions	Unique Audit file is produced in the correct folder with correct election information.
Main Course	1. User opens runs program through command line 2. User is prompted for an election file for input. 3. User inputs election file (AC 1) 4. Program processes input file and audit is outputted into the correct folder. (EX 1) 5. User can then go into and select audit file to view it
Alternate Courses	AC 1 User inputs a non-election file 1. Program outputs an error message indicating a non-election file has been inputted 2. Return to Main Course 2
Exceptions	EX 1 Program fails to process file correctly 1. Indicate to the user an error has occurred processing the file. 2. Return to Main course 2.